Izbrana poglavja iz diskretne matematike

Zbornik seminarjev

Ljubljana, 2012

Kazalo

1 Pregled centralnostnih indeksov **9**

1.1 Pregled . . . 9

1.1.1 Centralnost vozlišč . . . 9

1.1.2 Centralnost povezav . . . 10

1.2 Centralnost . . . 11

1.2.1 Groba definicija . . . 11

1.2.2 Soseščina in razdalje . . . 12

1.2.3 Lokacijski problemi . . . 12

1.2.4 Strukturne lastnosti . . . 15

1.2.5 Najkrajše poti . . . 19

1.2.6 Vitalnost . . . 22

1.2.7 Električni tok . . . 25

1.2.8 Naključni procesi . . . 28

2 Algoritmi za indekse centralnosti **33**

2.1 Indeks centralnosti . . . 33

2.1.1 Primer uporabe . . . 33

2.1.2 Ohlapna definicija . . . 33

2.1.3 Sosedska centralnost . . . 34

2.1.4 Ekscentrična centralnost . . . 34

2.1.5 Bližinska centralnost . . . 35

2.1.6 Centralnost vmesnika na najkrajši poti . . . 35

2.1.7 Spletna centralnost . . . 36

2.2 Osnovni algoritmi . . . 39

2.2.1 Najkrajše poti . . . 39

2.2.2 Najkrajše poti med vsemi pari vozlišč . . . 40

2.2.3 Dinamični APSP . . . 42

2.2.4 Računanje največje lastne vrednosti . . . 42

2.3 Specifični centralni algoritmi . . . 42

2.3.1 Indeks centralnosti vmesnika na najkrajši poti . . . 42

2.3.2 Shortcut indeks . . . 45

2.4 Hitri približki algoritmov . . . 46

2.4.1 Aproksimacija centralnosti, ki temeljijo na izračunu vseh parov najkrajših potih . . . 47

2.4.2 Aproksimacija spletnih centralnosti . . . 49

2.4.3 Zaključek razdelka . . . 56

3 Napredni koncepti centralnosti **57**
3.1 Normalizacija . . . 58
3.1.1 Primerjava centralnosti elementov znotraj grafa . . . 58
3.1.2 Primerjava centralnosti elementov med različnimi grafi . . . 59
3.2 Personalizacija . . . 60
3.2.1 Personalizacija centralnosti, ki temeljijo na razdalji in najkrajših poteh . . . 60
3.2.2 Personalizacija spletnih centralnosti . . . 61
3.3 Štiri dimenzije centralnega indeksa . . . 64
3.3.1 Osnovni izraz . . . 65
3.3.2 Operator izraza . . . 66
3.3.3 Personalizacija . . . 66
3.3.4 Normalizacija . . . 66
3.3.5 Neodvisnost dimenzij . . . 67
3.3.6 Konstrukcija centralnega indeksa . . . 67
3.4 Aksiomatizacija . . . 69
3.4.1 Aksiomatizacija vozliščne centralnosti glede na razdaljo . . . 69
3.4.2 Aksiomatizacija povratne centralnosti . . . 71
3.5 Stabilnost in občutljivost . . . 76
3.5.1 Stabilnost centralnosti, ki obravnavajo razdaljo . . . 76
3.5.2 Stabilnost in občutljivost spletnih centralnosti . . . 78
3.5.3 Stabilnost razvrščanja . . . 81

4 Lokalna gostost **85**
4.1 Popolnoma goste grupe - klike . . . 86
4.1.1 Reševanje problemov . . . 88
4.1.2 Iskanje globalno maksimalnih klik . . . 88
4.1.3 Aproksimiranje velikosti maksimalnih klik . . . 91
4.1.4 Iskanje klik s fiksno velikostjo . . . 91
4.1.5 Štetje maksimalnih klik . . . 93
4.2 Strukturno goste množice . . . 97
4.2.1 Plexi . . . 97
4.2.2 Jedra . . . 100
4.3 Statistično goste grupe . . . 102
4.3.1 Gosti podgrafi . . . 103
4.3.2 Najgostejši podgrafi . . . 105
4.3.3 Najgostejši podgrafi dane velikosti . . . 109
4.3.4 Parametrizirana gostota . . . 111
4.4 Poglavje opomb . . . 111

5 Povezanost **115**
5.1 Osnovni izreki . . . 116
5.2 Uvod v minimalne prereze . . . 119
5.3 Minimalni prerezi vseh parov točk . . . 119
5.4 Lastnosti minimalnega prereza v neusmerjenih grafih . . . 120
5.5 Predstavitev minimalnih prerezov s kaktusom . . . 127
5.6 Algoritmi na grafih: povezanost na podlagi pretoka . . . 128
5.6.1 Algoritmi na grafih: povezanost po točkah . . . 129

5.6.2 Algoritmi na grafih: povezanost po povezavah . . . 130
5.7 Algoritmi, ki ne temeljijo na pretoku . . . 133
5.7.1 Algoritem: Minimalni prerez (Stoer in Wagner) . . . 133
5.8 2-povezane komponente . . . 135
5.9 Močno povezane komponente . . . 136
5.10 3-povezane komponente . . . 138

6 Gručavost **141**
6.1 Osnovni pojmi . . . 142
6.2 Merjenje kakovosti gručavosti . . . 143
6.2.1 Splošna oblika indeksov . . . 143
6.2.2 Pokritost . . . 144
6.2.3 Prevodnost . . . 146
6.2.4 Zmogljivost . . . 149
6.3 Metode za reševanje problema gručavosti . . . 151
6.3.1 Požrešna metoda . . . 153
6.3.2 Proces združevanja . . . 153
6.3.3 Proces delitve . . . 155
6.3.4 Premična metoda . . . 157
6.3.5 Splošni optimizacijski pristopi h grupiranju . . . 158
6.4 Algoritmi za grupiranje . . . 160
6.4.1 Vhodi za proces združevanja . . . 160
6.4.2 Vhodi za proces delitve . . . 161
6.4.3 Nestandardni vhodi za procesa delitve in združevanja . . . 162
6.5 Gručavost - alternativni pristopi . . . 163
6.5.1 Prehodna gručavost . . . 163
6.5.2 Gručavost s predstavnikom . . . 164
6.5.3 Gnezdena gručavost . . . 165
6.6 Aksiomatika . . . 165
6.7 Zaključek . . . 167

7 Dodelitve vlog **169**
7.1 Uvod . . . 169
7.1.1 Oznake in definicije . . . 170
7.1.2 Graf vlog . . . 170
7.2 Strukturna ekvivalenca . . . 171
7.2.1 Mreža ekvivalenčnih relacij . . . 172
7.2.2 Mreža strukturnih ekvivalenc . . . 173
7.2.3 Izračun strukturnih ekvivalenc . . . 173
7.3 Regularna ekvivalenca . . . 175
7.3.1 Osnovne lastnosti . . . 175
7.3.2 Struktura mreže in regularna notranjost . . . 177
7.3.3 Izračun regularne notranjosti . . . 179
7.3.4 Problem dodelitve vlog . . . 185
7.3.5 Obstoj k-dodelitev vlog . . . 187
7.4 Ostale ekvivalenčne relacije . . . 188
7.4.1 Ekstaktna dodelitev vlog . . . 188
7.4.2 Avtomorfne in orbitne ekvivalenčne relacije . . . 190

7.4.3 Popolna ekvivalenčna relacija . . . 190
7.4.4 Relativna regularna ekvivalenčna relacija . . . 191
7.5 Grafi z več relacijami . . . 192
7.6 Polgrupa grafa . . . 194
7.6.1 Winship-Pattisonova ekvivalenca vlog . . . 197

8 Omrežne statistike 199
8.1 Uvod . . . 199
8.2 Lastnosti omrežnih statistik . . . 199
8.3 Porazdelitev stopenj vozlišč . . . 200
8.4 Razdalja . . . 201
8.4.1 Povprečna ali karakteristična razdalja . . . 202
8.4.2 Radij, diameter in ekscentričnost . . . 202
8.4.3 Okolice točk . . . 203
8.4.4 Efektivna ekscentričnost in diameter . . . 203
8.4.5 Algoritmi . . . 204
8.5 Število najkrajših poti . . . 204
8.6 Koeficient popačenja . . . 206
8.7 Koeficient gručavosti in tranzitivnost . . . 206
8.7.1 Osnovni pojmi in definicije . . . 206
8.7.2 Odnos med tranzitivnostjo in koeficientom gručavosti . . . 207
8.7.3 Izračun koeficineta gručavosti in tranzitivnosti . . . 208
8.7.4 Aproksimacija števila ciklov dolžine tri . . . 208
8.8 Omrežni motivi . . . 210
8.9 Vrste omrežnih statistik . . . 210
8.9.1 Transformacije omrežnih statistik . . . 211
8.9.2 Vizualizacija . . . 215
8.9.3 Vzorčenje lokalnih mer . . . 217
8.10 Zaključek . . . 217

9 Primerjava omrežij 219
9.1 Izomorfizem grafov . . . 219
9.1.1 Iskanje izomorfizma . . . 221
9.1.2 Preprost algoritem sestopanja . . . 222
9.1.3 McKay-ev Nauty algoritem . . . 223
9.1.4 Zahtevnost iskanja izomorfizmov oziroma kako prelisičiti nauty algoritem . . . 230
9.2 Podobnost grafov . . . 232
9.2.1 Urejevalna razdalja . . . 233
9.2.2 Razlika v dolžini poti . . . 234
9.2.3 Največji skupni podgrafi . . . 236

10 Spektralna teorija grafov 239
10.1 Temeljne lastnosti . . . 239
10.1.1 Spekter grafa . . . 239
10.1.2 Laplaceova matrika . . . 244
10.1.3 Normalizirana Laplaceova matrika . . . 246
10.1.4 Primerjava spektrov . . . 246

10.1.5 Primeri spektrov . . . 248
10.2 Numerične metode . . . 249
10.2.1 Metode za izračun lastnih vrednosti majhnih polnih matrik . . . 249
10.2.2 Metode za izračun nekaj lastnih vrednosti velike razpršene matrike . 250
10.3 Podgrafi in operacije na grafih . . . 251
10.3.1 Izrek o prepletanju . . . 252
10.3.2 Grafting . . . 254
10.3.3 Operacije na grafih . . . 255
10.4 Meje parametrov grafa . . . 256
10.4.1 Dokaz izreka 10.33 . . . 264

11 Robustnost in odpornost 271
11.1 Uvod . . . 271
11.2 Najslabši primer merila: povezanost . . . 272
11.2.1 Klasična povezanost . . . 272
11.2.2 Kohezivnost . . . 272
11.2.3 Minimalna m-stopnja . . . 273
11.2.4 Žilavost (Toughness) . . . 274
11.2.5 Pogojna povezanost . . . 276
11.3 Najslabši primer merila: razdalja . . . 276
11.3.1 Vzdržljivost . . . 277
11.3.2 Naraščajoča razdalja in zaporedje premerov . . . 277
11.4 Povprečno merilo robustnosti . . . 279
11.4.1 Povprečna povezanost . . . 279
11.4.2 Povprečna razdalja povezave in razdrobitev . . . 280
11.4.3 Odpornost na uravnotežen rez . . . 283
11.4.4 Učinkovit premer (Effective diameter) . . . 284
11.5 Verjetnostno merilo robustnosti . . . 284
11.5.1 Zanesljivostni polinom . . . 285
11.5.2 Verjetnostna odpornost . . . 286

12 Risanje velikih grafov s pomocjo linearne algebre 289
12.1 Definicije . . . 289
12.2 Klasični MDS . . . 290
12.3 Pivot MDS . . . 292
12.3.1 Izbira pivotov . . . 293
12.4 Implementacija . . . 293

13 Hevristike za Identifikacijo Grafov 295
13.1 Naključni Grafi . . . 295
13.2 Nove Grafovske Statistike . . . 298
13.3 Spektralne Lastnosti Naključnih Grafov . . . 299

14 Pajek 303
14.1 Uvod . . . 303
14.2 Podatkovne strukture . . . 305
14.3 Omrežje . . . 305
14.3.1 Zapis omrežja v formatu Pajek networks . . . 306

14.3.2 Grafični prikaz omrežja . . . 307
14.3.3 Grafično (interaktivno) ustvarjanje omrežja . . . 309
14.3.4 Generiranje naključnega omrežja . . . 310
14.3.5 Operacije . . . 310
14.4 Razbitje . . . 312
14.4.1 Operacije . . . 312
14.5 Vektor . . . 315
14.5.1 Operacije . . . 317
14.6 Primeri uporabe . . . 319
14.6.1 Delo s razbitji . . . 319
14.6.2 Delo z vektorji . . . 321
14.7 Zaključek . . . 321

15 Omrežne strukture v resničnem svetu in njihovi mehanični omrežni modeli 323
15.1 Omrežne strukture v resničnem svetu . . . 323
15.1.1 Mali svetovi . . . 323
15.1.2 Brez-lestvična omrežja . . . 325
15.1.3 Robustnost naključnega in brez lestvičnega omrežja . . . 326

Poglavje 1

Pregled centralnostnih indeksov

Matjaž Krnc

1.1 Pregled

Centralnostni indeksi so kvantitativna mera, da so nekatera vozlišča oz. povezave v omrežju pomembnejše od drugih. Glede na različne interpretacije pomembnosti obstaja mnogo različnih definicij centralnosti, prve definicije pa so se pojavile že v letih 1950, npr. Baveliasov indeks [50,51], centralnost stopnje [483], centralnost odmeva [510], ter druge. Videli bomo, da ni vsak indeks primeren za vsako aplikacijo na danem omrežju. Ogledali si bomo nekaj najpomembnejših klasičnih primerov centralnostnih indeksov.

V nadaljevanju si bomo ogledali zglede, ki nam bodo orisali različne pristope do centralnosti.

1.1.1 Centralnost vozlišč

V tem zgledu si bomo ogledali tri uporabe različnih iterpretacij centralnosti. Šolski razred s 30 dijaki mora izbrati razrednega predstavnika in vsak dijak lahko voli za enega izmed sošolcev. Iz te situacije lahko zgradimo različne grafe, ki jih nato tudi obravnavamo z različnimi indeksi centralnosti.

Omrežje volilnega rezultata

Volitve so pomemben dogodek v vsaki družbi, z njimi se srečujemo precej pogosto. Seveda so tu nekateri ljudje oz. politične skupine vplivnejše od drugih. Vprašanje je, kako bi lahko izmerili pomembnost, ki smo jo lahko intuitivno opazimo.

Najprej si poglejmo omrežje, ki direktno predstavlja volilni rezultat. Definirajmo graf:

Vozlišča: Vsi dijaki.

Povezave: Usmerjena povezava $\overrightarrow{AB}$ obstaja natanko v primeru, ko je dijak A volil dijaka B.

V tej situaciji lahko rečemo, da je dijak, za katerega je volilo več sošolcev bolj pomemben. To centralnost bomo kasneje poimenovali tudi kot *centralnost vhodne stopnje.*

Vplivnostno omrežje

Na isto skupino ljudi, lahko pogledamo tudi drugače. Rečemo lahko, da je nek dijak, ki prepriča veliko sošolcev, da volijo za njegovega favorita, vplivnejši kot nek drug dijak, čigar mnenje nikjer nič ne šteje. Tako lahko definiramo drugačen graf:

Vozlišča: Vsi dijaki.

Povezave: Usmerjena povezava $\overrightarrow{AB}$ obstaja natanko v primeru, ko je dijak A prepričal dijaka B, da je volil za njegovega najboljšega kandidata.

Tako omrežje imenujemo vplivnostno omrežje. Denimo, da je naš razred glede na najljubšega kandidata v glavnem razdeljen na dve skupini X ter Y. Če neka oseba A iz skupine X uspe prepričati velik del ljudi iz skupine Y da spremenijo mnenje, ter volijo kandidata skupine X, lahko rečemo da je oseba A precej pomembna, saj posredno precej vpliva na končni rezultat volitev, opisan v prejšnjem primeru. Centralnostnim indeksom, ki temeljijo na tej intuiciji, bom kasneje rekli tudi *vmesnostna centralnost (betweenes centrality).*

Socialno omrežje

Na naš razred lahko pogledamo kot klasično socialno mrežo prijateljstev. Definirajmo graf:

Vozlišča: Vsi dijaki.

Povezave: Neusmerjena povezava $\overrightarrow{AB}$ obstaja natanko v primeru, ko je dijak A prijatelj dijaka B.

Če je nek dijak večinoma prijatelj pomembnejših oseb, s tem tudi sam postane pomembnejši. V nasprotju prijateljstvo nepomembnega dijaka ni tako pomembno in ne doprinese k pomembnosti. Opazimo lahko, da povečana pomebnost nekega dijaka kakor odmev vpliva na pomembnosti njegovih sosedov itn.; centralnostnim indeksom, ki temeljijo na tej intuiciji bomo kasneje rekli *odmevna centralnost (feedback centrality).*

Zaključek

Vsi zgornji primeri potrjujejo, da ne obstaja centralnostni indeks, ki bi ustrezal vsem situacijam, ter da se lahko neko konkretno omrežje učinkovito analizira na več popolnoma različih načinov, glede na različna vprašanja, na katera želimo odgovoriti. V analogiji z zgornjimi primeri lahko podobno intuicijo prenesemo tudi na povezave. O tem govori naslednji razdelek.

1.1.2 Centralnost povezav

Podobno kot pri vozliščih, lahko tudi pri povezavah vidimo, da so nekatere pomembnejše od drugih. Za lažjo predstavo lahko za primer vzamemo kar ogrodije internetnega omrežja, kjer so nekatere povezave (prekooceanske, optične) očitno mnogo pomembnejše kot druge

povezave (do končnih uporabnikov). V osnovi ločimo dva različna pristopa za določanje centralnosti povezav.

1. V prvem primeru štejemo za določeno povezavo neko število struktur, ki to povezavo vsebujejo (št. najkrajših poti,...). Tak primer je na primer vmesnostna centralnost povezav, ki jo bomo opisali kasneje.

2. V drugem primeru opazujemo, kako se nek parameter spremeni, če izbrano povezavo odstranimo. Taka centralnost je npr. pretočni vmesnostni vitalnostni indeks.

Vidimo lahko, da nam lahko različni pogledi na centralnost odprejo mnogo pogledov na neko situacijo v omrežju, ter nam pomagajo pri njeni analizi. Pomembno je, da se zavedamo, da noben izmed pristopov, ki jih bomo opisali v splošnem ni pomembnejši ali boljši od drugih. Vsak izmed pristopov je primeren le za kakšno izmed različnih postavljenih vprašanj.

1.2 Centralnost

Preden se lotimo konkretnega opisovanja različnih pristopov do centralnosti, se spodobi, da na grobo orišemo nekaj skupnih lastnosti, ki jih bomo od vsakega indeksa centralnosti zahtevali. Kot smo videli, obstaja mnogo zelo različnih, a smiselnih intuicij za merjenje centralnosti v grafu, zato je točno definicijo centralnosti zelo težko določiti, v zgodovini pa tudi ne obstaja splošno sprejeta definicija o tem, kaj je centralnostni indeks. Kljub temu lahko nekaj grobih skupnih točk vendarle najdemo.

1.2.1 Groba definicija

Najprej se spomnimo, da sta si dva grafa $G = (V_1, E_1)$ ter $H = (V_2, E_2)$ izomorfna, če obstaja bijekcija $\phi : V_1 \to V_2$, tako da za vsako povezavo (u, v) iz E_1, velja $(u, v) \in E_1 \leftrightarrow (\phi(u), \phi(v)) \in E_2$.

Za vsak centralnosti indeks želimo, da je odvisen le od strukture grafa, poleg tega pa želimo, da je centralnost izražena v nekem realnem številu. Temu primerno bomo rekli, da je indeks centralnosti *strukturni indeks*.

Definicija 1.1 (strukturni indeks) Naj bo $G = (V, E)$ utežen, usmerjen ali neusmerjen multigraf in naj bo X množica njegovih vozlišč (povezav). Realni funkciji $s_G : X(G) \to \mathbb{R}$ pravimo *strukturni indeks* natanko tedaj, ko je izpolnjen naslednji pogoj:

$$(\forall x \in X) \wedge (G \simeq H) \Longrightarrow s_G(x) = s_H(\phi(x))$$

kjer funkcija ϕ predstavlja izomorfizem med G in H.

Iz definicije je razvidno, da je se centralnost nekih grafovskih struktur znotraj izomorfizmov ohranja. Kljub temu, da je vsak centralnostini indeks tudi strukturni indeks, velja pri tem opozoriti, da vsak strukturni indeks še ni centralnostni indeks. Pri obravnavi centralnoh indeksov se bomo med drugim poskušali držati *načela monotonosti*, da pomembnejša vozlišča (povezave) dobijo v funkciji centralnosti večjo vrednost od manj pomembnih vozlišč (povezav).

1.2.2 Soseščina in razdalje

V tem razdelku bomo opisali osnovne ideje o indeksih centralnosti, ki izhajajo iz soseščine oz. razdalje med posameznimi vozlišči oz. merijo dosegljivost nekega vozlišča. Najprej si oglejmo najosnovnejšega izmed centralnostnih indeksov – indeks stopnje.

Stopnja

Seveda je najenostavnejše merilo za centralnost kar stopnja točke, ki jo bomo označili z c_D. Opis; uporaba; dopolnilo za usmerjene grafe.

Definicija 1.2 (centralnost glede na stopnjo) Naj bo $G(V, E)$ enostavni graf in naj bo $x \in V(G)$ poljubno njegovo vozlišče. Potem velja $c_D = d(x)$. Naj bo $H(V, E)$ usmerjeni graf in naj bo $x \in V(H)$ poljubno njegovo vozlišče. Potem velja $c_{iD} = d^-(x)$, ter $c_{oD} = d^+(x)$.

Opisali smo že, da te vrste centralnosti intuitivno predstavljajo rezultate volitev, saj je volilni rezultat določen le s številom glasov.

1.2.3 Lokacijski problemi

Lokacijski problemi so klasični optimizacijski problemi, ki se zelo pogosto pojavljajo v mnogo situacijah v industriji ter ekonomiji. Lokacija nekega objekta se zaradi različnih komunikacijskih, transportnih, ter drugih pogojev namreč zelo pogosto zelo skrbno izbere, pri čemer se upošteva mnogo kriterijev.

Obstaja veliko načinov klasifikacije različnih lokacijskih problemov. Hakami [271], je lokacijske probleme razdeliv v dve veliki družini, pri čemer bomo tu mi dodali še tretjo družino. Prva družina je družina problemov, ki v svojem bistvu uporabijo min-max kriterij za določanje optimalnosti. Kot primer lahko vzamemo že iskanje optimalne lokacije za novo bolnišnico na nekem območju, pri čemer moramo minimizirati maksimalno razdaljo do vseh prebivalcev, ki živijo na nekem podanem območju. Druga družina lokacijskih problemov obravnava tiste probleme, ki v svojem bistvu uporabljajo kriterij minimalne vsote. Primer iz te družine je na primer lokacija novega trgovskega centra, ki želi minimizirati skupno razdaljo vseh potencialnih strank na nekem območju. Tretja družina teh problemov se ukvarja z lokacijo nekega poslovnega objekta, ki deluje v konkurenčnem okolju. Namen optimizacijskih problemov iz te družine je predvideti tržni delež, ki ga na neki lokaciji poslovni objekt zajame, ter, glede na konkurenco, maksimizirati svoj tržni delež.

Glede na predstavljene tri družine lokacijskih problemov bomo v tem razdelku spoznali tri pomembne centralnostne indekse, ki nam bodo merili pomembnost določenih lokacij, glede na izbran problem.

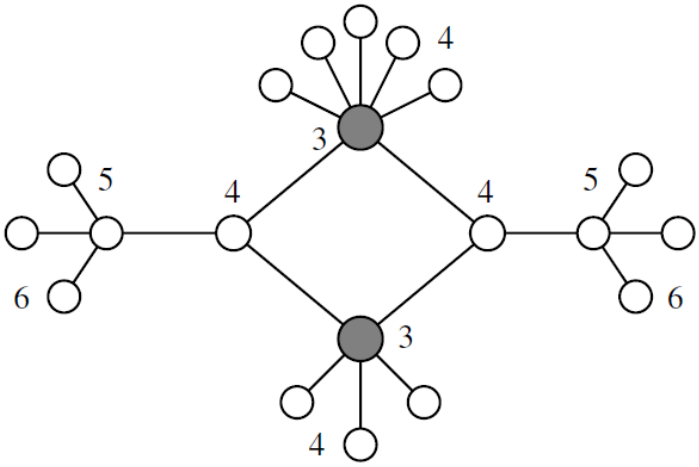
Ekscentričnost

Primer lokacijskega problema iz prve družine je naslednji:

Iščemo lokacijo nove bolnišnice, ki jo lahko postavimo na poljubno lokacijo v našem omrežju. Pri tem bi seveda radi minimizirali največjo razdaljo do vseh prebivalcev na območju, ki ga bo bolnišnica pokrivala. Pri vsaki konkretni lokaciji lahko izračunamo razdaljo do najbolj oddaljenega prebivalca regije. Ta primer zajame intuicijo računanja ekscentričnosti za poljubno vozlišče v grafu ter ga bomo podrobneje predstavili kasneje.

Definicija 1.3 Naj bo $G(V, E)$ poljubni graf. *Ekscentričnost* poljubnega vozlišča $u \in V(G)$ je definirana kot

$$e(u) = \max_{v \in V(G)} \{d(u, v)\}.$$



Slika 1.1: Postavitev nove bolnišnice na optimalno lokacijo. Na sliki je za vsako lokacijo izračunana razdalja do najbolj oddaljenega vozliščav grafu, vozlišči z najboljšo ekscentričnostjo pa sta povdarjeni.

Hage in Harary sta zgornjo ekscentričnost preoblikovala, tako da je v skladu z načelom monotonosti iz 1.2.1. Inverzna vrednost ekscentričnosti nam porodi novo centralnost:

Definicija 1.4 (centralnost glede na ekscentričnost)
Naj bo $G(V, E)$ poljubni graf. Potem za poljubno vozlišče $u \in V(G)$ velja:

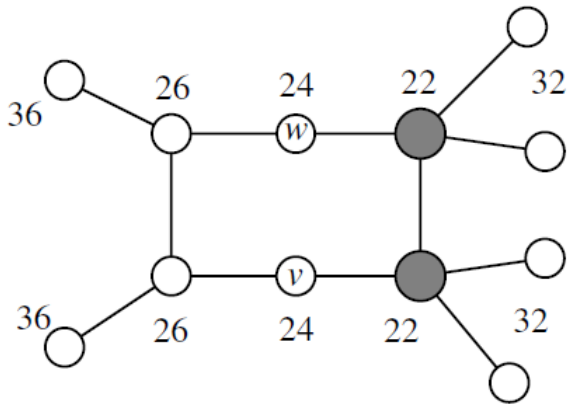
$$c_E(u) = \frac{1}{e(u)} = \frac{1}{\max_{v \in V(G)} \{d(u, v)\}}.$$

Bližina

Primer lokacijskega problema iz druge družine (glede na Hakami [271]) je naslednji:

Iščemo lokacijo nove trgovine. Pri tem bi radi minimizirali skupno razdaljo od trgovine do vseh prebivalcev na območju, ki ga bo trgovina pokrivala. Pri vsaki konkretni lokaciji lahko izračunamo vsoto vseh razdalj do vseh prebivalcev regije. Ta primer zajame intuicijo računanja bližinskega indeksa centralnosti za poljubno vozlišče v grafu.

V socialnih omrežjih se indeks centralnosti, ki temelji na tej intuiciji imenuje *bližina*. Tako lahko npr. za neko osebo merimo skupno razdaljo do vseh ostalih članov omrežja. Ljudje z majhno skupno razdaljo bodo pomembnejši od tistih, ki imajo skupno razdaljo večjo. V skladu s to intuicijo je nastalo več centralnih indeksov, ki hkrati upoštevajo tudi načelo monotonosti (glej 1.2.1). Tu bomo predstavili dva najpogostejša, pri čemer je prvi primer le inverzija zgoraj opisane skupne razdalje. V obeh primerih bo $G(V, E)$ poljubni graf ter $u \in V(G)$ njegovo poljubno vozlišče.



Slika 1.2: Postavitev nove trgovine na optimalno lokacijo. Na sliki je za vsako lokacijo izračunana skupna vsota razdalj do vseh preostalih vozlišč v grafu, vozlišči z najboljšo centralnostjo glede na bližino pa sta povdarjeni.

Definicija 1.5 (centralnost glede na ekscentričnost)

$$c_C(u) = \left(\sum_{v \in V(G)} d\left(u, v\right) \right)^{-1}$$

Drugo definicijo sta uvedla Valente in Foreman [558]. Ta mera je še nekoliko zanimivejša, saj nam meri, kako je neko vozlišče integrirano v omrežje.

Definicija 1.6 (Foreman, Valente)

$$c_R(u) = \frac{\sum_{v \in V}(D(G) - d(u, v) + 1)}{n - 1},$$

$n = |V(G)|$, kjer je $D(G)$ diameter grafa.

Trditev 1.7 *V poljubnem grafu za poljubno vozlišče u velja $c_R(u) > 0$.*

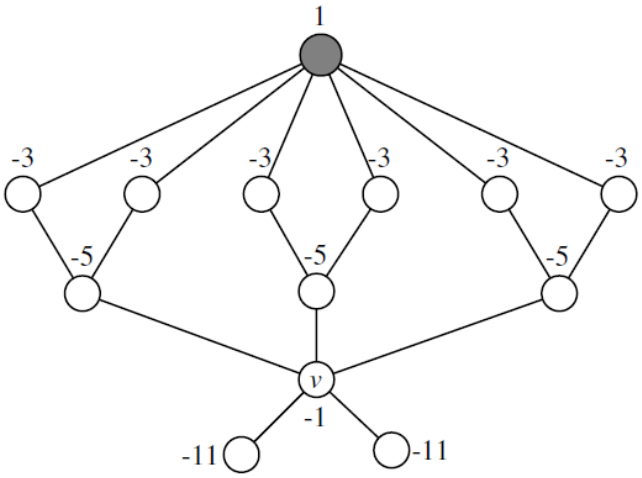
Dokaz. Ker je imenovalec vedno pozitiven, je dovolj videti, da je vsak izmed sumandov v vsoti $\sum_{v \in V}(D(G) - d(u, v) + 1)$ vedno večji od nič.

To očitno drži, saj za pojuben par $u, v \in V(G)$ vedno velja $d(u, v) \leq D(G)$.

□

Centroidne vrednosti

Zgornji primer obravnava okolje, ki ima znotraj nekega območja le eno trgovino, v realnosti pa vemo, da temu ponavadi ni tako. Pogosto je pojavi vsaj ena konkurenčna trgovina, ki ponuja enake storitve. Zamislimo si naslednjo poenostavljeno situacijo: Nek trgovec si izbere lokacijo za svojo trgovino, pri tem pa ve, da njegov tekmec spremlja njegove odločitve ter bo temu primerno izbral lokacijo svoje konkurenčne trgovine. Pri tem sta obe trgovini prebivalcem enako privlačni, ter se bodo odločili za najbližjo. Katero lokacijo (oz. katero vozlišče) naj trgovec izbere?



Slika 1.3: Graf z enim pozitivnim centroidnim vozliščem, ki je povdarjeno. Opazimo lahko, da je glede na centralnot bližine najboljše vozlišče v.

Definicija 1.8 Naj bo G povezan graf na n vozliščih in naj bosta u, v poljubni različni vozlišči v G. Potem:

$$\gamma_u(v) = |\{z; z \in V(G), d_G(z,u) > d_G(z,v)\}| .$$

Naj predstavlja u lokacijo našega trgovca, v pa lokacijo njegovega tekmeca. Torej bo naš trgovec zajel $\gamma_u(v) + \frac{1}{2}(n - \gamma_u(v) - \gamma_v(u)) = \frac{n}{2} + \frac{1}{2}(\gamma_u(v) - \gamma_v(u))$ strank. Njegov tekmec bo seveda izbral njemu optimalno lokacijo, ki minimizira $\gamma_u(v) - \gamma_v(u)$. Naš trgovec to strategijo pozna ter lahko za vsako vozlišče izračuna c_F, ki je hkrati tudi naša nova mera za centralnost.

Definicija 1.9 Naj bo G poljuben povezan graf. Vrednosti $c_F(u)$ pravimo centroidna vrednost vozčlišča in je definirana kot:

$$c_F(u) = \min_{v \in V(G) \setminus u} \{\gamma_u(v) - \gamma_v(u)\} .$$

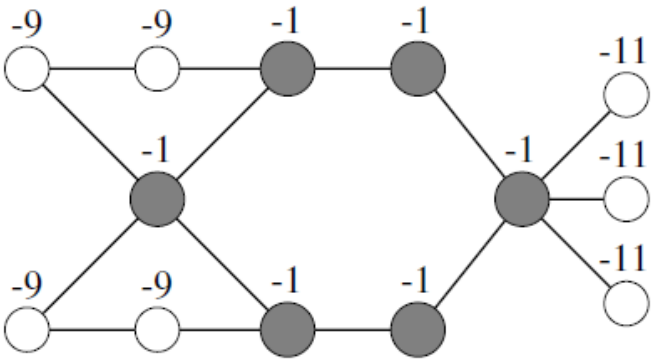
Vrednost $c_F(u)$ torej meri prednost lokacije u v primerjavi z drugimi lokacijami. Najboljša izbira je torej lokacija z največjo centroidno vrednostjo.

1.2.4 Strukturne lastnosti

Centralnostni indeksi vozlišč, ki smo jih obravnavali v prejšnjem razdelku nam včasih porodijo tudi smiselne strukturne indekse celotnega grafa. Najprej pa (glede na podan indeks centralnosti) definirajmo množico vozlišč z največjo centralnostjo.

$$S_c(G) = \left\{ u \in V(G) : c(u) = \max_{v \in V(G)} \{c(v)\} \right\},$$

kjer je c katerakoli izmed obravnavanih definicij centralnosti.



Slika 1.4: Centroidne vrednosti vseh vozlišč v zgornjem grafu so negativne. V tem primeru, je trgovec, ki lokacijo izbira prvi na slabšem.

Center grafa

V definiciji 1.3 smo ekcentričnost definirali kot $e\left(u\right) = \max_{v \in V(G)} \left\{d\left(u, v\right)\right\}$. Spomnimo se, da smo pri iskanju vozlišča z najmanjšo ekscentričnostjo rešili lokacijski problem optimalne lokacije za novo bolnišnico. V teoriji grafov temu minimumu pravimo *radius* grafa.

Definicija 1.10 (Radius) Naj bo G poljuben graf. Potem je njegov *radius* definiran kot:

$$r(u) = \min_{u \in V(G)} \left\{ \max_{v \in V(G)} \left\{d_G(u, v)\right\} \right\} = \min_{u \in V(G)} \left\{e(u)\right\}$$

S pomočjo radiusa lahko zdaj definiramo še *center* grafa.

Definicija 1.11 Naj bo G poljuben graf. Potem je njegov *center* definiran kot:

$$C(G) = \left\{u \in V(G) : r(G) = e(u)\right\}.$$

Ni težko videti, da za c_E velja $S_{c_E} = C(G)$, malo težje pa je videti , kje se ta center nahaja. Osnovni izrek o centru grafa za drevesa iz [336] pravi:

Izrek 1.12 *Za poljubno drevo T velja $|C(T)| \leq 2$.*

Dokaz. Izrek bomo dokazali z indukcijo. Če je naše drevo na dveh ali na eni točki, izrek očitno drži. Naj bo T poljubno drevo, ter T' manjše drevo, ki ga iz T dobimo, če mu odstranimo vse liste. Pokazali bomo, da ima drevo T enaka centralna vozlišča kot drevo T'. Vozlišču v, za katerega velja $d(u, v) = e(u)$, bomo rekli ekscentrično vozlišče vozlišča u; $u, v \in V(T)$. Ni težko videti, da je ekscentrično vozlišče lahko le list. Ker je ekscentričnost vsakega vozlišča v T' za ena manjša od ekscentričnosti istega vozlišča v T, imata T ter T' enak center. Ko ta postopek nadaljujemo, na koncu pridemo do poddrevesa na dveh ali eni točki.

□

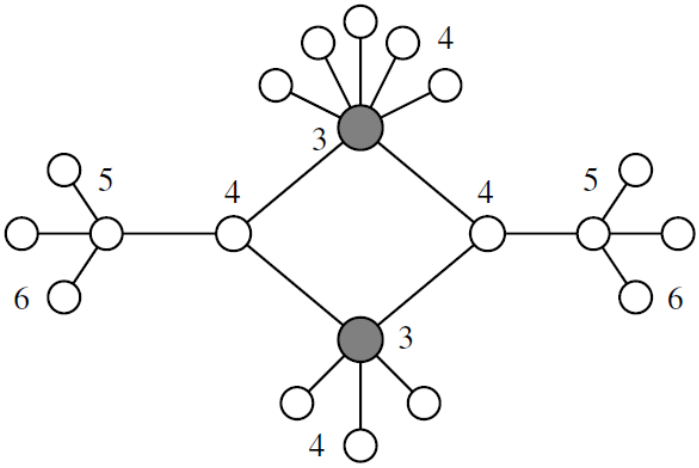
Izrek 1.13 *Naj bo G povezan neusmerjen graf. Potem v G obstaja po povezavah $2-$povezan podgraf, ki vsebuje vsa vozlišča iz $C(G)$.*

Dokaz. Denimo najprej, da izrek ne drži. Torej v G obstaja taka prerezna točka $u \in V(G)$, tako da graf $G - u$ razpade na podgrafa G_1 ter G_2, ki oba vsebujeta vsaj po eno centralno vozlišče. Naj bo v ekscentrična točka od u, torej taka točka, da je najkrajša pod P med u in v dolga $e(u)$. Brez škode za splošnost velja $u \in G_2$. Vemo, da v G_1 obstaja vsaj ena centralna točka w, ki ne leži na P. V naslednjem nizu neenakosti si oglejmo najkrajšo pot med w in v, pri čemer opazimo, da mora ta pot potekati skozi u.

$$e(w) \geq d(w,u) + d(u,v) \geq d(w,u) + e(u)$$

Torej w ne moree biti centralno vozlišče, kar privede do protislovja.

□



Slika 1.5: Na zgornjem grafu ni težko najti 2−povezanega podgrafa, ki vsebuje obe centralni vozlišči.

Mediana grafa

V 1.2.3 smo obravnavali centralnost glede na bližino, kjer smo za izbrano vozlišče določili skupno razdaljo do vseh preostalih vozlišč. Označimo minimum po skupni razdalji poljubnega vozlišča v G s preostalimi z $s(G) = \min_{u \in V(G)} \{s(u)\}$, pri čemer je $s(u) = \sum_{v \in V(G)} d(u,v)$. Zdaj lahko definiramo mediano grafa kot množico vozlišč, kjer je najmanjša vrednost funkcije s dosežena.

Definicija 1.14 (mediana grafa)
Naj bo G poljuben graf. *Mediana* grafa G je definirana kot:

$$M(G) = \{u \in V(G) : s(u) = s(G)\}.$$

Ni težko videti, da velja $M(G) = S_{c_C}(G)$.

Truszczynski [552] je preučeval mediano grafov za povezane neusmerjene grafe, ter prišel do naslednjega rezultata.

Izrek 1.15 *Mediana povezanega neusmerjenega grafa G leži v nekem 2 po povezavah povezanem podgrafu grafa G.*

Zgornji izrek porodi naslednjo posledico.

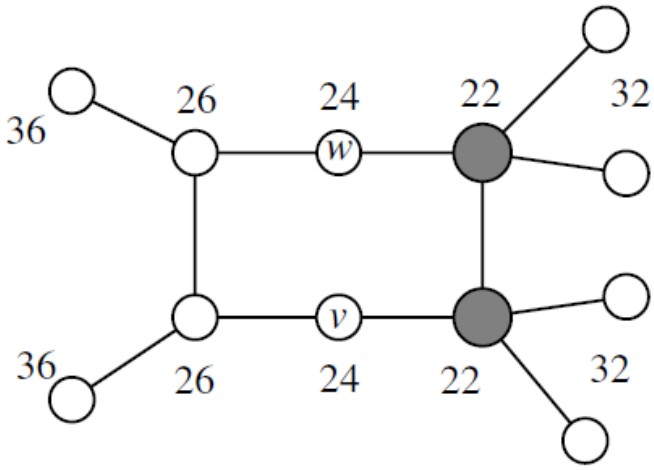
Posledica 1.16 *Mediano drevesa sestavljata dve sosednji vozlišči, ali pa eno samo vozlišče.*

Dokaz. Naj bo T poljubno drevo ter denimo najprej, da je $u \in V(T)$ list, ki je hkrati tudi mediana, ter naj bo v edini sosed od u. Potem velja $s(v) = s(u) - (|V(T)| - 2) \leq s(u)$, protislovje. Vidimo torej, da mediana ne more biti list, razen v primeru, ko $|V(T)|-2 = 0$.

Izrek bomo dokazali z indukcijo. Če je naše drevo na dveh ali na eni točki, izrek očitno drži. Naj bo T' drevo, ki ga iz T dobimo, če mu odstranimo vse liste. Pokazali bomo, da ima drevo T enaka medianska vozlišča kot drevo T'. Ker so vsa medianska vozlišča vsebovana v T', lahko brez škode odmislimo vse liste v T. Ko ta postopek nadaljujemo, na koncu pridemo do poddrevesa na dveh ali eni točki.

□

Kasnejše raziskave (Hendry [293], Holbert [300]) kažejo tudi, da sta omenjena 2-povezana grafa od centralnih oz. medianskih vozlišč lahko poljubno oddaljena. To dejstvo niti ni presenetljivo, saj sta funkciji za mediano ter za center grafa različni.



Slika 1.6: Na zgornji sliki vidimo podgraf iz šestih vozlišč, ki vsebuje obe s sivo označeni medianski vozlišči. Vozlišča v centru grafa so označena z v in w, od koder vidimo, da za zgornji graf velja tudi $C(G) \cap M(G) = \emptyset$.

Centroid grafa

Izračun *centroida grafa* predstavlja rezultat maxmin optimizacijskega problema. Kot smo videli v 1.9 lahko v grafu vsakemu vozlišču u dodelimo centroidno vrednost $c_F(u)$.

Definicija 1.17 Naj bo G poljuben povezan graf. *Centroid* grafa G je množica vozlišč, definirana kot:

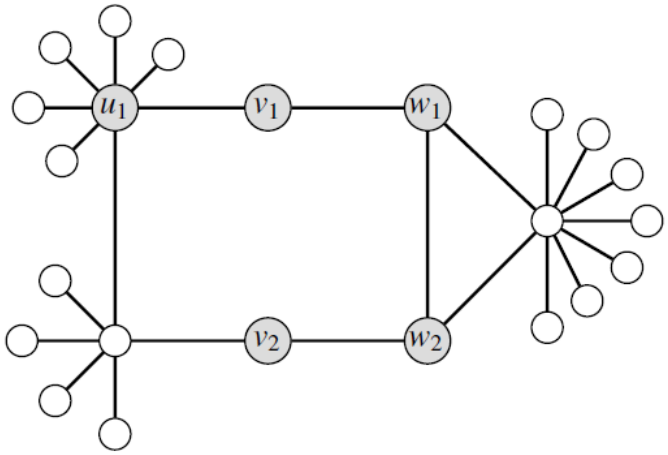
$$\begin{aligned} Z(G) &= \left\{ u \in V(G) : c_F(u) = \max_{v \in V(G)} \{c_F(v)\} \right\} \\ &= S_{c_F}(G) \end{aligned}$$

Vprašajmo se, kje lahko centroid grafa leži? Najprej se lotimo družine dreves. Zenlinka je v [594] dokazal znan rezultat, ki ga je Slater v [524] uporabil, ter dokazal naslednji izrek.

Izrek 1.18 *V poljubnem drevesu sta si mediana ter centroid grafa identična.*

Posplošitev zgornjega izreka sta dokazala Smart in Slater v [527]:

Izrek 1.19 *V poljubnem povezanem neusmerjenem grafu ležita mediana ter centroid na istem 2- po povezavah povezanem podgrafu.*



Slika 1.7: $C(G) = \{v_1, v_2\}, M(G) = \{u_1\}, Z(G) = \{w_1, w_2\}$
Slika prikazuje graf, ki ima paroma disjunktne množice centralnih, medianskih ter centroidnih vozlišč.

1.2.5 Najkrajše poti

Naslednji indeksi centralnosti temeljijo na najkrajših poteh. Najkrajše poti lahko obravnavamo tako med vozlišči kot med povezavami. Nekateri centralnostni indeksi so bili tako najprej predstavljeni za razdalje med vozlišči, ter šele kasneje posplošeni tudi na povezave. Ker lahko večkrat na enak način obravnavamo tako centralnost povezav kot centralnost vozlišč, bomo včasih uporabljali izraz *element*, ki lahko predstavlja tako povezave kot votlišča.

Centralnost obremenitve

Pri tej centralnosti preprosto štejemo število najkrajših poti, ki tečejo skozi izbran element. Ta intuicija je bila prvič predstavljena v [519], ter zajema omrežja, kjer se neka informacija ali transport vedno pretaka po najkrajših poteh. Tako lahko na nek način merimo "obremenitev" nekega elementa.

Definicija 1.20 (centralnost obremenitve) Naj bo G poljuben povezan graf, in naj $\sigma_{st}(v)$ predstavlja število najkrajših poti med s ter t, ki potekajo skozi element v. Potem lahko centralnost glede na obremenitev poljubnega vozlišča $v \in V(G)$ definiramo kot:

$$c_S(v) = \sum_{s \in V(G) \setminus v} \sum_{s \in V(G) \setminus s} \sigma_{st}(v),$$

za poljubno povezavo $e \in E(G)$ pa zelo podobno:

$$c_S(e) = \sum_{s \in V(G)} \sum_{s \in V(G)} \sigma_{st}(e).$$

Naj σ_{ab} predstavlja število najkrajših poti med a in b. Zgornji formuli sta si med seboj precej povezani z naslednjo lemo.

Lema 1.21 *V povezanem grafu G za vse $v \in V(G)$ velja:*

$$c_S(v) = \frac{1}{2} \sum_{e \in \Gamma(v)} c_S(e) - \sum_{s \in V(G) \setminus v} \sigma_{sv} - \sum_{t \in V(G) \setminus v} \sigma_{vt}$$

Dokaz. Vzemimo poljubno najkrajšo pot, ki povezuje par vozlišč $s \neq t \in V(G)$. Če bi preprosto sešteli vse centralnosti sosednih povezav od v, bi opazili, da smo vsako najkrajšo pot, ki poteka skozi v, šteli dvakrat. Tej intuitivni vsoti moramo seveda še odšteti število najkrajših poti, ki se začenjajo oz. končujejo v v, saj teh v zgornji intuiciji nismo šteli dvakrat. Ta sklep nam porodi zgornjo enačbo.

□

Vmesnostna centralnost glede na najkrajše poti

Na to centralnostno vrednost lahko gledamo kot na neke vrste normirano centralnost obremenitve. Naj bo $\delta_{st}(v)$ delež najkrajših poti med vozliščrms s in t, ki vsebujejo element$v$:

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

Ta delež si lahko smiselno predstavljamo kot verjetnost, da bo vozlišče oz. povezava v vpletena v komunikacijski kanal med vozliščema s ter t. Zdaj lahko definiramo vmesnostno centralnost najkrajših poti.

Definicija 1.22 Naj bo G povezan graf. Potem je vmesnostna centralnost najkrajših poti za poljubno vozlišče v definirano kot:

$$c_B(v) = \sum_{s \in V(G) \setminus v} \sum_{t \in V(G) \setminus v} \delta_{st}(v).$$

Vmesnostna centralnost najkrajših poti za povezave je definirana zelo podobno:

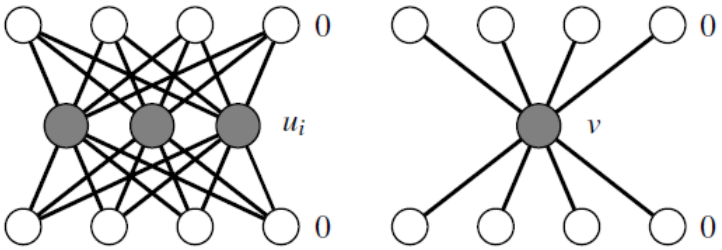
$$c_B(e) = \sum_{s \in V(G)} \sum_{t \in V(G)} \delta_{st}(e),$$

kjer je $e \in E(G)$.

Tudi tukaj so, podobno kot pri centralnosti obremenitve, trivialne najkrajše poti dolžine 0 izključene. Vmesnostna centralnost je bila predstavljena v [32,226] ter ima mnogo uporabnih interpretacij.

V [226] se je ta indeks izkazal za uporabnega, saj centralnost bližine ni primerna za nepovezane grafe. Razdalja med nepovezanima vozliščema je ponavadi neskončno, tu pa se pojavi problem, saj bi bila vrednost vseh vozlišč v takem grafu enaka $\frac{1}{\infty}$. Vmesnostna centralnost s tem problemom nima težav, saj bo vsak nepovezan par, ki torej nima nobene najkrajše poti, dodal 0 k centralnosti poljubne druge točke.

V [32] verjetnostno centralnost najkrajših poti avtorji poimenujejo *gneča*, ter nanjo gledajo kot centralnost pretoka, pravijo: "Gneča nekega elementa je delež pretoka skozenj, glede na celoten pretok v vsakem paru vozlišč v omrežju." Po tej definiciji bi lahko na $\delta_{st}(v)$ pogledali kot na količino pretoka skozi element v, če pošljemo eno enoto pretoka iz vozlišča s v vozlišče t, pri določenih pravilih tokovnih poti.



Slika 1.8: Centralnosti prvega grafa sta$c_S(u_i) = 16$ in $c_B(u_i) = 1/3$. Centralnosti drugega grafa sta $c_S(v) = 16$ in $c_B(v) = 1$. Kot vidimo centralnost obremenitve ni predvidena za merjenje komunikacijske pomembnosti. Boljša mera za to je vmesnostna centralnost najkrajših poti.

V naslednjem izreku bomo naredili povezavo med obema vmesnostnima centralnostima povezav ter vozlišč $c_B(e)$ in $c_B(v)$.

Izrek 1.23 *V usmerjenem grafu $G = (V, E)$ sta vmesnostni povezavi vozlišč in povezav povezani v formuli:*

$$c_B(v) = \sum_{e \in \Gamma^+(v)} c_B(e) - (n-1) = \sum_{e \in \Gamma^-(v)} c_B(e) - (n-1)$$

za vse $v \in V$, kjer je $n = |V|$.

Dokaz. Vzemimo poljubno najkrajšo pot, ki povezuje par vozlišč $s \neq t$. Ta pot prispeva k vmesnostni centralnosti vseh vozlišč ter povezav na njej natanko $\frac{1}{\sigma_{st}}$. Opazimo lahko, da poljubne poti, ki se ne začnejo oz. končajo v v, prispevajo vozlišču v enako, kot prispevajo tudi neki vhodni oz. izhodni sosednji povezavi. Kadar pa se najkrajša pot konča v v (takih poti je natanko $n-1$), pa k vozlišču ta pot prispeva $\frac{\sigma_{st}}{\sigma_{st}} = 1$. Ta intuicija nam porodi zgornjo enačbo.

□

1.2.6 Vitalnost

Mere vitalnosti nam predstavljajo še en pogled na pomembnost vozlišč oz. povezav v grafu. Glede na neko podano funkcijo, ki grafu G priredi neko realno število, nam vitalnost meri razliko funkcijskih vrednosti grafov G ter $G \setminus u$, za izbrano vozlišče oz. povezavo u. Vzemimo preprost primer velikega komunikacijskega omrežja, v katerem mora biti vsako vozlišče posredno, preko nekaj razvejitvenih točk, povezano z vsemi ostalimi vozlišči. Kvaliteto takega omrežja bi lahko opisali z Wienerjevim indeksom omrežja, tj. z vsoto preko vseh razdalj v grafu (glej definicijo 1.26). Potem predstavlja vitalnost poljubnega elementa x tega omrežja padec kvalitete omrežja, če bi bil element x odstranjen iz omrežja. Bolj formalno lahko to intuicijo definiramo takole:

Definicija 1.24 [Vitalnostni indeks] Nej bo $\mathbb{G}$ množica enostavnih, neusmerjenih ter neobteženih grafov G, ter naj bo $f : \mathbb{G} \to \mathbb{R}$ funkcija nad $\mathbb{G}$. Vitalnostni indeks $V(G, x)$ je definiran kot razlika vrednosti funkcije f na grafu G z ozirama brez elementa x:

$$V(G, x) = f(G) - f(G \setminus \{x\}).$$

Vmesnostna vitalnost pretoka

Tu si bomo ogledali centralnost po vozliščih, ki temelji na pretoku skozi omrežje. Predstavili bomo mero tipa maksimalnega pretoka, ki je podobna vmesnostni razdalji najkrajših poti v razdelku 1.2.5. To mero je predlagal Freeman s sodelavci [229]. Kot opažata Stephenson ter Zelen v [533], ni nujno, da neka informacija v komunikacijskem omrežju vedno potuje po najkrajši poti, kar v realnih analizah s pomočjo vmesnostne centralnosti najkrajših poti iz definicije 1.22 privede do zavajujočih rezultatov. Torej moramo upoštevati tudi drugačne poti.

Če izhajamo iz primera komunikacijskega omrežja, lahko vzamemo informacijo kot pretok, ter določimo vsaki povezavi neko nenegativno vrednost, ki predstavlja največjo količino informacije, ki je lahko posredovana preko nje. S tem razširimo model vmesnostne centralnosti v omrežja pretokov, kjer lahko vozčišče u vidimo kot vmesno vozlišče med različnimi ostalimi vozlišči. Naš cilj je izmeriti, v kolikšni meri je pretok med temi ostlimi vozlišči odvisen od vozlišče u.

Tej centralnosti bomo rekli vmesnostna vitalnost maksimalnega pretoka. Zaradi večje preprostosti, bomo smatrali, da je naš graf $G = (V, E)$ povezan neusmerjen graf z nenegativnimi pretočnimi vrednostmi na njegovih povezavah. Z f_{st} označimo vrednost največjega pretoka med vozliščema s in t, z $f_{st}(u)$ pa količino maksimalnega pretoka f_{st}, ki gre skozi u, glede na podane omejitve pretoka na povezavah. Naj bom $\tilde{f}_{st}(u)$ maksimalni pretok med u in v v grafu $G \setminus u$. Zanimalo nas bo, kako se vrednost f_{st} spremeni, če odstranimo iz omrežja vozlišče u. Glede na vmesnostno centralnost najkrajših poti (razdelek 1.2.5), brez problema zapišemo naslednjo definicijo.

Definicija 1.25 (vmesnostna cenralnost maksimalnega pretoka) Naj bo G povezan neusmerjen graf z nenegativnimi pretočnimi utežmi na povezavah. Potem je vmesnostna centralnost maksimalnega pretoka za vozlišče u definirana kot:

$$c_{mf}(u) = \sum_{\substack{s,t \in V(G) \\ u \neq s, u \neq t \\ f_{st} > 0}} \frac{f_{st}(u)}{f_{st}},$$

kjer velja $f_{st}(u) = f_{st} - \tilde{f}_{st}(u)$.

Vitalnost bližine

Analogno z centralnostnimi indeksi, predstavljenimi v razdelku 1.2.3, bomo tu predstavili novo centralnost, ki temelji na Wienerjevemu indeksu.

Definicija 1.26 Wienerjev indeksgrafa G označimo z $I_W(G)$ ter je definiran kot vsota preko vseh razdalj vseh parov vozlišč, torej:

$$I_W(G) = \sum_{v \in V(G)} \sum_{w \in V(G)} d(v, w).$$

Ni težko videti, da lahko Wienerjev indeks zapišemo tudi s pomočjo indeksa centralnosti bližine $c_C(v)$, kot

$$I_W(G) = \sum_{v \in V(G)} \frac{1}{c_C(v)}.$$

Zdaj bomo definirali novo centralnost, ki ji bomo rekli *vitalnost bližine.*

Definicija 1.27 (Vitalnost bližine) Naj bo x poljubno vozlišče ali povezava v grafu G. *Vitalnost bližine* je definirana kot:

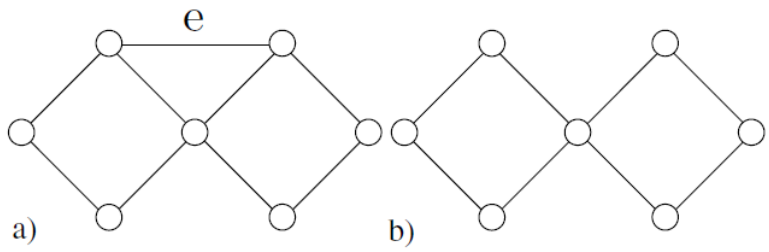
$$c_{CV}(x) = I_W(G) - I_W(G \setminus \{x\}).$$

Očitno je tudi ta centralnost vitalnost (glej definicijo 1.24), pri čemer smo za funkcijo nad grafom vzeli kar Wienerjev indeks. Kaj pa nam ta centralnostni indeks meri? Naj razdalja med dvema vozličlema predstavlja ceno komunikacije med tema vozliččema. Potem nam vitalnost bližine pove, za koliko se skupni stroški komunikacije med vsemi vozlišči v omrežju povečajo, če iz njega odstranimo element x. Z majhnim popravkom v formuli za Wienerjev indeks bi lahko izračunali tudi povprečno razdaljo med poljubnima vozličema v grafu:

$$d_{\hat{o}}(G) = \frac{I_W(G)}{n(n-1)},$$

kjer je $n = |V(G)|$.

Majhna težava pri tej ideji o vitalnosti bližine se pojavi, če odstranjeni element x razdeli graf na več komponent. V tem primeru je $c_{CV}(x) = -\infty$. V te robne primere se v tem zborniku ne bomo spuščali.



Slika 1.9: Slika kaže situacijo, ko odstranitev povezave e povzroči povečanja skupnega števila najkrajših poti v grafu.

Indeks bližnjic

Kljub temu, da indeks bližnjic ni vitalnostni indeks v smislu definicije 1.24, ga tu predstavimo, ker je intuitivno zelo povezan z vitalnostjo.

Definicija 1.28 Naj bo $e = (u, v)$ poljubna povezava v našem grafu G. *Indeks bližnjic* za povezavo e označimo z $c_{SV}(e)$, ter je definiran kot največje povečanje razdalje med poljubnima vozliščema v grafu, če je e odstranjen iz grafa, torej:

$$c_{SV}(e) = \max_{a,b \in V(G)} \left(d_{G \setminus e}(a, b) - d_G(a, b) \right)$$

Trditev 1.29 *Naj bo G poljuben graf. Maksimalna vrednost indeksa bližnjic iz definicje 1.28 za povezavo $e = (u, v)$ je dosežena pri izbiri vozlišč $(a, b) = (u, v)$.*

Dokaz. Naj bo $\alpha = d_{G \setminus e}(a, b) - d_G(a, b)$. Očitno velja $\alpha > 0$. Vzemimo poljubno najkrajšo pot dolžine l med poljubnim parom vozlišč a ter b. Denimo najprej, da ta najkrajša pot ne poteka skozi e. V tem primeru se dolžina poti pri odstranitvi e iz grafa ne spremeni, torej ta pot ne maksimizira izraza v zgornji definiciji.

Denimo zdaj, da najkrajša pot gre skozi e. Nova najkrajša pot se ne more podaljšati za več kot α, saj lahko brez težav najdemo pot med a in b, dolžine $d_G(a, b) - 1 + \alpha$, pri čemer si pomagamo z najkrajšo potjo med u in v v grafu $G \setminus e$.

□ Indeks bližnjic bi lahko direktno razširili na vozlišča, kjer bi namesto povezav odstranjevali vozlišča ter merili, koliko se poveča najbolj kritična najkrajša pot.

Vitalnost obremenitve

Na centralnost obremenitve iz razdelka 1.2.5 lahko gledamo tudi kot na vitalnostno mero. Pri centralnosti obremenitve smo preverjali delež najkrajših poti med vsemi pari, ki poteka skozi nek izbrani element - tj. na nek način delež najkrajših poti, ki bi jih izgubili, če bi izbrani element odstranili iz grafa.

Kljub temu, da to zveni zelo podobno ideji za vitalnost, ne smemo spregledati ključne razlike, ki je v definiciji vitalnosti: Količina "izgubljenih" najkrajših poti bi morala biti merjena relatovno z količino preostalih najkrajših poti v originalnem grafu. Ta podrobnost je pomembna, saj lahko že na preprostem primeru iz slike 1.9 vidimo, da lahko odstranitev povezave včasih celo poveča skupno število najkrajših poti.

Na levi strani slike 1.9(a) vidimo majhen graf z skupno 54 najkrajšimi potmi, izmed katerih 8 poteka skozi povezavo e. Po odstranitvi povezave e lahko v grafu najdemo 64 najkrajših poti. Seveda je 18 izmed njih zdaj daljših, kot so bile prej. Če denimo vzamemo najkrajšo pot med najbolj levim ter najbolj desnim vozliščem v (a), lahko vidimo, da se je pot po odstranitvi povezave e povečala za 1, pri čemer se je število najkrajših poti med vozliščema povečalo za tri.

Če želimo interpretirati centralnost obremenitve kot vitalnostno mero, takih podaljšanih najkrajših poti ne smemo upoštevati. Zato bomo temu primerno zdaj definirali vitalnost obremenitve.

Definicija 1.30 Naj bo $f(G)$ število najkrajših poti v grafu G ter naj bo v poljubno vozlišče. Potem je $f(G \setminus \{v\}$ definiran kot:

$$f(G \setminus \{v\}) = \sum_{s \in V(G)} \sum_{t \in V(G)} \sigma_{st} \left[d_G(s,t) = d_{G \setminus \{v\}}(s,t) \right],$$

pri čemer razumemo $\sigma_{st}\,[\mathtt{p}]$ kot

$$\sigma_{st}\,[\mathtt{p}] = \begin{cases} \sigma_{st} & ;\mathtt{p} \sim true \\ 0 & ;\mathtt{p} \sim false \end{cases}.$$

Podobno definiramo še za povezave. Če je e poljubna povezava v G, potem je $f(G \setminus \{e\}$ definiran kot:

$$f(G \setminus \{e\}) = \sum_{s \in V(G)} \sum_{t \in V(G)} \sigma_{st}(e) \left[d_G(s,t) = d_{G \setminus \{e\}}(s,t) \right],$$

Zgornjo definicijo lahko razumemo kot vsoto vseh tistih najkrajših poti, ki so pri odstranitvi izbrane povezave ohranile svojo dolžino. Po takih definicijah lahko centralnost obremenitve za element x izrazimo kar kot

$$c_S = f(G) - f(G \setminus \{x\}),$$

kar pa že izgleda zelo podobno kriteriju vitalnostnih mer iz definicije 1.24.

1.2.7 Električni tok

Centralnosti najkrajših poti temeljijo na ključni predpostavki, ki pravi, da ves pretok informacij, oz. prenos nekih dobrin poteka le po najkrajših poteh v omrežju. V tem razdelku si bomo pogledali tokovne centralnostne indekse, ki te predpostavke ne upoštevajo; predpostavili bomo, da transport po našem omrežju posnema prenos električnega toka po ustreznem električnem omrežju.

Električna omrežja

Tokovne centralnosti temeljijo na pretoku električnega toka v električnem omrežju; temu primerno na kratko opišimo notacijo ter potrebne funkcije, ki jih bomo v električnem omrežju uporabljali.

Definicija 1.31 *Električno omrežje* je definirano kot neusmerjen, povezan, enostaven graf $G = (V, E)$, ki mu pridružimo:

- funkcijo uporov $c : E \to \mathbb{R}$, ki za vsako povezavo določi njen upor;
- funkcijo $b : V \to \mathbb{R}$, ki predstavlja pritekanje oz. odtekanje elektrinega toka v to omrežje.

Pozitivne vrednosti funkcije b predstavljajo pritekajoči električni tok, med tem ko negativne vrednosti predstavljajo tok, ki iz omrežja odteka ($\sum_{v\in V} b(v)$ mora biti enaka 0). Ker je uporabno govoriti o smeri električnega toka, lahko za električno omrežje tudi vsako povezavo $e \in E$ poljubno usmerimo ter pridobomo usmerjeno povezavo $\overrightarrow{e}$, kar nam poda usmerjeno množico vozlišč $\overrightarrow{E}$.

Definicija 1.32 V električnem omrežju lahko definiramo tudi funkcijo $x : \overrightarrow{E} \to \mathbb{R}$, ki ji pravimo funkcija (električnega) toka v našem omrežju, če za vsak $v \in V$ velja

$$\sum_{w\in V|\{vw\}\in\overrightarrow{E}} x(vw) - \sum_{w\in V|\{wv\}\in\overrightarrow{E}} x(vw) = b(v)$$

ter

$$\sum_{e\in C} x(\overrightarrow{e}) = 0,$$

za vsak neusmerjen cikel $C \subseteq E$.

Zgornja pogoja funkcije x sta natanko oba Kirchoffova zakona o električnem toku ter potencialu. Negativne vrednosti funkcije x interpretiramo kot elektični tok, ki teče v nasprotni smeri od izbrane usmeritve ustrezne povezave.

Alternativno tokovni funkciji x lahko električni pretok v omrežju predstavimo z potenciali.

Definicija 1.33 Funkcija $p : V \to \mathbb{R}$ je (električni) potencial, če $p(v)-p(w) = x(vw)/c(vw)$ za vse $vw \in \overrightarrow{E}$.

Laplaceova matrika $L = L(G)$ v omrežju G naj bo

$$L_{vw} = \begin{cases} \sum c(e)_{e\ni v} & \text{if } v = w \\ -c(e) & \text{if } \{v, w\} \in E \\ 0 & otherwise \end{cases}$$

za vsaka $u, w \in V$.

Kot ima neko električno omrežje s podanima funkcijama c, b, enolično tokovno funkcijo x, ima podobno enolično podano tudi funkcijo p. Funkcijo p lahko izračunamo preko linearnega sistema $Lp = b$.

Za potrebe centralnosti, ki jih opišemo spodaj, uvedimo še poseben primer funkcije b, ko električni tok prihaja ter odhaja iz omrežja le v enem vozlišču, količina tega toka pa je ena enota. Z zapisom b_{st} označimo funkcijo b, za katero velja $b_{st}(s) = 1$, $b_{st}(t) = -1$ ter $b_{st}(x) = 0$ za vsak $x \neq \{s, t\}$.

Vmesnostna centralnost električnega toka

Newman [443] je prvi opisa centralnostni indeks, ki temelji na električnem toku. Kot v prejšnjem razdelku tudi tu operiramo z električnim omrežjem $G = V, E, c$. Vmesnostna centralnost električnega toka nekega vozlišča predstavlja delež enote pri funkciji $b(G) = b_{st}$, ki gre skozi izbrano vozlišče. Za izbran par s, t si igra pretok skozi izbrano vozlišče v podobno vlogo kot delež najkrajših poti σ_{st} skozi v, na katerem je temeljila vmesnostna centralnost najkrajših poti.

Definicija 1.34 Pretok električnega toka skozi vozlišče v pri $b = b_{st}$ za izbrana vozlišča v, s, t lahko je definiran kot:

$$\tau_{st}(v) = \frac{1}{2}\left(- |b_{st}(v)| + \sum_{e \ni v} |x\left(\overrightarrow{e}\right)| \right).$$

Tu izraz $- |b_{st}(v)|$ poskrbi, da je $\tau_{st}(s) = \tau_{st}(t) = 0$, ulomek $\frac{1}{2}$ pa poskrbi za dvojno štetje tokov, saj v zgornji enačbi seštevamo vse vhodne ter izhodne tokove. Z uporabo teh definicij lahko že definiramo nov indeks vmesnostne centralnosti

Definicija 1.35 Naj bo $G = V, E, c$ električno omrežje. Vmesnostna centralnost (električnega) toka $c_{CB} : V \to \mathbb{R}$ je definirana kot:

$$c_{CB}(v) = \frac{1}{(n-1)(n-2)} \sum_{s,t \in V} \tau_{st}(v)$$

za vse $v \in V$.

Tudi tu ulomek $\frac{1}{(n-1)(n-2)}$ služi kot faktor, s katerim normiramo tokove, ki smo jih prevečkrat šteli. Ker ima električno omrežje enolično definiran tok za podano funkcijo b, je tudi centralnost c_{CB} dobro definirana.

Centralnost bližine za tokovna omrežja

Podobno kot za vmesnostne centralnosti lahko prevedemo na električna omrežja tudi centralnost bližine. Pri najkrajših poteh nam bližina kaže najkrajšo dolžino od nekega vozlišča do vseh preostalih vozlišč. Pri električnih omrežjih bomo to intuicijo zajeli tako, da bomo za mero centralnosti bližine med vozliščema v in w vzeli kar razliko njihovih potencialov $p(v) - p(w)$. Tako pridemo do definicije tokovne centralnosti bližine:

Definicija 1.36 (Tokovna centralnost bližine) Naj bo $G = V, E, c$ naše električno omrežje. Potem je tokovna centralnost bližine $c_{CC}(v) : V(G) \to \mathbb{R}$ definirana kot

$$c_{CC}(v) = \frac{n-1}{\sum_{t \in V \setminus v} (p_{vt}(v) - p_{vt}(t))},$$

za vse $v \in V$, kjer p_{vt} razumemo kot funkcijo potencialov, ki je izračunana glede na podano funkcijo $b = b_{vt}$. Faktor $n - 1$ tu spet normira prevečkrat štete tokove.

Presenetljivo je, da je tokovna centralnost bližine enaka informacijski centralnosti. Stephenson ter Zelen [533] sta predstavila informacijsko centralnost, ki je zanimiva, ker vključuje pretoke preko vseh poti v omrežju, ter ne le preko najkrajših poti. Poleg tega

informacijska centralnost upošteva, ta prenašajo nekatere poti več informacij kot druge. Kot zanimivost zapišimo formulo za informacijsko centralnost:

$$c_I(v)^{-1} = nM_{vv} + \text{trace}(M) - \frac{2}{n},$$

kjer je M definirana kot $M = (L + U)^{-1}$- L je Laplaceova matrika, U pa je matrika enake velikosti, kjer so vse vrednosti enake 1.

1.2.8 Naključni procesi

Včasih se nam lahko zgodi, da prej omenjenih centralnosti ne moremo izračunati zaradi pomanjkanja informacij o grafu, ali pa zaradi velikosti grafa ter premajhne računske moči. V teh primerih centralnosti najkrajših poti nimajo nobenega smisla, zato bomo v tem razdelku predstavili alternativne metode naključnega sprehajanja. V naključnih sprehajanjih se po grafovskih povezavah sprehajamo od vozlišča do vozlišča, vedno z neko stopnjo naključnosti.

Naključni sprehodi in centralnost stopnje

V primeru neusmerjenih grafov lahko hitro pridemo do ugotovitve, ki poveže centralnost naključnega sprehoda z najpreprostejšim konceptom v omrežju - s stopnjo vozlišča. V sledečem izreku dokažemo, da so stacionarne verjetnosti vozlišč v standardnem naključnem sprehodu sorazmerne s stopnjo vozlišč.

Izrek 1.37 $p_{ij} = \frac{a_{ij}}{d(i)} \Rightarrow \pi_i = \frac{d(i)}{\sum_{v \in V} d(v)}$

Dokaz.

$$(\pi P)_j = \sum_{i \in V} \pi_i p_{ij} = \frac{\sum_{i \in V} d(i) p_{ij}}{\sum_{v \in V} d(v)} = \frac{\sum_{i \in V} a_{ij}}{\sum_{v \in V} d(v)} = \frac{d(j)}{\sum_{v \in V} d(v)} = \pi_j$$

□

Vmesnostna centralnost naključnih sprehodov

Vmesnostna centralnost naključnih sprehodov je bila prvič predstavljena v [443] ter temelji na naslednji ideji. Predpostavimo, da smo v neuteženem, neusmerjenem, povezanem grafu, ter da ima votlišče s sporočilo za vozlišče t, vendar niti s, niti nobeno drugo vozlišče v grafu ne pozna načina, kako bi sporočilo prenesli po najkrajši poti. V času 0 bo vozlišče poslalo sporočilo enemu od sosedov. Vsako vozlišče, ki dobi sporočilo, bo le poslalo sporočilo naprej naključnemu sosedu. Če je sporočilo po nekaj časa doseglo vozlišče t, se ne bo več pošiljalo naprej, ter se bo proces zaključil. Bolj formalno naj bo m_{ij} verjetnost, da vozlišče j pošlje sporočilo vozlišču i v nekem času, če je vozlišče v v prejšnji časovni periodi imelo sporočilo:

$$m_{ij} = \begin{cases} \frac{a_{ij}}{d(j)} & \text{če } j \neq t \\ 0 & \text{sicer} \end{cases},$$

kjer a_{ij} predstavlja ij-ti element sosednostne matrike A, $d(j)$ pa je stopnja vozlišča j. Dobljeno matriko označimo z M. Naj bo D matrika stopenj našega grafa.

$$d_{ij} = \begin{cases} d(i) & \text{če } i = j \\ 0 & sicer \end{cases}$$

Brez težav opazimo, da je inverz matrike D diagonalna matrika z invertiranimi stopnjami vozlišč po diagonali. Zaradi posebnega obnašanja matrika M pri vozlišču t ne moremo poenostavljeno napisati $M = A \cdot D^{-1}$. Temu primerno naj bo X_t kar poljubna matrika X, ki ji izbrišemo t-ti stolpec ter vrstico. Zdaj lahko brez slabe vesti zapišemo relacijo med tremi matrikami:

$$M_t = A_t \cdot D_t^{-1}.$$

Vmesnostna centralnost naključnih sprehodov upošteva vse možne poti, ki jih naključni sprehod lahko uporabi, vključno z verjetnostmi, s katerimi se za določene poti odločamo. Tu se pojavi vprašanje, kako izračunati množico vseh poti, ter kako izračunati verjetnost, da smo neko izbrano pot uporabili. Za lažje razumevanje problema bomo najprej pokazali, koliko različnih $i - j$ sprehodov dolžine r obstaja v grafu (i in j sta tu poljubno izbrani vozlišči). Ni težko videti, da je odgovor $(A^r)_{ij}$ kjer je A^r kar sosednostna matrika A, ki jo r-krat potenciramo. Ker pa nas število različnih sprehodov ne zanima, si bomo raje ogledali verjetnost nekega sprehoda dolžine r, ki se začne v vozlišču s ter konča v vozilšču j. To verjetnost pa dobimo z r-to potenco matrike M_t v vrstici j ter stolpcu s, kar označimo z $(M_t^r)_{js}$. Tako lahko zapišemo verjetnost, da je bilo sporočilo poslano vozlišču i v koraku $r + 1$:

$$\left(M_t^{r+1}\right)_{js} = m_{ij}^{-1} \left(M_t^r\right)_{js}.$$

Nas zanimajo verjetnosti, da pošilja vozlišče j sporočilo, ki potuje po poti od s do i, ne glede na dolžino teh sprehodov. Torej moramo sešteti vse verjetnosti za poljubne dolžine sprehodov od 0 do ∞:

$$\sum_{r=0}^{\infty} m_{ij}^{-1} \left(M_t^r\right)_{js} = m_{ij}^{-1} \left[\left(I_{n-1} - M_t\right)^{-1}\right]_{js},$$

kjer je I_{n-1} identična matrika dimenzije $n - 1$.

Naj bo $\mathbf{s}$ vektor dimenzije $n-1$, ki ima vrednost 1 pri vozlišču s ter vrednost 0 povsod drugod. Če zgornjo enačbo zapišemo v matrični notaciji, dobimo:

$$\begin{aligned} v^{st} &= D_t^{-1} \cdot \left(I - M_t\right)^{-1} \cdot \mathbf{s} \\ &= \left(D_t - A_t\right)^{-1} \cdot \mathbf{s} \end{aligned} \tag{1.1}$$

Vektor v^{st} tu predstavlja verjetnost, da najdemo sporočilo pri vozlišču i, ko je na poti od vozliša s do vozlišča t. Seveda lahko naključni sprehodi vključujejo trivialne dele, npr, obhod iz vozlišča a v b, ter potem nazaj v a, itd. Intuitivno se zdi, da nekemu vozlišču ni smiselno dati visoke centralnosti, če je večina obhodov skozenj trivialnega, zgoraj opisanega tipa, ter podobnih trivialnih ponovitev skozi nek poljuben cikel. Pomembno je, da se zavedamo, da v^{st} upošteva le verjetnosti, ki teh ciklov ne upoštevajo.

Povezava med naključnimi sprehodi ter električnimi omrežji

Na tej točki nam lahko postane jasno, da so naključni sprehodi tesno povezani s tokovnimi grafi električnih omrežij, ki smo jih opisali v prejšnjem razdelku 1.2.7. Vzemimo poljubno električno omrežje $G = (V, E, c)$ z enotskimi utežmi na povezavah, torej $e \in E \to c(e) = 1$. Temu primerno dobimo Laplaceovo matriko $L(G) = D - A$, kjer sta D ter A zaporedoma matrika stopenj ter sosednostna matrika, kot ju imamo tudi v zgornjih enačbah. Torej je potencial p_{st} v G, če vzamemo za vhodni tok funkcijo $b = b_{st}$, rešitev problema $Lp_{st} = b_{st}$. Rank matrike L sicer ni poln, a to težavo lahko odpravimo, če popravimo potencial enega vozlišča, denimo vozlišča v. To lahko storimo, saj so potenciali enolični, glede na aditivni faktor. Izbrišimo zdaj vrstico ter stolpec, ki ustreza izbranemu vozlišču v v matrikah L, D, V ter dobljene matrike zaporedoma označimo z L_v, D_v ter A_v, kjer ima L_v zdaj polni rank, ter je zato obrnljiva. Glede na zgoraj napisano notacijo, lahko zapišemo potencial p_{st}:

$$p_{st} = L_v^{-1} b_{st} = (D_v - A_v)^{-1} b_{st},$$

kar je ekvivalentno enačbi 1.1 zgoraj, kar kaže na relacijo med električnimi tokovi, potenciali v električnih omrežjih ter naključnimi sprehodi. Globljo obravnavo omenjene povezave najdemo v [67].

Opazili smo torej, da je vmesnostna centralnost naključnih sprehodov $c_{RWB} : V \to \mathbb{R}$ v resnici ekvivalentna vmesnostni tokovni centralnosti, torej $v \in V(G) \to c_{RWB}(v) = c_{CB}(v)$.

Naključni sprehodi in centralnost bližine

Podoben pristop nam poda še eno vrsto centralnosti bližine naključnih sprehodov, ki temelji na *povprečnem času prvega srečanja (PČPS)* ter je bila prvič predstavljena kot Markovska centralnost v [580].

Definicija 1.38 Povprečni čas prvega srečanja m_{st} je pričakovano število vozlišč, ki jih naše sporočilo sreča, dokler prvič ne sreča končno vozlišče. Če sporočilo začne svoj naključni sprehod v vozlišču s ter konča v t, lahko zapišemo:

$$m_{st} = \sum_{n=1}^{\infty} n \cdot f_{st}^{(n)},$$

kjer $f_{st}^{(n)}$ predstavlja verjetnost, da smo od vozlišča s prvič prispeli v vozlišče t po natančno n korakih.

Označimo z M matriko PČPS, ki ima v stolpcih ter vrsticah s, t elemente m_{st}. M lahko izračunamo preko naslednje relacije:

$$M = (I - EZ_{dg})\, D,$$

kjer velja

- I je identična matrika,
- E je matrika samih enic,
- Z_{dg} je matrika, ki je enaka fundamentalni matriki Z na diagonali, povsod drugod pa je nič,

- Z je fundamentalna matrika, podana z relacijo

$$Z = \left(I - A - 1_n \pi^T\right)^{-1}$$

- 1_n je stolpec samih enic, dolžine n
- π je vektor dimenzuje n, ki za vsako vozlišče označuje stancionarno distribucijo naključnega sprehoda podanega grafa v tem vozlišču, tj. pričakovan delež časa, ko se bo sporočilo nahajalo na naključnem sprehodi nahajalo v izbranem vozlišču. (Ta model predpostavlja, da se transport sporočila od ene do druge točke zgodi v trenutku.)

Definicija 1.39 Markovska centralnost $c_M(v)$ je definirana kot inverz povprečne PČPS matrike za vse naključne sprehode, ki se začnejo v poljubni točki s ter imajo cilj v poljubni točki v:

$$c_M(v) = \frac{n}{\sum_{s \in V} m_{sv}}.$$

To centralnost lahko definiramo tako za usmerjene, kot za neusmerjene grafe.

Poglavje 2

Algoritmi za indekse centralnosti

Domen Blenkuš, Matej Filip, Tilen Marc, Aljaž Zalar

2.1 Indeks centralnosti

Z indeksom centralnosti izmerimo intuitiven občutek, da so nekatere točke oz. povezave v grafu bolj pomembne (centralne) od drugih. Vendar pa takšne centralnosti ne moremo enolično definirati, saj si pod tem pojmom, glede na okoliščine, predstavljamo različne stvari (npr. pomembnost, vplivnost, nadzor,...).

2.1.1 Primer uporabe

Za primer vzemimo volitve predstavnika študentov v letniku. Najprej moramo narediti matematični model dane situacije. Študente in odnose med njimi predstavimo z usmerjenim grafom, kjer vsaka točka predstavlja enega študenta, s povezavami med njimi pa lahko predstavimo več odnosov. Tako lahko usmerjena povezava od točke A do točke B pomeni:

1. da je oseba A glasovala za osebo B,
2. da je oseba A prepričala osebo B, da je glasovala za njenega favorita,
3. da je oseba A prepričala osebo B, da je glasovala za drugega kandidata, kot bi prvotno.

V prvem primeru je najbolj centralna oseba, ki je prejela največ glasov, v drugem primeru oseba z največjim vplivom, v tretjem primeru pa oseba, ki je najboljši prepričevalec. V naslednjih razdelkih bomo vsaki točki v grafu priredimo realno število, ki predstavlja njeno centralnost. Bolj kot je točka centralna, večje število ima.

2.1.2 Ohlapna definicija

Preden se posvetimo indeksom centralnosti, poskušajmo matematično formulirati indeks centralnosti. Zaradi raznolike uporabe indeksa centralnosti skozi zgodovino poenotena in

splošno sprejeta definicija ne obstaja. Zaradi tega si bomo pogledali samo najosnovnejše pogoje, ki jih mora indeks centralnosti izpolnjevati. Kot smo že omenili, bi radi, da bolj pomembnim točkam priredimo večjo vrednost, vendar pa je taka prireditev vse prej kot enolična. Vseeno pa bi radi, da je indeks centralnosti določene točke odvisen le od strukture grafa. To zajamemo v naslednji definiciji.

Spomnimo se, da sta grafa $G = (V, E)$ in $H = (\widetilde{V}, \widetilde{E})$ izomorfna ($G \simeq H$) natanko tedaj, ko obstaja taka bijektivna preslikava $\phi : V \to \widetilde{V}$, da je (u, v) povezava v E natanko tedaj, ko je $(\phi(u), \phi(v))$ povezava v $\tilde{E}$.

Definicija 2.1 Naj bo $G = (V, E)$ uteženi, usmerjen ali neusmerjen multigraf. Funkcija $c : V \to \mathbb{R}$ se imenuje indeks centralnosti natanko takrat, ko iz $G \simeq H$ z izomorfizmom ϕ sledi $c_G(v) = c_H(\phi(v))$, $\forall v \in V$, kjer $c_G(v)$ označuje vrednost funkcije c v grafu G.

V definiciji smo definirali indeks centralnosti za vozlišča. Podobno lahko definiramo tudi indeks centralnosti za povezave.

Poudariti moramo, da nam po zgornji definiciji razlika med indeksoma centralnosti dveh točk v splošnem ne pove nič o tem, koliko je ena točka bolj centralna od druge. Ali nam omenjena razlika kaj pove, je odvisno od konkretnega problema.

2.1.3 Sosedska centralnost

Sosedska centralnost je najenostavnejši primer centralnosti. Prvi zgled so volitve, pri čemer nas zanima, kateri kandidat ima največ volilcev. V jeziku grafov nas torej zanima, katero vozlišče ima največjo vhodno stopnjo. Drugi zgled pa je problem lokacije objekta, npr. skladišča, glede na to, da bomo od njega dosegli čim več drugih lokacij. V jeziku grafov nas zanima, katertočka ima največjo izhodno stopnjo. Tako pridemo do naslednje definicije.

Definicija 2.2 Naj bo $G = (V, E)$ digraf. Vhodno sosedsko centralnost točke $v \in V$ označimo z $c_{iD}(v)$ in je enaka številu vhodnih povezav točke v ($c_{iD}(v) = d^-(v)$). Podobno izhodno sosedsko centralnost označimo z $c_{oD}(v)$ in je enaka številu izhodnih povezav ($c_{oD}(v) = d^+(v)$).

Če imamo neusmerjen graf, imamo samo eno sosedsko centralnost in je ta enaka številu vseh povezav točke.

Sosedska centralnost je primer lokalne centralnosti, saj je indeks točke odvisen le od njenih sosed.

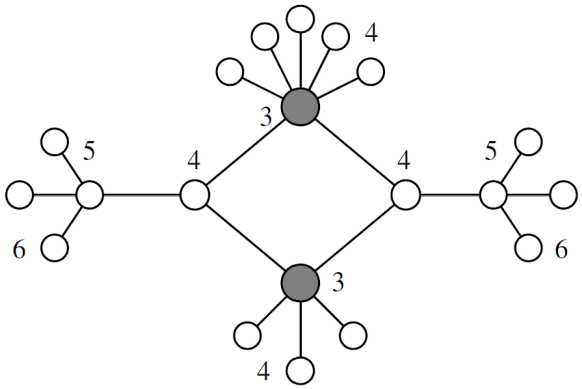
2.1.4 Ekscentrična centralnost

V naslednjem primeru se ukvarjamo z uteženim grafom, kjer so vozlišča lokacije nekega mesta, povezave pa predstavljajo obstoj prometne povezave med dvema lokacijama. Teže povezav predstavljajo čas vožnje. Vprašanje je, kam postaviti bolnišnico, gasilski dom ali drugo urgentno ustanovo, da bodo imeli reševalci čim krajši najdaljši odzivni čas. Da odgovorimo na to vprašanje, moramo za vsako točko pogledati najkrajše poti do vseh ostalih točk in potem točki priredimo največjo izmed teh vrednosti. Vidimo, da je iskana točka tista, ki ima najmanjšo vrednostjo. Za postavljeno vprašanje je taka točka najbolj centralna. Če hočemo, da ima prirejen indeks v tej točki največjo vrednost, samo še invertiramo izračunane vrednosti. To vrsto centralnosti lahko preprosto zapišemo v matematičnem jeziku v naslednji definiciji:

Definicija 2.3 Naj bo $G = (V, E)$ graf. Ekscentrična centralnost točke $v \in V$ je

$$c_E(v) = \frac{1}{max\{d(v,u); u \in V\}}$$

Problem 2.4 *Na sliki 2.1 vidimo primer uporabe ekscentrične centralnosti. Številke poleg točk označujejo maksimalno najkrajšo pot, najcentralnejši točki pa sta pobarvani sivo.*



Slika 2.1: Uporaba ekscentrične centralnosti

2.1.5 Bližinska centralnost

Tudi v naslednjem zgledu se bomo ukvarjali z uteženim grafom, kjer so vozlišča lokacije nekega mesta, povezave pa predstavljajo obstoj prometne povezave med dvema lokacijama. Teže povezav tokrat ne predstavljajo časa vožnje, ampak potne stroške. Radi bi postavili nakupovalno središče na čim optimalnejše mesto. Naš cilj je minimizirati potne stroške **vseh** kupcev, zato želimo minimizirali vsoto vseh cen, ki jih morajo prevoziti do nakupovalnega središča. Torej za vsako vozlišče izračunamo vsoto najkrajših poti do vseh ostalih vozlišč in za najugodnejše vozlišče izberemo tisto z najmanjšo vsoto. Spet lahko invertiramo vrednosti, da dobimo premo odvisnost pomembnosti vozlišč z njihovimi vrednostmi. Ta vrsta centralnosti se v matematičnem jeziku glasi:

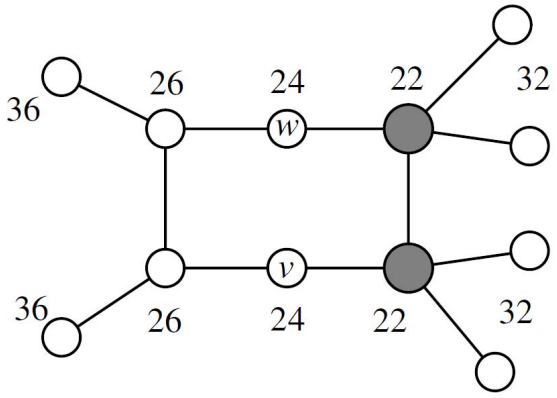
Definicija 2.5 Naj bo $G = (V, E)$ graf. Bližinsko centralnost točke $v \in V$ definiramo kot

$$c_C(u) = \frac{1}{\sum_{v \in V} d(u,v)}.$$

Problem 2.6 *Na sliki 2.2 vidimo primer uporabe bližinske centralnosti. Številke poleg točk označujejo vsoto najkrajših poti do vseh ostalih točk, najcentralnejši točki pa sta pobarvani sivo.*

2.1.6 Centralnost vmesnika na najkrajši poti

V tem razdelku bi si radi pogledali, kakšen vpliv ima določeno vozlišče pri komunikaciji med drugimi vozlišči. Izberimo različna vozlišča $s, t, v \in V$. Pomembnost vozlišča v za komunikacijo med vozliščema s in t bomo merili z deležem najkrajših poti med s in t, ki gredo skozi v.



Slika 2.2: Uporaba bližinske centralnosti

Vpeljimo nekaj oznak. Naj σ_{st} označuje število najkrajših poti med točkama s in t, $\sigma_{st}(v)$ pa število tistih, ki gredo skozi c. Z $\delta_{st}(v)$ označimo še delež najkrajših poti med točkama s in t, ki vsebujejo točko v. Torej velja

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

Ob predpostavki, da za pot med dvema vozliščema vedno uporabimo najkrajšo pot, si lahko $\delta_{st}(v)$ razlagamo tudi kot verjetnost, da se vozlišče v pojavi na najkrajši poti med vozliščema s in t. Če za fiksno vozlišče v seštejemo te verjetnosti za vse pare vozlišč (s, t), potem dobimo merilo, kako pomemben posrednik za komuniciranje v celotnem omrežju je vozlišče v. Spet formalizirajmo v definiciji:

Definicija 2.7 Indeks centralnosti vmesnika na najkrajši poti definiramo kot

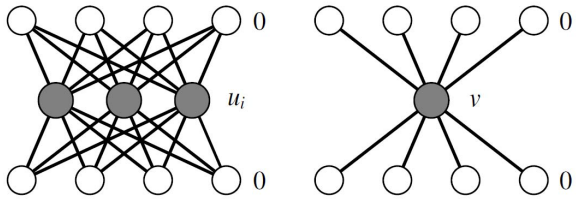
$$c_B(v) = \sum_{s \neq v} \sum_{t \neq v} \delta_{st}(v).$$

Vmesniško centralnost so uvedli, ker bližinske centralnosti ne moremo uporabljati na nepovezanih grafih, saj razdaljo med točkama v različnih komponentah ponavadi definiramo kot ∞ iz tega pa sledi, da imajo vse točke indeks centralnosti enak $\frac{1}{\infty}$, s tem pa centralnost izgubi svoj pomen. Pri vmesniški centralnosti se s tem problemom ne srečamo, saj vsaki nepovezani točki samo dodata 0 k centralnosti vseh ostalih točk, če definiramo, da je $\delta_{st} = 0$ v primeru, ko je $\sigma_{st} = 0$.

Problem 2.8 *Na sliki 2.3 lahko vidimo, kako se centralnost vmesnika na najkrajši poti lahko uporabi na trodelnem grafu. Na levi sliki lahko iz prvega v tretji nivo pridemo čez katerokoli izmed točk u_i, na desni sliki pa lahko pridemo samo čez točko v. Torej je točka v veliko bolj pomembna za komunikacijo med prvim in tretjim nivojem kot posamezna izmed točk u_i.*

2.1.7 Spletna centralnost

Veliko ljudi uporablja svetovni splet za iskanje informacij. Zaradi velike obsežnosti spleta, sestavljenega iz spletnih strani, ki so povezane s hiperpovezavami, so potrebni zahtevni iskalni algoritmi.



Slika 2.3: Uporaba centralnosti posrednika na najkrajši poti

Ključno vprašanje je, kako naj se spletni iskalnik odloči, katere strani so najbolj primerne za določeno poizvedbo. Za to moramo razvrstiti spletne strani glede na primernost in pomembnost. Delno je to narejeno s tekstovnim iskanjem po straneh, vendar pa napredni spletni iskalniki uporabijo tudi stukturo spleta, da določijo pomembnost posamezne strani. Na tem mestu pa pridejo v uporabo indeksi centralnosti, ki si jih bomo v nadaljevanju podrobneje pogledali.

Za začetek si poglejmo model slučajnega spletnega uporabnika, saj je njegovo poznavanje ključno za smiselno izbiro centralnih indeksov, ki jih bomo uporabili.

Model slučajnega spletnega uporabnika

V tem podrazdelku bomo v matematičnem jeziku opisali obnašanje slučajnega spletnega uporabnika. Najenostavnejši model simulira njegovo obnašanje kot slučajni sprehod po grafu. Svetovni splet predstavimo kot graf $G = (V, E)$, kjer je V množica vseh spletnih strani i, množica E pa množica usmerjenih povezav (i, j), pri čemer usmerjena povezava med stranjo i in j obstaja natanko tedaj, ko na strani i obstaja hiperpovezava do strani j.

Slučajni spletni uporabnik začne sprehod na slučajno izbrani spletni strani in nadaljuje po njenih sosedah. To predstavimo z matriko P, kjer element p_{ij} predstavlja verjetnost, da uporabnik sprehod s strani i nadaljuje na stran j.

Naš cilj v nadaljevanju bo razvrstiti spletne strani glede na verjetnost, da se bo slučajni uporabnik po velikem številu korakov nahajal na določeni spletni strani. Da bo ta verjetnost dobro definirana, moramo zahtevati dvoje. Kot prvo mora limitna verjetnost obstajati in kot drugo potrebujemo neodvisnost od začetne strani.

V teoriji markovskih verig se obstoju take verjetnosti reče stacionarna porazdelitev. Ni težko videti, da je v primeru obstoja stacionarne porazdelitve to kar levi lastni vektor (vektor x, za katerega velja $xA = \lambda x$, za neko lastno vrednost *lambda* matrike A) lastne vrednosti 1 prehodne matrike P. Izkaže pa se, da stacionarna porazdelitev markovske verige obstaja, če je prehodna matrika stohastična, tj. vsota vsake vrstice mora biti 1, in nerazcepna, tj. za vsak par vozlišč obstaja neka pot med njima. V jeziku grafov se obe lastnosti odražata v krepki povezanosti grafa.

Ker pa svetovni splet očitno ni krepko povezan, matrika prehodnih verjetnosti P ni nujno stohastična niti nerazcepna. Da bodo imele zgornje ideje smisel, moramo spremeniti matriko P tako, da bo stohastična in nerazcepna, pri čemer ne smemo preveč pokvariti modela slučajnega uporabnika. To naredimo v dveh korakih:

1. Da naredimo matriko P stohastično, v vsaki ničelni vrstici vse vrednosti nastavimo na $\frac{1}{n}$ (uporabnik slučajno skoči na neko spletno stran). Tako dobimo stohastično matriko P', ki pa je še vedno lahko razcepna.

2. Da naredimo matriko P' nerazcepno, najprej definiramo matriko E, ki je enake velikosti kot P' in ima vse elemente enake $\frac{1}{n}$. To matriko lahko imenujemo *matrika naključnega skoka*. Da naredimo matriko P' nerazcepno, izberemo neko konveksno kombinacijo matrik P' in E. Tako definiramo matriko $P'' = \alpha P' + (1-\alpha)E$, kjer je $\alpha \in [0,1]$. Seveda je v praksi pomembno, kakšen je α, saj s tem spreminjamo naravo spletnega uporabnika, vendar nas na tem mestu to ne zanima. Omenimo še, da model točno določenega uporabnika dobimo z ustreznim spreminjanjem matrike E.

Indeks PageRank

PageRank je eden najpomembnejših indeksov, ki ga spletni iskalniki, med njimi tudi Google, uporabljajo v algoritmih v svojih iskalnikih.

Ideja indeksa je, da stran oceni glede na njene topološke značilnosti. Topološke lastnosti strani so odvisne samo od njenega položaj v spletu, a so neodvisne od njene vsebine.

PageRank indeks strani je lokalni indeks, saj je odvisen samo od PageRank indeksov njenih sosedov. Torej pomembnost strani narašča s številom sosedov in njihovimi pomembnosti. Določanje indeksov $c_{PR}(p)$, kjer je $p \in V$, tako postane rekurzivni problem podan z enačbami:

$$c_{PR}(p) = d \sum_{q \in \Gamma_p^-} \frac{c_{PR}(q)}{d^+(q)} + (1-d),$$

kjer je $c_{PR}(q)$ PageRank strani q, d pa je obtežitveni faktor. Zapisano v matrični obliki se to glasi:

$$c_{PR} = dPc_{PR} + (1-d)1_n,$$

kjer je matrika prehodnih verjetnosti P definirana z

$$p_{ij} = \begin{cases} \frac{1}{d^+(i)}, & \text{if } (i,j) \in E \\ 0, & \text{sicer} \end{cases}.$$

Ta sistem ponavadi rešimo s preprosto Jacobijevo iteracijo:

$$c_{PR}^k = dPc_{PR}^{k-1} + (1-d)1_n$$

Naslednji izrek nam zagotavlja konvergenco in enolično rešljivost iteracije.

Izrek 2.9 *Če je $d \in [0,1)$ ima zgornja enačba enolično rešitev $c_{PR}^* = (1-d)(I_n - dP)^{-1}1_n$ in za rešitev iteracije velja $\lim_{k\to\infty} c_{PR}^k = c_{PR}^*$ za poljuben začetni vektor c_{PR}^0.*

Dokaz. Navedimo samo idejo dokaza. Potrebno je dokazati, da ima matrika P spektralni radij manjši od 1 oz. da so vse njene lastne vrednosti absolutno manjše od 1. Iz numerične matematike vemo, da v tem primeru zaporedje konvergira k rešitvi (po Izreku 4.1 v [1]).

□

2.2 Osnovni algoritmi

Uporabnost različnih indeksov centralnosti zavisi od zmožnosti hitrega izračuna le teh. To je pomemben problem v računalništvu in veliko napora je vloženega za razvoj in analizo učinkovitih algoritmov. V sledečem razdelku bomo obravnavali algoritme za izračun indeksov obravnavanih v prejšnjem poglavju.

Večina naštetih centralnosti temelji na izračunu razdalj med vozlišči. Naivni pristop je torej, da najprej izračunamo vse najkrajše poti in nato uporabimo definicijo centralnosti. Ta pristop je dober, saj najhitrejši algoritmi zmorejo izračun najkrajših poti nekje med n^2 in n^3 korakov in za izračun centralnosti praviloma porabimo še okoli n^2 korakov, torej končamo v polinomskem času. Kljub temu nam to v primeru velikih omrežji ni dovolj, zato potrebujemo specializirane algoritme, ki na konkretnem omrežju izkoriščajo njegove lastnosti. Obstajajo pa tudi omrežja (npr. graf spleta), kjer tudi specializirani algoritmi niso dovolj, saj njihova velikost prevelika za eksaktno računanje. Takrat je potrebna aproksimacija, ki jo lahko izračunamo že v linearnem času.

Pomemben aspekt omrežij, ki se pojavljajo v resničnem življenju, je, da se s časom pogosto spreminjajo. Najpomembnejši primer takega grafa je graf spleta. Zato bi radi, da nam ni treba ob vsaki spremembi grafa indeksov centralnosti računati na novo, raje bi jih dobili iz prejšnjih rezultatov. Taki algoritmi niso pomembni samo v spremenljivih okoljih, izboljšajo tudi zmogljivost izračuna indeksov centralnosti, pri katerih sama definicija zahteva odstranjevanje elementov iz grafa. Primer takih so algoritmi za dinamične najkrajše poti vseh parov vozlišč.

V nadaljevanju bomo obravnavali različne algoritme, nekatere bomo podrobneje opisali in predstavili njihove glavne ideje, druge bomo samo omenili in navedli vir, kjer lahko najdemo podrobnejše informacije. Vpogled v osnovne algoritme nam bo omogočil znanje, ki ga bomo uporabili za specifične algoritme indeksov centralnosti in kasneje za opise delovanj algoritmov, s katerimi analiziramo večja omrežja.

2.2.1 Najkrajše poti

Eden najbolj klasičnih algoritmov je algoritem za iskanje najkrajših poti med izbranim vozliščem imenovanim vir in ostalimi vozlišči. Znan je pod imenom Single Source Shortest Path (SSSP) problem.

Prvi algoritem, ki problem reši v polinomskem času, je predstavil Dijkstra [4] . Namenjen je za grafe, ki nimajo negativnih uteži na povezavah. Označimo s s vir, torej vozlišče iz katerega računamo dolžine poti. Algoritem deluje s pomočjo treh seznamov, ki jih po korakih spreminja. Ima seznam vrednosti $d(s, v)$ (v vozlišča), tj. seznam najkrajših poti med vozliščema s in v, ki so vse na začetku nastavljene na neskončno. Ima seznam že obravnavanih vozlišč, ki je na začetku prazen in seznam vozlišč, ki jih še mora pregledati, ki na začetku seveda vsebuje vsa vozlišča.

Algoritem začne proces tako, da prestavi vir s med obravnavana vozlišča in nastavi za vsa sosednja vozlišča v, $d(s, v) = \omega(s, v)$, kjer je $\omega(s, v)$ utež povezave med s in v. Nato algoritem preprosto ponavlja naslednji korak: Iz seznama nepregledanih vozlišč izbere tisto vozlišče v, ki ima trenutno najkrajšo pot $d(s, v)$ in ga premakne v seznam že obravnavanih vozlišč. Nato pregleda vsa sosednja vozlišča v-ja in za vsakega preveri, če je pot preko v-ja krajša (preveri, če je $d(s, v) + \omega(v, u) < d(s, u)$) in če je, popravi vrednost $d(s, u)$. To ponavlja dokler niso vsa vozlišča pregledana.

Algoritem se tudi da enostavno dopolniti, da nam ne izračuna samo najkrajših poti, ampak tudi poda drevo, ki predstavlja najkrajše poti. Algoritem v tem primeru skrbi še za en seznam, in sicer za vsako vozlišče v si zapomni še njegovega predhodnika na poti do s, označenega s $pred(v)$. Algoritem vsakič ko popravi vrednost $d(s,v)$ v $d(s,v)+\omega(v,u)$, označi še predhodnika za v, da je ta u. S pomočjo predhodnikov lahko najkrajšo pot med s in v najdemo kot $v, pred(v), pred(pred(v)), pred(pred(pred(v))), ..., s$. To drevo ni nujno enolično, ampak to za algoritem ni pomembno.

Algoritem zapišemo v psevdokodi:

function SSSP($G=(V,E)$, ω)
  $P=\emptyset$
  $T=V$
  $d(s,v)=\infty$ za vse $v\in V$, $d(s,s)=0$
  while $P\neq V$ **do**
    $v=$ vozlišče z najmanjšim $d(s,v)$
    $P=P\cup\{v\}$, $T=T\backslash\{v\}$
    for u je sosed v in $u\in T$ **do**
      if $d(s,u)>d(s,v)+\omega(v,w)$ **then**
        $d(s,u)=d(s,v)+\omega(v,w)$
        $pred(u)=v$
      end if
    end for
  end while
end function

Ko algoritem konča, nam vrednosti $d(s,v)$ povedo dolžino najkrajše poti med s in v. To je res, saj ko neko vozlišče izberemo iz seznama nepregledanih vozlišč, izberemo vedno takega, ki ima minimalen $d(s,v)$. To pa pomeni, da če obstaja neka druga pot do njega, ta pot gotovo poteka preko nekega drugega vozlišča, ki je že sam bolj oddaljen od vira. Torej je ta pot daljša.

Preštejmo, koliko operacij potrebuje algoritem. Vsako vozlišče in vsako povezavo preveri samo enkrat, tako da tukaj ni problema. Največja zahtevnost se pojavi v koraku, ko želimo izbrati vozlišče z najmanjšim $d(s,v)$. Zato je pametno vozliča in vrednosti $d(s,v)$ hraniti v takšni strukturi, ki omogoča hitrejše iskanje najmanjšega elementa in še vedno ne potrebuje preveč časa za spreminjanje vrednosti $d(s,v)$. Izkaže se, da je najbolj uporabna t. i. Fibonaccijeva kopica (Fibonacci heap) s katero ima algoritem zahtevnost $O(m+n\log(n))$, kjer je n število vozlišč in m število povezav. V primeru, ko povezave nimajo uteži, ta problem imenujemo Breadth-First Search (BFS) in lahko uporabimo kar navaden seznam in imamo zahtevnost $O(m+n)$.

2.2.2 Najkrajše poti med vsemi pari vozlišč

V primeru, ko želimo poznati najkrajše poti med vsemi pari vozlišč in ne samo poti iz enega vira, imamo problem znan pod imenom All-Pairs Shortest Paths problem (APSP). Seveda lahko ta problem prevedemo na prejšnjega, tako da za vsako vozlišče poiščemo drevo najkrajših poti. Ta rešitev je celo uporabna za grafe z relativno malo povezavami in potrebuje $O(nm+n^2)$ operacij, če povezave nimajo uteži. V splošnem primeru je ta metoda seveda preveč potratna in potrebujemo boljši algoritem. Idejo za algoritem nam da preprosta opazka, da če $d(v,u)>d(v,w)+d(w,u)$ za neke $v,u,w\in V$ potem $d(v,u)$

seveda ni najkrajša pot in jo lahko popravimo. S pomočjo te opazke in izreka Warshalla [11] je Floyd [6] razvil APSP algoritem, ki preprosto za vse možne pare $v, u, w \in V$ preveri če neenakost drži, in če drži, popravi najkrajšo pot. Torej izvede trojno zanko po vozliščih in zato potrebuje $O(n^3)$ operacij.

Floyd-Warshallov algoritem lahko zapišemo v psevdokodi:

function APSP($G = (V, E)$, ω)
  $d(v, v) = 0$
  $d(v, u) = \omega(v, u)$, če $\{v, u\} \in E$, sicer $d(v, u) = \infty$
  $pred(u, v) = u$, če $\{v, u\} \in E$, sicer $pred(u, v) = 0$
  for $v \in V$ **do**
    for $u \in V$ **do**
      for $w \in V$ **do**
        if $d(u, w) > d(u, v) + d(v, w)$ **then**
          $d(u, w) = d(u, v) + d(v, w)$
          $pred(u, w) = pred(v, w)$
        end if
      end for
    end for
  end for
end function

Ni očitno, da ko algoritem zaključi, nam vrednosti $d(v, u)$ res povedo dolžine najkrajših poti. Zato potrebujemo naslednji izrek.

Izrek 2.10 *Naj bo $G = (V, E)$ graf in $\omega : E \to \mathbb{R}$ otežitev povezav. Potem s Floyd-Warshallovem algoritmom izračunani $d(v, u)$-ji res podajo najkrajše poti med poljubnimi $v, u \in V$.*

Dokaz. Dokažimo z indukcijo, da po k pregledanih vozliščih v v prvi zanki algoritma nam vrednosti $d(u, w)$ povedo dolžino najkrajše poti med poljubnima u in w iz V, ki ima vmesna vozlišča samo iz množice že pregledanih vozlišč. Torej ko se bo zanka iztekla in bomo pregledali vsa vozlišča bodo vrednosti $d(u, w)$ za poljubna $u, w \in V$ res podala vrednost najkrajše poti.

Indukcijska hipoteza očitno drži, ko algoritem zaženemo. Poglejmo kaj se zgodi v $k+1$-tem koraku: Po indukcijski hipotezi poznamo dolžino take najkrajše poti med poljubnima u in w iz V, ki poteka po že pregledanih vozliščih. Radi bi preverili, če obstaja krajša pot preko vozlišča v. Prav tako vemo dolžino najkrajše poti med u in v ter med v in w, ki potekata po že pregledanih vozliščih. V $k + 1$ zanki se lahko zgodita dve stvari.

1. Najkrajša pot med poljubnima u in w, ki poteka po že pregledanih vozliščih in vozlišču v, ne vsebuje vozlišča v. V tem primeru tudi ne velja $d(u, w) > d(u, v) + d(v, w)$ zato algoritem v tekoči zanki dolžine poti ne popravi.

2. Najkrajša pot med u in w, ki poteka po že pregledanih vozliščih in vozlišču v, vsebuje vozlišče v. Torej $d(u, w) > d(u, v) + d(v, w)$ in algoritem v tekoči zanki dolžino poti popravi.

Torej algoritem popravi poti, da po $k+1$-tem koraku velja trditev, kar dokaže indukcijsko hipotezo in konča dokaz.

□

2.2.3 Dinamični APSP

Kot že omenjeno, veliko grafov iz resničnega življenja hitro spreminja svojo obliko, kljub temu bi pa še vedno radi poznali najkrajše razdalje med vozlišči. Zato bi radi imeli algoritem, ki ob izbrisu ali pojavitvi novega vozlišča hitro popravi dolžine novih poti. Prej opisani algoritem nam tega seveda ne omogoča. Izkaže se, da so algoritmi za rešitev tega problema veliko kompleksnejši in problem še vedno ostaja aktualen. Zato opis teh algoritmov presega meje tega dela. Navedimo samo nekaj referenc. Demerescu in Italiano [3] sta opisala algoritem za dinamični APSP z nenegatinimi utežmi, ki potrebuje $O(n^2 \log^3 n)$ operacij za vzdrževanje najkrajših poti. Prav tako je tudi Thorup [10] predstavil svoj algoritem, ki potrebuje $O(n^2(\log^2(m + n/m)))$ operacij. Po drugi strani sta Roditty in Zwick [9] predstavila zelo učinkovit verjetnostni algoritem z visoko verjetnostjo za grafe z malo povezavami, ki rezultat vrne v amortiziranem času $O(m \log n)$.

2.2.4 Računanje največje lastne vrednosti

Veliko različnih meritev centralnosti temelji na izračunu lastnih vrednosti danih matrik. To je netrivialen in računsko zahteven problem in pomemben del sodobne matematike je osredotočen za reševanje le tega. Do velike računske zahtevnosti pride pri zelo velikih matrikah, in zato obstajajo metode razvite za različne matrike posebnih oblik. Primer takega algoritma je znan Arnoldijev algoritem.

Najpreprostejša metoda za računanje največje lastne vrednosti je dobro znana potenčna metoda in je predstavljena v naslednjem algoritmu:

function POTENČNA METODA(A)
  q, λ začetna približka
  ϵ izbrana natančnost
  while $Aq - \lambda > \epsilon$ **do**
    $q = \frac{Aq}{||Aq||}$
    $\lambda = q^T Aq$
  end while
end function

2.3 Specifični centralni algoritmi

V tem poglavju bomo podrobneje predstavili indeks posrednika na najkrajši poti, shortcut indeks in razvili algoritme za njun izračun.

2.3.1 Indeks centralnosti vmesnika na najkrajši poti

Spomnimo se, da je indeks centralnosti vmesnika na najkrajši poti vozlišča $v \in V$ enak številu:

$$c_B(v) = \sum_{s \neq v} \sum_{t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

kjer je σ_{st} število najkrajših poti med s in t, število $\sigma_{st}(v)$ pa je število najkrajših poti med s in t, ki potekajo skozi v. Vidimo, da indeks centralnosti vmesnika na najkrajši poti meri pomembnost vozlišča na podlagi števila najkrajših poti. Toda to število predstavlja

le števec zgornjega ulomka, zakaj pa normiramo in delimo z σ_{sv}, pa si poglejmo na sliki 2.3.

Označimo z $c_C(v) := \sum_{s \neq v} \sum_{t \neq v} \sigma_{st}(v)$. Vidimo, da je velja $c_C(u_i) = \binom{8}{2} = 28$, $c_C(v) = \binom{8}{2} = 28$. To pomeni, da bi bili v primeru ocenjevanja pomembnosti vozlišča na podlagi indeksa c_C ti vozlišči enako pomembni. Očitno pa dobimo z odstranitvijo vozlišča v v drugem grafu dve ločeni komponenti, prvi graf pa ostane povezan. Za razliko od indeksa c_C, ima centralni indeks c_B v vozlišču u_i vrednost $\binom{8}{2} \cdot \frac{1}{3} = 8\frac{1}{3}$, v vozlišču v pa vrednost $\binom{8}{2} = 28$. Torej indeks c_B zazna, da je vozlišče v bolj pomembno od vozlišča u_i, zato je bolje uporabljati centralni indeks c_B.

V nadaljevanju bi radi za dano vozlišče v naredili čim boljši algoritem za računanje indeksa centralnosti vmesnika na najkrajši poti $c_B(v)$. Torej direktno iz definicije, bi algoritem za izračun indeksa $c_B(v)$ naredili tako, da bi najprej izračunali število najkrajših poti med vsemi vozlišči grafa. Potrebujemo še nekaj definicij. Definirajmo $\text{pred}(s, v)$, kot množico vozlišč, ki so sosede od v in ležijo na najkrajši poti med s in v. Vozlišče v je na najkrajši poti med vozliščema s in t natanko tedaj, ko velja $d(s, t) = d(s, v) + d(v, t)$. Torej velja

$$\sigma_{sv} = \sum_{u \in \text{pred}(s,v)} \sigma_{su}.$$

Iz definicije vidimo, da velja $\text{pred}(s, v) = \{s\}$ za vsak $v \in N(s)$, kjer smo z $N(s)$ označili množico sosedov vozlišča s. Da bomo s pomočjo te rekurzije izračunali σ_{sv}, moramo poznati množico $\text{pred}(s, v)$. Spomnimo se, da nam Dijkstrov algoritem vrne drevo najkrajših poti, torej shranjuje le en element iz množice $\text{pred}(s, v)$. V spodnjem algoritmu pa bomo videli, da lahko računanje množice $\text{pred}(s, v)$ enostavno vpletemo v Dijkstrov algoritem. V Dijkstrov algoritem lahko vpletemo tudi računanje σ_{sv} in sicer preko rekurzivne formule, ki smo jo zgoraj podali. Prilagojeni Dijkstrov algoritem izgleda takole

function SSSP($G = (V, E)$, ω)
  $P = \emptyset$
  $T = V$
  $d(s, v) = \infty$ for all $v \in V$, d(s,s)=0
  while $P \neq V$ **do**
    $v =$ vozlišče z najmanjšim $d(s, v)$
    $P = P \cup \{v\}$, $T = T \backslash \{v\}$
    for u je sosed v in $u \in T$ **do**
      if $d(s, u) > d(s, v) + \omega(v, u)$ **then**
        $d(s, u) = d(s, v) + \omega(v, u)$
        $\text{pred}(s, u) = v$
        $\sigma_{su} = \sigma_{sv}$
      end if
      if $d(s, u) = d(s, v) + \omega(v, u)$ **then**
        $\text{pred}(s, u).append(v)$
        $\sigma_{su} += \sigma_{sv}$
      end if
    end for
  end while
end function

Izkaže se, da je časovna zahtevnost prilagojenega Dijkstrovega algoritma kar enaka časovni zahtevnosti Dijkstrovega algoritma, torej $O(n + n \log n)$ (to pokažemo s pomočjo

Fibonaccijeve kopice). Ker velja $\sigma_{st}(v) = \sigma_{sv} \cdot \sigma_{vt}$, je $c_B(v) = \sum_{s \neq v} \sum_{t \neq v} \frac{\sigma_{sv} \cdot \sigma_{vt}}{\sigma_{st}}$, pri čemer koeficiente $\frac{\sigma_{sv} \cdot \sigma_{vt}}{\sigma_{st}}$ dobimo s pomočjo prilagojenega Dijkstrovega algoritma. Ker pa na $\binom{n-1}{2}$ načinov lahko izberemo vozlišči s in v, ki sta različni od v, vidimo, da je časovna zahtevnost res $O(n^2)$. Tako ima izračun indeksa centralnosti za vsako vozlišče $v \in V$ s tem postopkom časovno zahtevnost $O(n^3)$.

Brandes je predstavil algoritem za izračun indeksa centralnosti vseh vozlišč v grafu, ki ima časovno zahtevnost $O(mn + n^2 \log n)$ za utežene grafe (torej enako, kot je n-kratna časovna zahtevnost Dijkstrovega algoritma) Ta algoritem bomo sedaj opisali, najprej pa si oglejmo še nekaj definicij.

Definicija 2.11 Odvisnost para vozlišč $s, t \in V$ od vozlišča $v \in V$ označimo z $\delta_{st}(v)$ in jo definiramo kot

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

Odvisnost vozlišča $s \in V$ od vozlišča $v \in V$ označimo z $\delta_{s*}(v)$ in jo definiramo kot

$$\delta_{s*}(v) = \sum_{t \in V} \delta_{st}(v).$$

S pomočjo zgornjih definicij lahko indeks centralnosti vozlišča v zapišemo v bolj kompaktni obliki:

$$c_B(v) = \sum_{s \neq v} \delta_{s*}(v).$$

Izrek 2.12 *Za odvisnost $\delta_{s*}(v)$ vozlišča s od poljubnega vozlišča v velja formula*

$$\delta_{s*}(v) = \sum_{w : v \in pred(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_{s*}(w)).$$

Dokaz. Najprej definirajmo $\sigma_{st}(v, e)$ kot število najkrajših poti od s do t, ki vsebujejo $v \in V$ in $e \in E$. Nadalje definirajmo parno odvisnost vozlišč s, t od vozlišča v in povezave e kot $\delta_{st}(v, e) = \sigma_{st}(v, e)/\sigma_{st}$. Vidimo, da velja

$$\delta_{s*}(v) = \sum_{t \in V} \delta_{st}(v) = \sum_{t \in V} \sum_{w : v \in \mathrm{pred}(s,w)} \delta_{st}(v, (v, w)). \tag{2.1}$$

Sedaj bomo pokazali, da velja:

$$\sum_{w : v \in \mathrm{pred}(s,w)} \delta_{st}(v, (v, w)) = \begin{cases} \sum_{w : v \in \mathrm{pred}(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}} & \text{če } t = w, \\ \sum_{w : v \in \mathrm{pred}(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot \frac{\sigma_{st}(w)}{\sigma_{st}} & \text{če } t \neq w. \end{cases}$$

V primeru $t = w$ je število najkrajših poti od s do t, ki vsebujejo povezavo (v, w) enako številu najkrajših poti med s in v, kar pa je ravno σ_{sv}. Če pa je $t \neq w$, moramo pokazati, da je število najkrajših poti, ki vsebujejo v in povezavo (v, w) enako $\sigma_{sv}\sigma_{st}(w)/\sigma_{sw}$. Upoštevamo, da velja $\sigma_{st}(w) = \sigma_{sw}\sigma_{wt}$ in potem vidimo, da mora biti število najkrajših poti, ki vsebujejo v in povezavo (v, w), enako $\sigma_{sv}\sigma_{wt}$, kar pa drži. Torej smo zgornjo formulo dokazali.

Sedaj v enačbo 2.1 vstavimo formulo za $\delta_{st}(v, (v, w))$, ki smo jo izpeljali zgoraj ter zamenjamo vrstni red seštevanja in tako dobimo:

$$\delta_{st}(v,(v,w)) = \sum_{w:v\in \mathrm{pred}(s,w)} \sum_{t\in V} \delta_{st}(v,(v,w)) = \sum_{w:v\in \mathrm{pred}(s,w)} (\frac{\sigma_{sv}}{\sigma_{sw}} + \sum_{t\in V-w} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot \frac{\sigma_{st}(w)}{\sigma_{st}})$$
$$= \sum_{w:v\in \mathrm{pred}(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}}(1+\delta_{s^*}(w)),$$

kar pa smo želeli pokazati.

□

Izrek 2.13 *Indeks centralnosti vseh vozlišč lahko izračunamo s časovno zahtevnostjo* $O(mn+n^2\log n)$.

Dokaz. Najprej s pomočjo prilagojenega Dijkstrovega algoritma izračunamo n dreves najkrajših poti (n za vsako vozlišče $s \in V$) ter vse množice $\mathrm{pred}(s,v)$ in število najkrajših poti σ_{sv}. Vemo, da zgornje podatke lahko izračunamo s časovno zahtevnostjo $O(mn + n^2\log n)$. Nato izračunamo $\delta_{s^*}(v)$ za vozlišče v_1, ki je najbolj oddaljeno od vozlišča s. Vemo, da zanj velja $\delta_{s^*}(v_1) = 0$. Potem iz množice vozlišč $V - \{s, v_1\}$ izberemo tisto vozlišče v_2, za katerega velja da najbolj oddaljeno od vozlišča s. Sedaj vrednost $\delta_{s^*}(v_2)$ izračunamo s pomočjo rekurzivne formule iz izreka. Če je množica $w \in V; v \in \mathrm{pred}(s,w)$ prazna velja $\delta_{s^*}(v_2) = 0$, v nasprotnem primeru pa velja rekurzivna zveza

$$\delta_{s^*}(v) = \sum_{w:v\in \mathrm{pred}(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}}(1+\delta_{s^*}(w)),$$

pri čemer se ne more zgoditi, da bi v množici $\mathrm{pred}(s,w)$ imeli vozlišče v, za katero ne bi poznali vrednosti $\delta_{s^*}(v)$.

Ta postopek induktivno nadaljujemo ter tako vidimo, da ima izračun vrednosti $\delta_{s^*}(v)$ zanemarljivo časovno vrednost v primerjavi z prej omenjeno časovno zahtevnstjo $O(mn+n^2\log n)$, prav tako ima zanemarljivo časovno vrednost končen izračun indeksa centralnosti za vsa vozlišča, saj moramo namreč n-krat izračunati vsoto $\sum_{s\neq v}\delta_{s^*}(v)$ (spomnimo se, da velja $c_B = \sum_{s\neq v}\delta_{s^*}(v)$), kar pa nanese časovno zahtevnost $O(n^2)$, torej je časovna zahtevnost indeksa centralnosti vseh vozlišč res enaka $O(mn + n\log n)$.

□

2.3.2 Shortcut indeks

V prejšnjem podpoglavju smo obravnavali indeks, ki meri pomembnost vozlišč, v tem poglavju pa bi radi na podobne način definirali nek indeks, ki bo meril pomembnost povezav. Ta indeks bomo imenovali shortcut indeks, meril pa bo maksimalno povečanje cene najkrajše poti med poljubnima vozliščema u, v v V, če odstranimo dano povezavo e.

Shortcut indeks povezave $e \in E$ označimo s $c_S(e)$ in je enak

$$c_S(e) = \max_{u,v\in V}\{d_{G\setminus e}(u,v) - d_G(u,v)\}.$$

Vrednost povečanja označimo z $\mathrm{sc}(e)$, kjer je $e \in E$. V tem razdelku bomo podali algoritem za izračun shortcut indeksa za vse povezave usmerjenega grafa $G = (V, E)$.

Izberimo si neko vozlišče u. Označimo z α_i dolžino najkrajše poti, od vozlišča u do vozlišča i (prej smo označevali $d(u,i)$). Nadalje s $\tau_i \in V$ označimo drugo vozlišče najkrajše poti od u do i (če je to vozlišče enolično določeno, v nasprotnem primeru pa pišemo $\tau_i = ND$). Torej, če je $\tau_i = ND$ potem to implicira, da obstajata vsaj dve poti dolžine α_i od u do i, ki se začneta z različnima povezavama. Nadalje definiramo še vrednost β_i, kot razdaljo najkrajše poti od u do i, ki ne vsebuje vozlišča τ_i. Če taka pot ne obstaja, potem označimo $\beta_i = \infty$, če pa je $\tau_i = ND$ pa definiramo $\beta_i = \alpha_i$. Sedaj si izberimo začetno vozlišče u in predspostavimo, da smo izračunali vse vrednosti α_v, τ_v in β_v za neko vozlišče v. Potem je sc vrednost za povezavo (u,v) enaka α_i, če $\tau_v \neq v$, torej če povezava (u,v) ni edina najkrajša povezava od u do v. Če pa je $\tau_v = v$, potem pa je β_v vrednost sc za (u,v). Torej nam preostane izračunati vrednosti α_i, τ_i in β_i za vsak $i \in V$. Podali jih bomo v rekurzvni obliki. Najprej definiramo $\alpha_u = 0$, $\tau_u = \emptyset$, $\beta_u = \infty$ (u je začetno vozlišče).

Potem vidimo, da velja

$$\alpha_j = \min_{i:(i,j)\in E}(\alpha_i + \omega(i,j)).$$

Preden definiramo rekurzivno vrednost za τ_j, definirajmo še množico I_j:

$$I_j = \{i \mid (i,j) \in E \text{ in } \alpha_j = \alpha_i + \omega(i,j)\}.$$

Potem velja

$$\tau_j = \begin{cases} j & \text{če } I_j = \{u\}, \\ a & \text{če } a = \tau_i \text{ za vsak } i \in I_j, \\ ND & \text{sicer.} \end{cases}$$

Vrednost τ_j je definirana samo, če se vse najkrajše poti do vozlišča j začnejo z isto povezavo, kar se zgodi samo v primeru, ko je vrednost τ_i na vseh vozliščih $i \in I_j$ enaka. Če velja $\tau_j = ND$, potem je $\beta_j = \alpha_j$, drugače pa je

$$\beta_j = \min\left\{\min\{\beta_i + \omega(i,j); i : (i,j) \in E, \tau_i = \tau_j\}, \min\{\alpha_i + \omega(i,j); i : (i,j) \in E, \tau_i \neq \tau_j\}\right\}.$$

Ker je α_i odvisen le od α_j, ki so manjši od α_i in ker enako velja tudi za β_i, vidimo, da lahko zgornje rekurzije enostavno vstavimo v Dijkstrov algoritem in tako dobimo analogno kot v primeru indeksa centralnosti, tudi v tem primeru lahko shortcut indeks vseh povezav izračunamo s pomočjo n-krat uporabljenega Dijkstrovega algoritma in tako dobimo da je časovna zahtevnost shortcut indeksov vseh povezav enaka $O(nm + n^2 \log n)$.

2.4 Hitri približki algoritmov

Spomnimo se, kaj smo se o indeksih centralnosti naučili do sedaj. V prvem razdelku smo se seznanili z nekaterimi indeksi, v drugem razdelku smo si ogledali nekaj osnovnih algoritmov za njihovo računanje, v tretjem razdelku pa smo spoznali še malo naprednejše algoritme. Med drugim nas je zanimala tudi časovna zahtevnost predstavljenih algoritmov. Izkazalo se je, da so izračunljivi v polinomskem času, kar je v splošnem dober rezultat. Problem pa se pojavi na velikih omrežjih, ko polinomsko časovno zahtevni algoritmi niso nujno izračunljivi v realnem času. Ta fenomen se veliko pojavlja tudi med analiziranjem grafa spleta. Za velike grafe zato ne želimo izvajati celotnih algoritmov, temveč samo njihove približke. Ena možnost za tak približek je izvedba algoritma samo na nekaterih vozliščih in sklepanje o vrednostih indeksa na preostalih vozliščih. Temeljno vprašanje pri tem pa je, kako točne ocene pridobimo. V tem razdelku obravnavamo aproksimacijske algoritme, ki zagotavljajo dober kompromis med časom izvajanja in natančnostjo.

2.4.1 Aproksimacija centralnosti, ki temeljijo na izračunu vseh parov najkrajših potih

Kot smo omenili, je lahko računanje določenih indeksov centralnosti zelo časovno potratno. To se nanaša tudi na indekse, ki temeljijo na računanju najkrajših poti za vse pare vozlišč. V večini primerov je zato bolje izračunati dobre približke za vrednosti indeksov, saj je to veliko hitrejše. Kot primer navedimo tehniko slučajnega vzorčenja, ki sta jo razvila Eppstein in Wang (glej [1]). S to tehniko lahko izračunamo bližinjsko centralnost in centralnost posrednika na najkrajši poti (glej tretji razdelek) vseh vozlišč v uteženem, neusmerjenem grafu precej hitreje kot z eksaktnim algoritmom. Poglejmo si natančneje njuno idejo.

Eppstein-Wangova aproksimacija bližinske centralnosti

V drugem razdelku smo definirali eno verzijo indeksa bližinske centralnosti. Indeks bližinske centralnosti bi lahko definirali še drugače. Za vozlišče $v \in V$ je to izraz $c_{C'}(v) = \frac{\sum_{x \in V} d(v,x)}{n-1}$. Vrednost $c_{C'}(v)$ bi želeli samo oceniti z neko natančnostjo. Eppstein in Wang predlagata slučajni izbor K vzorčnih vozlišč $v_1, \ldots, v_K$ iz V in izračun vseh najkrajših poti do vseh preostalih vozlišč samo za vzorčna vozlišča. Tako dobimo točne vrednosti indeksov $c_{C'}(v_i)$. Vrednosti $c_{C'}(v)$ za ostala potem ocenita s formulo:

$$\hat{c}_{C'}(v) = \frac{n}{K(n-1)} \sum_{i=1}^{K} d(v, v_i).$$

Ta indeks torej izračuna povprečno razdaljo od v do K vzorčnih vozlišč, potem pa množi z n, da oceni vsoto razdalj od v do vseh vozlišč in deli z $n-1$. Ker $c_{C'}$ in $\hat{c}_{C'}$ oba računata povprečni razdalji v grafu, sta njuni pričakovani vrednosti enaki za vsa vozlišča $v \in V$ in vsak vzorec K vozlišč.

Povzemimo idejo algoritma:

Eppstein-Wangov algoritem za izračun bližinske centralnosti:
1. Slučajno izberi K vzorčnih vozlišč $v_1, v_2, \ldots, v_K$.
2. Za vsako vzorčno vozlišče v_i reši problem vseh parov najkrajših poti.
3. Za vsako vozlišče v izračunaj $\hat{c}_{C'}(v)$.

Sedaj pa pridemo do ključnega problema predstavljenega algoritma. Najpomembnejše vprašanje je, koliko vzorčnih vozlišč potrebujemo, da ocenimo indekse $c_{C'}$ z željeno natančnostjo. Na to vprašanje se da odgovoriti s Hoeffdingovo oceno (za dokaz glej [2]) iz teorije verjetnosti:

Lema 2.14 (Hoeffdingova ocena) *Naj bodo spremenljivke $x_1, x_2, \ldots, x_K$ neodvisne in naj velja $a_i \leq x_i \leq b_i$, kjer so $a_i, b_i \in \mathbb{R}$. Če z μ označimo povprečje $E\left(\frac{\sum x_i}{K}\right)$, potem za vsak $\epsilon > 0$ velja*

$$P\left(\left|\frac{\sum x_i}{K} - \mu\right| \geq \epsilon\right) \leq 2 \cdot e^{\frac{-2K^2\epsilon^2}{\sum (b_i - a_i)^2}}.$$

Če v zadnjo lemo vstavimo $x_i = \frac{nd(v_i,u)}{n-1}$, $\mu = c_{C'}(v)$, $a_i = 0$ in $b_i = \frac{n\Delta}{n-1}$, lahko ocenimo verjetnost, da je napaka pri ocenjevanju $c_{C'}(v)$ s $\hat{c}_{C'}(v)$ več kot ϵ:

$$\begin{aligned} P\left(\left|\frac{\sum x_i}{K} - \mu\right| \geq \epsilon\right) &= P\left(|\hat{c}_{C'}(v) - c_{C'}(v)| \geq \epsilon\right) \leq \\ &\leq 2 \cdot e^{\frac{-2K^2\epsilon^2}{\sum (b_i - a_i)^2}} \\ &= 2 \cdot e^{\frac{-2K^2\epsilon^2}{K(\frac{n\Delta}{n-1})^2}} = \\ &= 2 \cdot e^{-2\left(\frac{n-1}{n}\right)^2\left(\frac{K\epsilon^2}{\Delta^2}\right)} \end{aligned}$$

Če sedaj za poluben $\hat{\epsilon} > 0$ postavimo še $\epsilon = \hat{\epsilon}\Delta$ in uporabimo $K = \frac{\log n}{2\hat{\epsilon}^2}$ vzorcev, z nekaj algebre v zgornjem izrazu hitro ugotovimo, da je verjetnost za napako večjo od $\hat{\epsilon}$ največ $\frac{1}{n}$.

Komentirajmo še, kaj s to aproksimacijo pridobimo na časovni zahtevnosti. Časovna zahtevnost ASSP algoritma je $O(n(n+m))$ za neutežene grafe in $O(n(m + n\log n))$ za utežene grafe (glej drugi razdelek). Če pa naredimo samo K SSSP algoritmov, dobimo časovni zahtevnosti $O(K(n+m))$ in $O(K(m + n\log n))$. Za $K = \frac{\log n}{2\hat{\epsilon}^2}$ ugotovimo, da se časovna zahtevnost za utežene grafe spremeni iz $O(n(m+n\log n))$ na $O(m\log n + n\log^2 n))$

Eppstein-Wangova aproksimacija centralnosti posrednika na najkrajši poti

Tudi centralnost posrednika na najkrajši poti temelji na ASSP algoritmu, zato lahko zanjo uporabimo Eppstein-Wangovo aproksimacijo. Spet se spomnimo tretjega razdelka, kjer je centralnost posrednika na najkrajši poti izračunana prek formule $c_B(v) = \sum_{u\in V} \delta_{u\cdot}(v)$, kjer je $\delta_{u^*}(v)$ odvisnost vozlišča u od vozlišča v. $c_B(v)$ ocenimo enako kot v primeru indeksa $c_C(v)$ kot

$$\hat{c}_B(v) = \frac{n}{K}\sum_{i=1}^{K} \delta_{v_i^*}(v).$$

Pričakovana vrednost $\hat{c}_B(v)$ je spet enaka $c_B(v)$ za vse K in vse v. V Hoeffdingovo oceno lahko postavimo $x_i = n\delta_{v_i^*}$, $\mu = c_B(v)$, $a_i = 0$ in $b_i = n(n-2)$ (saj je totalna odvisnost lahko največ $n-2$). Dobimo:

$$P\left(|\hat{c}_B(v) - c_B(v)| \geq \epsilon\right) \leq 2 \cdot e^{-2\left(\frac{K\epsilon^2}{(n(n-2))^2}\right)}.$$

Če za ϵ vstavimo $\hat{\epsilon}(n(n-2))$ in za $K = \frac{\log n}{2\hat{\epsilon}^2}$, z nekaj algebre ugotovimo, da je verjetnost za napako večjo od $\hat{\epsilon}(n(n-2))$ samo $\frac{1}{n}$.

Iz drugega razdelka vemo, da lahko $\delta_{v_i\cdot}(v)$ izračunamo v $O(n+m)$ korakih za neutežene grafe in $O(m+n\log n)$ za utežene grafe. Za $K = \frac{\log n}{2\hat{\epsilon}^2}$ tako dobimo časovni zahtevnosti za izračun indeksa vmestnosti $O\left(\frac{\log n}{2\hat{\epsilon}^2}(n+m))\right)$ in $O\left(\frac{\log n}{\hat{\epsilon}^2}(m + n\log n)\right)$. Faktor izboljšanja v primerjavi z eksaktnim algoritmom je spet $\frac{K}{n}$, ki nadomesti n iz celotnega algoritma.

Opomba 2.15 Pred nadaljevanjem še enkrat poudarimo, da lahko predstavljeno tehniko uporabimo za oceno vseh indeksov centralnosti, ki temeljijo na seštevanju določene količine po vseh vozliščih. Namesto tega seštevamo samo po podmnožici vozlišč in normaliziramo.

2.4.2 Aproksimacija spletnih centralnosti

Prišli smo do najpomembnejšega dela razdelka, tj. aproksimativnih in pospešitvenih tehnik za izračunavanje spletnih centralnosti. V prvem razdelku smo se seznanili s PageRank metodo, na kateri še vedno temeljijo izračuni centralnosti večine spletnih iskalnikov. Zato se bomo posvetili predvsem pospešitvenim tehnikam za izračun slednje. Obstaja več metod za pospešitev:

1. aproksimacija s cenejšimi izračuni (običajno se izognemu množenju z matriko),
2. pospešitev konvergence,
3. reševanje linearnega sistema enačb namesto reševanja problema lastnih vrednosti,
4. uporaba dekompozicij grafa spleta,
5. nadgradnja namesto ponovnega računanja.

Pobližje si poglejmo nekatere izmed zgoraj navedenih tehnik.

Pospešitev konvergence

Osnova za to pospešitveno metodo je potenčna metoda za določitev lastnega vektorja, ki pripada največji lastni vrednosti.

Ker vsaka iteracija potenčne metode zahteva množenje z matriko A, ki je za graf spleta zelo drago, je naš cilj zmanjšati število iteracij. Aitkenova metoda, temelji na predpostavki, da lahko približek $x^{(k-2)}$ zapišemo kot linearno kombinacijo prvih dveh lastnih vektorjev, to sta u in v. S to predpostavko sta tudi naslednja dva približka $x^{(k-1)}$ in $x^{(k)}$ linearni kombinaciji prvih dveh lastnih vektorjev:

$$x^{(k-2)} = u + \alpha v,$$

$$x^{(k-1)} = u + \alpha\lambda_2 v,$$

$$x^{(k)} = u + \alpha\lambda_2^2 v.$$

Če definiramo

$$y_i = \frac{\left(x_i^{(k-1)} - x_i^{(k-2)}\right)^2}{x_i^{(k)} - 2x_i^{(k-1)} + x_i^{(k-2)}},$$

z nekaj algebre pridemo do $y = \alpha v$ in tako

$$u = x^{(k-2)} - y.$$

Pripomnimo, da je predpostavka, da je $x^{(k-2)}$ linearna kombinacija vektorjev u in v samo aproksimacija, zato je tudi zadnji izraz za u samo aproksimacija za lastni vektor, ki ga potem izračunamo z običajno potenčno metodo.

Opomba 2.16 Obstajajo tudi posplošitve Aitkenove metode, ki temeljijo na predpostavki, da je približek $x^{(k-2)}$ linearna kombinacija prvih k lastnih vektorjev in izpeljemo podobne algoritme pravkar predstavljenemu.

Eksperimenti kažejo, da so te metode precej hitrejše od običajne potenčne metode.

Metoda linearnih sistemov

Vsak problem lastnih vrednosti $Ax = \lambda x$ lahko zapišemo kot homogen linearni sistem $(A-\lambda I)x = 0$. Najenostavnejši algoritem za reševanje linearnega sistema pa je Jacobijeva iteracija. To lahko uporabimo na PageRank sistemu

$$(1 - dP)c_{PR} = (1 - d)1_n.$$

Toda problem z Jacobijevo iteracijo je ta, da v splošnem ni boljša od potenčne metode. Namesto tega lahko uporabimo Gauss-Seidlovo iteracijo, definirano z

$$c_{PR}^{k+1}(i) = (1 - d) + d \sum_{j<i} p_{ij} c_{PR}^{(k+1)}(j) + \sum_{j>i} c_{PR}^{(k)}(j).$$

Za $d = 0.9$ so eksperimenti na grafu spleta pokazali, da Gauss-Seidlova iteracija konvergira precej hitreje kot potenčna metoda.

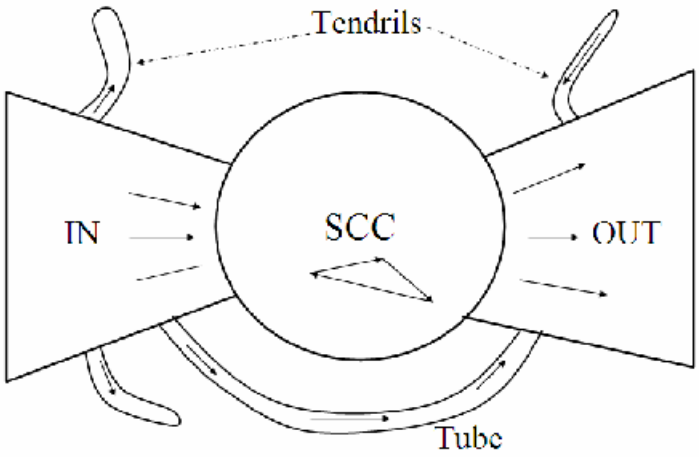
Vse do sedaj predstavljene tehnike pa lahko kombiniramo z nekaterimi dekompozicijskimi tehnikami, ki temeljijo na stukturi grafa spleta, ki jo bomo opisali v nadaljevanju. Izraba struktur grafa spleta se izkaže za najpomembnejšo možnost pospeševanja.

Dekompozicijske tehnike

Ker je graf spleta zelo velik in vsakodnevno raste, so raziskovalci predlagali, da se ga razbije na več manjših komponent. Tako določimo centralnosti znotraj posamezne komponente, nato pa združimo v izračun centralnosti v celem grafu.

Graf spleta kot "metuljček"

Prvo možnost, kako pogledati na graf spleta, je predlagal Broder z ekipo leta 1999 (glej [3]). Ugotovili so, da ima graf spleta obliko metuljčka, predstavljenega na sliki 2.4.



Slika 2.4: Graf spleta po Broderju

Poskusimo utemeljiti, kako so prišli do teh ugotovitev. Njihova delitev vozlišč grafa je temeljila na PageRank metodi za iskalnik AltaVista. Usmerjena povezava med vozliščema

i in j, ki predstavljata spletni strani, obstaja, če je na spletni strani i povezava do spletne strani j. Ugotovili so, da obstaja natanko ena velika strogo povezana komponenta grafa spleta. Torej se da iz vsakega vozlišča priti po neki poti do vsakega drugega vozlišča. Razlog za to je v delovanju iskalnikov. Ti imajo povezave do večine glavnih univerzitetnih strani, velikih podjetij in vladnih agencij. Iz teh strani pa se da doseči veliko strani na spletu. Te manjše strani pa zelo verjetno spet kažejo nazaj na te velike strani in tudi na iskalnike. To nam sklene en velik krog, ki predstavlja strogo povezano komponento grafa. Da ne obstajata dve veliki komponenti je intuitivno jasno. Če bi namreč obstajali, ne bi smeli obstajati povezavi med njima. To pa je zelo malo verjetno. Poleg te velike strogo povezane komponente pa obstajata še dva velika razreda strani:

- strani, ki kažejo na to komponento, jih pa ne moremo doseči iz nje,
- strani, ki jih lahko dosežemo iz te velike komponente, vendar se nanjo ne moremo vrniti.

Ugotovili pa so obstoj še treh tipov strani:

- lovke iz zgornjih dve komponent,
- nekatere povezave med dvema komponentama,
- manjši izolirani kosi.

Idejo, kako sedaj to strukturo uporabiti za olajšanje računanja v metodi linearnih sistemov pa je predlagal Arasu. Ideja je, da matriko povezav P razbijemo na bločno zgornje trikotno matriko. Poglejmo si to na primeru. Če razdelimo bloke samo na tri največje komponente opisane zgoraj, potem ima prehodna matrika obliko:

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} \\ 0 & P_{22} & P_{23} \\ 0 & 0 & P_{33} \end{bmatrix}$$

Če dopuščamo še druge tri tipe povezav, v splošnem dobimo obliko:

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} & \dots & P_{1K} \\ 0 & P_{22} & P_{23} & \dots & P_{2K} \\ \vdots & \ddots & P_{33} & \dots & P_{3K} \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & P_{KK} \end{bmatrix} \bigoplus Q_1 \bigoplus Q_2 \bigoplus \dots \bigoplus Q_L.$$

Enako delitev naredimo tudi za vektor c_{PR} in velik problem razdelimo na več manjših ter rašujemo s pomočjo obratne substitucije:

$$\begin{aligned} (I - dQ_L)c_{PR,K+L} &= (1-d)1_{m_L}, \\ &\vdots \\ (I - dQ_1)c_{PR,K+1} &= (1-d)1_{m_1}, \\ (I - dP_{KK})c_{PR,K} &= (1-d)1_{n_K}, \\ (I - dP_{ii})c_{PR,i} &= (1-d)1_{n_i} + d\sum_{j=i+1}^{K} c_{PR,i}. \end{aligned}$$

Kamvarjeva gnezdeno bločna oblika grafa spleta

V tem delu si bomo pogledali bločno obliko grafa spleta, kot so je leta 2001 odkrili in izkoristili Kamvar s svojo ekipo (glej [4]). Ugotovili so naslednjo strukturo grafa spleta:

1. Graf spleta ima bločno strukturo.

2. Bloki so veliko manjši kot je celoten graf.

3. Znotraj blokov obstajajo še gneznedi bloki, ki ustrezajo domenam, gostiteljem in poddirektorijem v imenu strani.

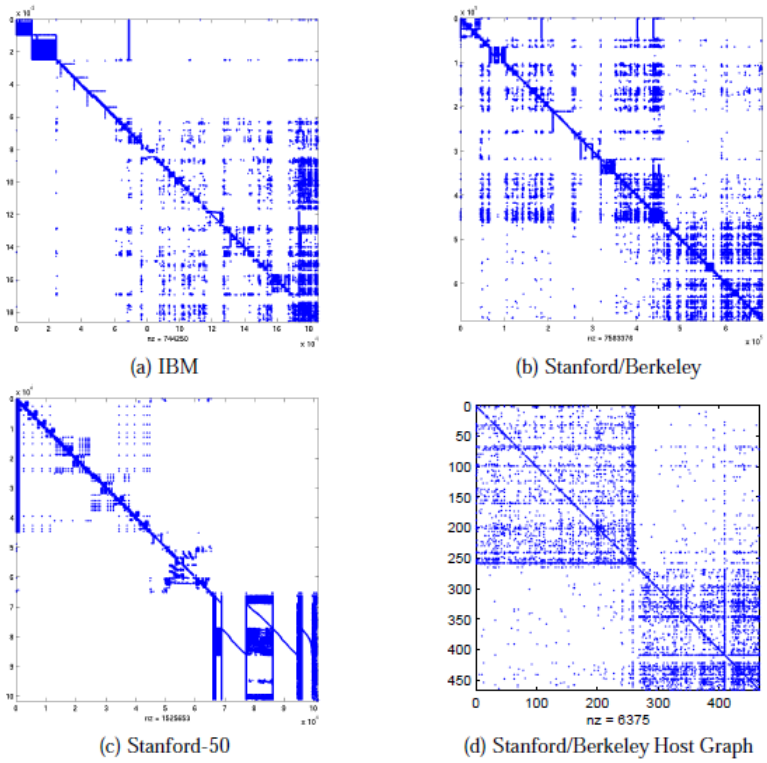
Nekaj primerov bločnih struktur posameznih komponent grafa spleta si lahko ogledamo na sliki 2.5. Komentirjamo te primere:

- Primer 1(a) prikazuje bločno zgradbo strani *ibm.com*. Opazimo lahko več blokov, ki ustrezajo različnim gostiteljem na tej strani. Prvi blok na primer pripada naslovu *almaden.ibm.com*, pri čemer je Almaden raziskovalni center družbe IBM. Poleg tega lahko opazimo štiri strukture na diagonali, ki izgledajo kot narobe obrnjena črka L. To pomeni, da znotraj naslovov, ki jim pripadajo te ti bloki, obstaja lepo vidna hierarhija od najpomembnejšega naslova do manj pomembnega. To si lahko predstavljamo kot pot od glavne mape do neke datoteke globoko v njej, tj. *mapa - podmapa - podpodmapa - ... - datoteka*. Na skrajnem desnem delu grafa lahko opazimo bolj gosto obarvano vertikalno območje. To kaže na dobro povezanost strani znotraj tega bloka. Izkaže se, da to ustreza stranem z naslovom *ibm.com/ ...*, torej tistim, ki pred *ibm.com* nimajo posebnega gostitelja.

- Primer 1(b) prikazuje domeno Stanford-Berkley-a. Tudi tu je bločna zgradba dobro opazna. Vidni so celo gnezdeni bloki. Največja dva bloka na sliki pa pripadata domenama $stanford.edu$ in $berkley.edu$.

- Primer 1(c) prikazuje po abecedi prvih 50 gostiteljev znotraj domene $stanford.edu$. Opazni so trije veliki bloki, ki pripadajo gostiteljem $acomp.stanford.edu$, tj. stran akademskega računalniškega centra, $cmgm.stanford.edu$, tj. spletnega informacijskega vira, in $daily.stanford.edu$, tj. spletne strani študentskega časopisa Stanford Daily. Če najprej opazujemo prvi blok, izstopa predvsem odebeljena vertikalna črta na začetku bloka. Ta pripada strani glavni strani $acomp.stanford.edu$, kajti večina podstrani kaže nazaj nanjo. Še posebej pa je zanimiva zgradba tretjega bloka, ki pripada Stanford Daily-ju. Opazimo nekaj vertikalnih črt, ki so med seboj povezane z drobnimi diagonalami. Prvih nekaj strani na strani $daily.stanford.edu$ je kar naslovnic časopisa iz zadnjih dni. Vertikalne črte predstavljajo te naslovnice, diagonale pa jih povezujejo. Diagonalnost črt izvira iz posebne leksikografske ureditve strani. Naslovnice kažejo na članke znotraj časopisa, ki jih predstavljajo srednji deli vertikalnih črt. Opazimo lahko sredinske odeblejene kvadrate, ki so povezave med članki. To so na na primer navezave na članke s sorodno vsebino ali članke z istim avtorjem. so med seboj povezani. Vidijo pa se še nekatere diagonalne povezave iz sredine, ki članek povezujejo s standardnimi stranmi, kot so *pošlji sporočilo uredniku* ali *komentiraj članek*.

- Primer 1(d) pa predstavlja omrežje Stanford-Berkley na nivoju gostiteljev. Na mestu (i, j) obstaja pikica, če sta gostitelja i in j povezana. Tudi tu lahko opazimo bločno zgradbo z dvema blokoma, ki očitno pripadata domenama *stanford.edu* in *berkley.edu* ter dva odebeljena za 90 stopinj v pozitivni smeri zavrtena L-ja v spodnjem desnem kotu blokov, ki kažeta krepko povezanost domen z ostalimi gostitelji.



Slika 2.5: Nekaj grafov s Kamvarjevo analizo

Na podlagi tega so predlagali naslednjo izboljšavo PageRank algoritma, ki so jo imenovali BlockRank algoritem:

BlockRank algoritem:
0. Razbij graf spleta na bloke $I \in \Gamma$, kjer je Γ indeksna množica..
1. Za vsak blok I izračunaj lokalne PageRank indeks $c_{PR(I)}(i)$ za vsak $i \in I$.
2. Naredi utežitev lokalnih indeksov, tako da upoštevaš pomembnost posameznega bloka.
3. Najdi začetni vektor za naslednji korak.
4. Uporabi standardni PageRank algoritem z začetnim vektorjem iz koraka 3.

V točki 1. in 4. zgornjega algoritma lahko uporabimo običajen PageRank algoritem. Glavni problem je, kako formalizirati točko 2. To se naredi tako, da se grafu spleta priredi bločni graf. Vsakemu bloku $I \in \Gamma$ ustreza vozlišče v tem grafu. Med vozliščema I in J obstaja usmerjena povezava $\vec{IJ}$ natanko tedaj, ko v prvotnem grafu obstaja povezava $\vec{ij}$ za neki vozlišči $i \in I$ in $j \in J$. Pri tem dopuščamo tudi zanke $\vec{II}$. Teže povezav ω_{IJ} pa določimo kot:

$$\omega_{IJ} = \sum_{i \in I, j \in J} a_{ij} c_{PR(I)}(i),$$

kjer je a_{ij} teža povezave $\vec{ij}$ v prvotnem grafu, $c_{PR(I)}(i)$ pa lokalni PageRank indeks vozlišča i v bloku I. Ni se težko prepričati, da je $\Omega = (\omega_{IJ})$ stohastična matrika in lahko uporabimo

običajen PageRank algoritem za bločni graf B, da dobimo teže povezav b_I.
Iz korakov 1 in 2 dobimo začetni vektor za korak 4, ki je

$$c_{PR}^{(0)}(i) = c_{PR(I)}(i) b_I \; \forall I \in \Gamma; \forall i \in I.$$

Kaj so prednosti BlockRank algoritma?

1. Glavna prednost je pohitritev algoritma, ki izvira iz dejstva, da ni potrebno hraniti celotnega PageRank vektorja v spominu računalnika, saj ta sestoji iz nekaj bilijonov vhodov. Takšen vektor je težko sploh spraviti v spomin računalnika. V algorimu vektor razbijemo na mnogo manjših vektorjev, na katerih potem izvajamo PageRank metodo. Izkaže se, da že s tem, ko izkoristimo bločno strukturo vektorja in na njem delamo običajen PageRank, pridobimo pol časa.

2. V BlockRank algoritmu veliko lokalnih PageRank vektorjev hitro skonvergira. Zato več časa lahko posvetimo slabo-konvergirajočim vektorjem. To so taki vektorji, ki pripadajo dobro gnezdenim blokom. To pomeni, da je markovska veriga prirejena stanjem teh blokov počasi konvergentna.

3. Lokalne PageRank vektorje lahko računamo vzporedno na več procesorjih, ki potem rezultate pošljejo glavnemu procesorju. Hkrati se izkaže tudi, da korak 1 v algoritmu lahko naredimo vzporedno s korakom 0, tj. določanjem prehodne matrike $A = (a_{ij})$. Ko je blok I preiskan, že lahko na njem izvedemo PageRank algoritem in korak 1 se zaključi tako rekoč hkrati s korakom 0, medtem ko pri običajnem algoritmu PageRank šele začnemo.

4. V veliko primerih lahko lokalne PageRank indekse uporabimo za dinamično računanje. Torej, če se zgodijo neke spremembe v grafu spleta samo za blok I, potem moramo ponoviti lokalni PageRank izračun za ta blok, za druge pa ga že poznamo. Potem nadaljujemo računanje direktno na koraku 2.

Kaj pa o navedenem algoritmu pove praksa?

1. Na sliki 2.6 si lahko pogledamo primerjavo lokalnih PageRank indeksov z globalnimi za domeno stanford.edu. Vidimo lahko, da je predvsem utežitev glavne strani, tj. *http://aa.stanford.edu*, v lokalnem PageRanku napačna, saj ne upoštevamo njene pozicije v grafu spleta. Drugače se lokalne pomembnosti ostalih podstrani lepo ujemajo z globalnimi. Ker se globalno pomembnost glavne strani množi z dva, je treba v grobem večini podstrani pomembnost razpoloviti. To pa je pri večini podstrani res.

2. Na sliki 2.7 si poglejmo, kaj so pokazali eksperimenti na modelu LARGEWEB grafa v zvezi s časovno zahtevnost korakov BlockRank algoritma. LARGEWEB je graf spleta, ki so ga generirali leta 2001 v projektu *Stanford WebBase*. Pregledali so 290 milijonov spletnih strani in našli bilijon povezav med njimi, kar 6 GB prostora. Ker je veliko od teh spletnih strani izoliranih, ali pa niso našli povezav iz njih v tej preiskavi spleta, je dovolj pregledovati samo 70 milijonov vozlišč v glavnih korakih osnovnega algoritma PageRank, ostala vozlišča pa se upošteva pozneje. Vendar pa vsaka iteracija PageRanka na teh 70 milijon vozliščih na računalniku AMD Athlon 1533 MHz s 3.5 GB osnovnega spomina traja 7 minut. Ker pa PageRank ponavadi

Web Page	Local	Global
http://aa.stanford.edu	0.2196	0.4137
http://aa.stanford.edu/aeroastro/AAfolks.html	0.0910	0.0730
http://aa.stanford.edu/aeroastro/AssistantsAero.html	0.0105	0.0048
http://aa.stanford.edu/aeroastro/EngineerAero.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/Faculty.html	0.0459	0.0491
http://aa.stanford.edu/aeroastro/FellowsAero.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/GraduateGuide.html	0.1244	0.0875
http://aa.stanford.edu/aeroastro/Labs.html	0.0387	0.0454
http://aa.stanford.edu/aeroastro/Links.html	0.0926	0.0749
http://aa.stanford.edu/aeroastro/MSAero.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/News.html	0.0939	0.0744
http://aa.stanford.edu/aeroastro/PhdAero.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/aacourseinfo.html	0.0111	0.0039
http://aa.stanford.edu/aeroastro/aafaculty.html	0.0524	0.0275
http://aa.stanford.edu/aeroastro/aalabs.html	0.0524	0.0278
http://aa.stanford.edu/aeroastro/admitinfo.html	0.0110	0.0057
http://aa.stanford.edu/aeroastro/courseinfo.html	0.0812	0.0713
http://aa.stanford.edu/aeroastro/draftcourses.html	0.0012	0.0003
http://aa.stanford.edu/aeroastro/labs.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/prospective.html	0.0100	0.0063
http://aa.stanford.edu/aeroastro/resources.html	0.0112	0.0058
http://aa.stanford.edu/aeroastro/visitday.html	0.0123	0.0068

Slika 2.6: Primerjava lokalni in globalnih PageRank indeksov bloka

zahteva 100 iteracij iz tega vidimo, da resnično potrebujemo dobre aproksimacijske metode za velike grafe. V tabeli opazimo, da je razmerje časovnih zahtevnosti korakov 1:2:4 približno 2:1:8, korak 3 pa je zanemarljiv, saj gre samo za množenje vektorjev.

Step	Wallclock time
1	17m 11s
2	7m 40s
3	0m 4s
4	56m 24s
Total	**81m 19s**

Slika 2.7: Časovna zahtevnost posameznega koraka za LARGEWEB graf

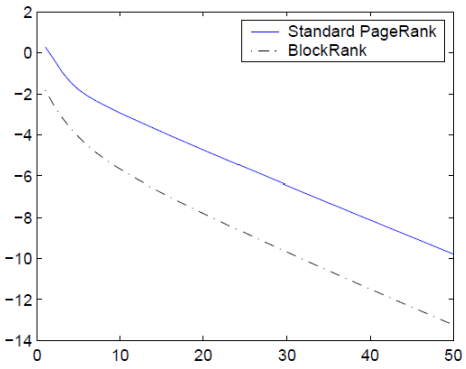
3. Na sliki 2.8 lahko vidimo še primerjavo časovnih zahtevnosti štirih različic algoritmov na grafu LARGEWEB. Prva varianta je običajen PageRank algoritem. Druga je PageRank algoritem, kjer spletne strani uredimo v bloke kot pri bločnem PageRank algoritmu, nato pa na tej ureditvi izvedemo običajen PageRank algoritem. Tretja je BlockRank algoritem predstavljen v tem razdelku. Četrta različica pa je BlockRank algoritem s paralelnim računanjem. Opazimo, da že v različici 2 pridobimo 1.5-kratno pospešitev algoritma. Različica 3 tako ni tako opazno izboljšava. Vendar pa hkrati opazimo, da je 4. različica še 1.5-kratna pospešitev 2. S paralelnim računanjem namreč koraki 1-3 v BlockRank algoritmu postanejo zanemarljivi in tako šteje samo korak 4. Korake 1-3 lahko namreč računamo vzporedno z izvajanjem koraka 0, tj. gradnjo prehodne matrike in samega grafa spleta. Tako se koraki 1-3 končajo skoraj hkrati s koncem koraka 0. Če torej izkoristimo vse možne pospešitve, ki jih omogoča BlockRank algoritem, na LARGEWEB-u skupno pridobimo 3-kratno pospešitev. Na drugih grafih je ta pospešitev lahko še bistveno večja.

4. O številu zahtevanih iteracij pa nam več pove slika 2.9. Graf prikazuje primerjavo števila potrebnih iteracij (x-os) za konvergenco običajnega PageRank algoritma in BlockRank algoritma do napake *log(napaka)* (y-os), pri čemer je napaka mišljena kot

Algorithm	Wallclock time
Standard	180m 36s
Standard (using url-sorted links)	87m 44s
BlockRank (no pipelining)	81m 19s
BlockRank (w/ pipelining)	57m 06s

Slika 2.8: Primerjava metod za LARGEWEB graf

razlika med zaporednima približkoma potenčne metode v BlockRank algoritmu. Tu je bil faktor α iz konstrukcije prehodne matrike (glej razdelek 1.7.1) 0.85. Vidimo, da je pridobitev približno 2-kratna. Izkaže se, da je za druge α pospešitev lahko tudi do 10-kratna (npr. $\alpha = 0.99$).



Slika 2.9: Primerjava metod po številu iteracij za Stanford/Berkley graf

2.4.3 Zaključek razdelka

Videli smo, da ima graf spleta gnezdeno bločno zgradbo. To je sicer ugotovitev, ki omogoča še veliko nadaljnega raziskovanja. To bločno zgradbo smo znali izkoristiti za prikrojitev PageRank algoritma v BlockRank algoritem. Slednji pa omogoča tudi 2- in večkratne časovne pospešitve računanja PageRank indeksa. Nadaljne pospešitve so možne v smeri ugotavljanja bolj natančne bločne strukture grafa spleta, ki tudi slabo konvergirajoče, dobro gnezdene bloke razbije na nove bloke. Še ena možna smer posploševanja pa je kombinacija PageRank metode s hibridnimi metodami za pospešitev njegove konvergence, npr. kombinacija kvadratne ekstrapolacije in Gauss-Seidlove metode.

Poglavje 3

Napredni koncepti centralnosti

Blažka Hunski, Barbara Ikica, Ana Špela Hodnik

V matematičnih delih se pojavlja zelo veliko različnih centralnih indeksov. Razvoj novih definicij je motiviran z nezmožnostjo že poznanih, da bi dobro pokazali centralnost vozlišč ali povezav v nekem novem, drugačnem primeru. V tem poglavju se bomo ukvarjali s podobnostjo, razlikami in povezavami med različnimi indeksi centralnosti. Cilj tega poglavja je narediti pregled nad temi povezavami in tako predstaviti nekakšno shemo obstoječih centralnih indeksov, zato se osredotočimo na formalne definicije, ki držijo za vsa omrežja. Običajno takšni pristopi ne upoštevajo posebne strukture omrežja, ki jo lahko poznamo v kakšnem primeru, vendar so primerni za raziskovanje abstraktnih definicij različnih centralnih indeksov.
Vemo, da je pomembno, katero definicijo centralnosti uporabiti v katerem primeru. To je še en razlog več za klasifikacijo centralnih indeksov, saj nam bi klasifikacija morda pomagala izbrati pravi indeks za določen primer. V splošnem to sicer ni mogoče, saj ne vemo, katero področje nas zanima in kako je zgrajeno to določeno omrežje. Vendar lahko sklepi, do katerih bomo prišli, služijo kot ideje, kako natančno, ali vsaj z dobrim približkom, modelirati dano situacijo.
V prvem delu bomo spoznali normalizacijo centralnih indeksov. Razlikovali bomo med pristopi, kjer primerjamo centralni indekse znotraj enega grafa in med primerjavo centralnih indeksov med različnimi grafi. Večina tehnik je tako splošnih, da bi jih lahko uporabili za vse centralne indekse, ki smo jih spoznali do sedaj.
Nato bomo proučevali možnost spreminjanja centralnega indeksa tako, da se bomo osredotočili na določeno podmnožico vozlišč. Ta množica je lahko npr. podmnožica spletnih strani, ki najbolj zanima izbranega uporabnika interneta. S takšno podmnožico lahko razporeditev strani personaliziramo glede na interese uporabnika. Ideja personalizacije je podrobneje opisana v drugem razdelku. Kot v primeru normalizacije, so tudi tukaj nekatere metode personalizacije primerne za vse do sedaj obravnavane centralne indekse, nekatere pa so zgrajene le za točno določen centralni indeks.
V tretjem razdelku bomo podali neformalen pristop k strukturiranju centralnih indeksov. Indekse bomo razdelili glede na različne komponente in tako dobili štiri kategorije centralnih indeksov. Tukaj bomo predstavili tudi shemo, ki lahko služi kot pomoč pri

konstruiranju novega centralnega indeksa.
S fundamentalnimi lastnostmi, ki jih mora imeti vsak centralni indeks, se bomo podrobneje ukvarjali v četrtem delu. Predlaganih in preučenih je kar nekaj takšnih lastnosti, ki nam dajo različne množice aksiomov za centralne indekse.
V zadnjem razdelku bomo ugotovili, kako centralni indeksi reagirajo na spremembe strukture omrežja. Tipični zgledi so npr. spletna omrežja, kjer se dodajanje strani in linkov nenehno dogaja in je vprašanje stabilnosti predmet velikega zanimanja. Predstavili bomo kar nekaj zgledov centralnih indeksov in njihov reakcij na takšne spremembe.

3.1 Normalizacija

Poznamo veliko različnih centralnostnih indeksov; tako za povezave kot za vozlišča grafov. Veliko izmed njih kot vrednost poda nenegativno realno število: sosedska centralnost nenegativno celo število, bližinska centralnost število z intervala $(0, 1]$... Vendar, ko se vprašamo, kaj nam pove centralni indeks 0.8 za neko vozlišče, ugotovimo, da brez poznavanja strukture grafa in števila vozlišč v grafu, to število pove prav malo. V tem razdelku se bomo ukvarjali z vprašanjem, ali obstajajo kakšni splošni koncepti normalizacije, ki bi omogočili primerjavo elementov znotraj grafa in med različnimi grafi. S tem sta se v večini ukvarjala Ruhnau[102] in Möller[106]. Ukvarjali se bomo z vozliščnimi centralnostmi, seveda pa bi lahko ugotovljeno posplošili tudi na povezave.

3.1.1 Primerjava centralnosti elementov znotraj grafa

Naj bo $G = (V, E)$ graf z n vozlišči. Zanima nas, kako lahko med vozlišči grafa primerjamo centralne vrednosti, ki so s pomočjo (različnih) centralnih indeksov pripisane vsakemu vozlišču. Da bo stvar čim bolj enostavna, bomo centralne vrednosti vozlišč zapisali v vektor: za vsako centralnost c_X, kjer je X centralnost, ki jo opazujemo (D-sosedska, C-bližinska,...), bomo definirali vektor $\boldsymbol{c_X}$, kjer je $\boldsymbol{c_X}_i = c_X(i)$ za vsako vozlišče i grafa G.
Vsak centralni vektor $\boldsymbol{c_X}$ lahko normaliziramo tako, da ga delimo s p-normo tega vektorja, kjer je p-norma definirana takole:

$$\|\boldsymbol{c_X}\|_p = \begin{cases} (\sum_{i=1}^{n} |\boldsymbol{c_X}_i|^p)^{1/p}, & 1 \leq p < \infty, \\ \max_{i=1,\ldots,n} \{|\boldsymbol{c_X}_i|\}, & p = \infty. \end{cases}$$

Tako za komponente normaliziranega vektorja velja $\boldsymbol{c_X}_i \leq 1$.
Glavna razlika med p- normo, kjer je $p < \infty$ in ∞-normo je ta, da je pri normalizaciji z ∞ normo vrednost 1 vedno dosežena vsaj v eni komponenti normaliziranega vektorja, kar pri $p < \infty$ ni nujno. Tako nam normalizacija z ∞-normo poda relativne centralne vrednosti za vsako vozlišče v grafu. Če vektor normaliziramo s pomočjo 1-norme, vsaki komponenti vektorja pripišemo procent centralnosti, ki jo ima vozlišče v grafu.

Pri centralnostih, ki kot rezultat lahko podajo tudi negativne vrednosti, komponente normaliziranega vektorja s p-normo za $p < \infty$ podamo malce drugače:

$$\boldsymbol{c'_X}_i = \begin{cases} \boldsymbol{c_X}_i / (\sum_{j:\boldsymbol{c_X}_j > 0} |\boldsymbol{c_X}_j|^p)^{1/p}, & \boldsymbol{c_X}_i > 0, \\ 0, & \boldsymbol{c_X}_i = 0, \\ \boldsymbol{c_X}_i / (\sum_{j:\boldsymbol{c_X}_j < 0} |\boldsymbol{c_X}_j|^p)^{1/p}, & \boldsymbol{c_X}_i < 0. \end{cases}$$

Vendar niti p-norma za $p < \infty$ niti ∞-norma nista primerni za primerjavo centralnih vrednosti med različnimi grafi. Pri ∞-normi je namreč vrednost 1 dosežena v vsakem grafu, ne glede na njegovo strukturo, pa tudi $p < \infty$ marsikdaj ne vrne rezultata, primernega za primerjavo.

3.1.2 Primerjava centralnosti elementov med različnimi grafi

Kadar primerjamo centralnosti vozlišč različnih grafov, nam jo lahko kar pošteno zagodejo različne velikosti grafov. Zato se moramo normalizacije lotiti drugače.
Naj bo $\mathcal{G}_n$ množica povezanih grafov $G = (V, E)$ z n vozlišči. Centralni vektor normaliziramo tako, da vsaki komponenti priredimo vrednost (točkovno centralnost)

$$\boldsymbol{c''_{X}}_i = \frac{\boldsymbol{c_X}_i}{\boldsymbol{c^*_X}},$$

kjer je $\boldsymbol{c^*_X} = \max_{G \in \mathcal{G}_n} \max_{i \in V(G)} \boldsymbol{c_X}_i$ maksimalna centralna vrednost, ki jo lahko doseže katerokoli vozlišče v množici grafov $\mathcal{G}_n$.
Pri normalizaciji s pomočjo točkovne centralnosti je največja vrednost 1 vedno dosežena v vsaj eni komponenti centralnega vektorja nekega grafa z n vozlišči iz dane množice. Torej je primerjava centralnosti v različnih grafih mogoča.
Največje možne vrednosti točkovne centralnosti, ki jih lahko dosežejo vozlišča v grafu z n vozlišči so:

- Sosedska centralnost
$$\boldsymbol{c^*_D} \leq n - 1$$
- Centralnost posrednika na najkrajši poti
$$\boldsymbol{c^*_B} \leq \frac{n^2 - 3n + 2}{2}$$
- Bližinska centralnost
$$\boldsymbol{c^*_C} \leq (n - 1)^{-1}$$
- Ekscentrična centralnost
$$\boldsymbol{c^*_E} \leq 1$$

Na žalost je to teoretično zelo lepo, v praksi pa se za veliko število velikih grafov izkaže za ne tako enostavno, saj je izračun $\boldsymbol{c^*_X}$ v splošnem netrivialen ali celo nemogoč za nekatere centralnosti. Primer za to je Katzev indeks stanja, ki spada med povratne centralnosti in meri ne le direkten, temveč tudi medsebojni posreden vpliv vozlišč, zato uvedemo faktor dušenja. Centralni indeks i-tega vozlišča se izračuna kot

$$c_K(i) = \sum_{k=1}^{\infty} \sum_{j=1}^{n} \alpha^k (A^k)_{ji},$$

če vsota konvergira. A v zgornji formuli je matrika sosednosti usmerjenega enostavnega grafa G z n vozlišči, $\alpha \in [0, 1]$ pa t. i. faktor dušenja.

Konvergenca vsote je pri izbranem α močno odvisna od največje lastne vrednosti matrike A, zato se lahko zgodi, da pri izbranem α za nekatere grafe lahko izračunamo Katzeve indekse stanja njihovih vozlišč, za druge pa ne. Poglejmo primer:

$$A_1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ in } A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Ker je ${A_1}^k$ ničelna matrika za vsak $k \geq 2$, je konvergenca vsote zagotovljena za poljuben $\alpha \in [0, 1]$. Če izberemo največjo možno vrednost $\alpha = 1$, potem neskončna vsota $\sum_{k=1}^{\infty} \alpha^k {A_2}^k$ ne konvergira, saj je vsota enaka $\lim_{K\to\infty} \sum_{k=1}^{K} \mathbf{1}_2 {\mathbf{1}_2}^T$. Zgled pokaže, da ni čisto jasno, kateri α izbrati za določitev vrednosti $\boldsymbol{c^*_K}$.

3.2 Personalizacija

Motivacija: predstavljajmo si, da bi lahko svoj brskalnik priredili tako, da bi odpiral spletne strani v skladu z našimi interesi in hotenji. Tako bi vsak uporabnik pri vsakem iskanju dobil kar najbolj ustrezne strani.
Imamo dva pristopa, kako priti do tega:

- prvi je spremeniti uteži na vozliščih (straneh) ali na povezavah (linkih) grafa spleta s pomočjo personalizacijskega vektorja $\boldsymbol{v}$. Uteži na vozliščih lahko opisujejo npr. čas, ki ga dnevno porabimo na tisti spletni strani, utež na povezavi pa lahko opisuje verjetnost, da bo link uporabljen;
- drugi pristop je izbrati korensko množico $R \subseteq V$ vozlišč in izmeriti pomembnost preostalih vozlišč glede na to množico.

Kasneje bomo videli, da lahko ta dva pristopa uporabimo kot dva operatorja $P_{\boldsymbol{v}}$ in P_R.

3.2.1 Personalizacija centralnosti, ki temeljijo na razdalji in najkrajših poteh

V tem razdelku se ne bomo ukvarjali s centralnostmi, ki določijo centralnostnih indeksov elementov glede na vsa vozlišča v grafu, ampak s tistimi, ki določijo centralnosti vozlišč glede na neko določeno korensko množico R. Množica R je določena tako, da se v njej nahajajo elementi, ki veljajo za pomembne. Sedaj se vprašamo, kako naj določimo centralnost ostalih vozlišč glede na vozlišča iz množice R. Pogledali si bomo pristop, ki sta ga uporabila White in Smyth [107].
Naj bo $c(v)$ nek centralični indeks vozlišča v. Potem $c(v|R)$ določa relativno pomembnost vozlišča v glede na izbrano korensko množico R. Naj $P(s, t)$ označuje katerokoli dobro definirano množico poti med vozliščema s in t. Avtorja predlagata tri različne množice poti:

- množica najkrajših poti;
- množica k-najkrajših poti, ki je definirana kot množica vseh poti z dolžino $\leq k$;
- množica k-najkrajših disjunktnih poti.

Množico najkrajših poti uporabljamo npr. pri centralnosti posrednika na najkrajši poti. Relativno centralnost posrednika $c_{RBC}(v)$ lahko definiramo na tri načine.
V prvem primeru je vozlišče v pomembno, če v deležu najkrajših poti od vozlišča $r \in R$ do vozlišča $t \in V$ nastopa v.
To t. i. *izvorno relativno centralnost posrednika* definiramo kot

$$c_{sRBC}(v) = \sum_{r \in R} \sum_{t \in V} \delta_{rt}(v).$$

Kadar je element v pomemben, če je vsebovan v velikem deležu najkrajših poti, ki se končajo v vozlišču $r \in R$, govorimo o ciljni relativni centralnosti posrednika, in jo izračunamo kot

$$c_{tRBC}(v) = \sum_{s \in V} \sum_{r \in R} \delta_{sr}(v).$$

Tretji način nastopi, kadar je element pomemben, če je vsebovan v velikem deležu najkrajših poti, ki vodijo iz množice R v množico R. To centralnost označimo in izračunamo kot

$$c_{RBC}(v) = \sum_{r_s \in R} \sum_{r_t \in R} \delta_{r_s r_t}(v).$$

Če izberemo katerokoli drugo množico poti $P(s,t)$, npr., če izberemo množico k-najkrajših poti, potem moramo namesto $\delta_{st}(v)$ izračunati

$$\delta_{st|P}(v) = \frac{\sigma_{st}(v)}{|P(s,t)|},$$

kjer je $\sigma_{st}(v)$ število poti $p \in P(s,t)$, ki vsebujejo vozlišče v.
Ta zgled predstavlja idejo, ki se skriva za takšno vrsto personalizacije. Seveda lahko to enostavno razširimo na vse centralnosti, ki temeljijo na razdalji.

3.2.2 Personalizacija spletnih centralnosti

Zopet si oglejmo model naključnega uporabnika interneta za spletne centralnosti, ki smo ga že spoznali v tretjem poglavju in predpostavimo, da je prispel na spletno stran, od koder ni izhodne povezave oziroma so le te nerelevantne. Logična domneva v tem primeru je, da bo skočil na naključno spletno stran, zato imajo vse mogoče spletne strani enako verjetnost. Seveda je jasno, da predpostavka iste verjetnosti ni ravno realistična: nekateri uporabniki interneta imajo raje strani o športu in bodo verjetno odšli na spletno stran iz tega področja, če se bodo ujeli v slepi ulici, spet drugi bodo odprli strani o novicah, itd. Vprašanje, ki se nam postavi je, kako modelirati toliko različnih uporabnikov interneta.
Oglejmo si enačbo za PageRank centralni indeks, ki smo ga spoznali v 3. poglavju:

$$\boldsymbol{c}_{PR}^{k} = dP\boldsymbol{c}_{PR}^{k-1} + (1-d)\mathbf{1}_n,$$

kjer je $\boldsymbol{c}_{PR}(q)$ PageRank indeks strani q, d fakor dušenja in $P = D^{+}A$, kjer je D^{+} diagonalna matrika z izhodno stopnjo $i-$tega vozlišča na $i-$ti diagonali, ko i preteče vsa vozlišča. A je matrika sosednosti. Zelo intuitiven pristop bi bil zamenjava vektorja enic $\mathbf{1}_n$ s *personalizacijskim vektorjem* $\boldsymbol{v}$, ki zadošča $v_i > 0$ in $\sum_i v_i = 1$. White in Smyth [107] sta predlagala, da bi normirali vrednosti v vozliščih glede na jedrno množico R:

$$v_i = \begin{cases} \frac{1-\varepsilon}{|R|}, & i \in R \\ \frac{\varepsilon}{n-|R|}, & i \notin R \end{cases},$$

kjer je $0 < \varepsilon << 1$. Predlagala sta tudi zelo podoben pristop, kot ga imamo pri algoritmu Hubs&Authorities. Namesto, da bi v algoritmu

$$\begin{aligned} \boldsymbol{c}^0_{HA-A} &= \mathbf{1}_n \\ \text{for } k &= 1 \dots \text{do} \\ \boldsymbol{c}^k_{HA-H} &= A_\sigma \boldsymbol{c}^{k-1}_{HA-H} \\ \boldsymbol{c}^k_{HA-A} &= A^T_\sigma \boldsymbol{c}^k_{HA-H} \\ \boldsymbol{c}^k_{HA-H} &= \frac{\boldsymbol{c}^k_{HA-H}}{\left\|\boldsymbol{c}^k_{HA-H}\right\|} \\ \boldsymbol{c}^k_{HA-A} &= \frac{\boldsymbol{c}^k_{HA-A}}{\left\|\boldsymbol{c}^k_{HA-A}\right\|}, \end{aligned}$$

uporabili iterativen proces, sta v vsakem koraku dodala personalizacijski vektor in tako dobila

$$\begin{aligned} \boldsymbol{c}^k_{HA-H} &= dA_\sigma \boldsymbol{c}^{k-1}_{HA-H} + (1-d)\boldsymbol{v} \\ \boldsymbol{c}^k_{HA-A} &= dA^T_\sigma \boldsymbol{c}^k_{HA-H} + (1-d)\boldsymbol{v} \\ \boldsymbol{c}^k_{HA-H} &= \frac{\boldsymbol{c}^k_{HA-H}}{\left\|\boldsymbol{c}^k_{HA-H}\right\|} \\ \boldsymbol{c}^k_{HA-A} &= \frac{\boldsymbol{c}^k_{HA-A}}{\left\|\boldsymbol{c}^k_{HA-A}\right\|}, \end{aligned}$$

kjer je $d \in [0,1]$ izbran tako, da uravnava vpliv $\boldsymbol{v}$.
Dokler so komponente vektorja $\boldsymbol{v}$ pozitivne in je $\boldsymbol{v}$ stohastičen vektor (vsota vseh komponent je 1), je pripadajoča Markovska veriga še vedno nerazcepna, saj se konvergence PageRank algoritma nismo dotikali.
Vendar ima ta pristop veliko slabost: kot že vemo, je izračun PageRank vektorjev zelo časovno potraten in ni, vsaj trenutno, nobene možnosti za izračun PageRank centralnosti za veliko različnih uporabnikov spleta. Vendar se obetajo novi pristopi za izračun personaliziranih PageRank vektorjev v sprejemljivem času. S tem namenom poglejmo splošen pristop k personalizaciji za PageRank, ki ga je vpeljal Taher H. Haveliwala [108]. Kot smo že povedali, personaliziran PageRank vektor dobimo kot rešitev enačbe

$$\boldsymbol{c}_{PR} = dP^T \boldsymbol{c}_{PR} + (1-d)\boldsymbol{v}.$$

Ker je $(I - dP^T)$ strogo diagonalno dominantna matrika, je obrnljiva, in zato

$$\boldsymbol{c}^{\boldsymbol{v}}_{PR} := \boldsymbol{c}_{PR} = (1-d)(I - dP^T)^{-1}\boldsymbol{v} =: Q\boldsymbol{v}.$$

($\boldsymbol{c}^{\boldsymbol{v}}_{PR}$ pišemo zato, da poudarimo odvisnost $\boldsymbol{c}_{PR}$ od $\boldsymbol{v}$.)
Če za $\boldsymbol{v}$ izberemo $i-$ti enotski vektor $\boldsymbol{v} = \boldsymbol{e}^i$, potem je $\boldsymbol{c}^{\boldsymbol{e}^i} = Q_{\cdot j}$ zato je množica stolpcev matrike Q baza personaliziranih vektorjev PageRank indeksov. Težava, ki se pojavi, je, da moramo za določitev matrike Q izračunati inverz matrike $I - dP^T$, kar je zelo zamudno, če so matrike velike. Tako Q raje aproksimiramo z $\hat{Q} \in \mathbb{R}^{n \times K}$.Za izračun približnega centralnega indeksa tako upoštevamo konveksno kombinacijo le K baznih vektorjev (linearno neodvisnih stolpcev Q):

$$\boldsymbol{c}^{\boldsymbol{w}}_{PR} = \hat{Q}\boldsymbol{w},$$

kjer je $\boldsymbol{w} \in \mathbb{R}^K$ stohastični vektor, $w_i > 0 \ \forall i$.
Haveliwala[108] in ostali so pokazali, da lahko sledeče tri pristope združimo v splošni opis, ki smo ga navedli zgoraj, saj se ti pristopi med sabo razlikujejo le v tem, kako naredimo aproksimacijo.

PageRank občutljiv na tematiko

Haveliwala[109] predlaga, naj se ravnamo po sestavljenem offline-online algoritmu, kjer prva faza (offline) sestoji iz naslednjih dveh korakov:

1. Izberemo K najbolj pomembnih tematik $t_1, t_2, \ldots, t_K$ in definiramo v_i^k kot (normalizirano) stopnjo povezanosti strani i s tematiko t_k, $i = 1, 2, \ldots, n$, $k = 1, 2, \ldots, K$.

2. Izračunamo $\hat{Q}_{\cdot k} = \boldsymbol{c}_{PR}^{\boldsymbol{v}^k}$, $k = 1, 2, \ldots, K$.

Druga faza (online) izgleda takole:

1. Za iskalno zahtevo σ izračunamo (normalizirane) tematske uteži $w_1^\sigma, w_2^\sigma, \ldots, w_K^\sigma$.

2. Sestavimo stolpce $\hat{Q}$ tako, da upoštevamo uteži, in dobimo

$$\boldsymbol{c}_{PR}^\sigma = \sum_{k=1}^{K} w_k^\sigma \hat{Q}_{\cdot k}.$$

Pri tem postopku moramo paziti, da je K dovolj majhen (npr. K=16) in da je nabor tematik dovolj širok.

Modularen PageRank

Drugi pristop sta predlagala Jennifer Widom in Glen Jeh[110]. Njun algoritem je sestavljen iz offline in online koraka.
V offline koraku izberemo K strani $i_1, i_2, \ldots, i_K$ z visokim rangom. Te visoko razvrščene strani tvorijo množico središč.
S pomočjo personalizacijskih vektorjev $\boldsymbol{e}^{i_k}$ izračunamo pripadajoče *bazne vektorje* oz. *središčne vektorje* $\boldsymbol{c}_{PR}^{\boldsymbol{e}^{i_k}}$. Za vsak personalizacijski vektor $\boldsymbol{v}$, ki je konveksna kombinacija $\boldsymbol{e}^{i_1}, \boldsymbol{e}^{i_2}, \ldots, \boldsymbol{e}^{i_K}$, lahko izračunamo pripadajoč PageRank vektor, ki je konveksna kombinacija središčnih vektorjev. Vendar z večanjem K ni več mogoče niti izračunati vseh središčnih vektorjev vnaprej niti jih učinkovito shraniti. Da bi premagala to oviro, sta Jeh in Widom predlagala uporabo *parcialnih vektorjev* in *središčnega skeleta*.Pokazala sta, da je mogoče učinkovito izračunati in shraniti vse parcialne vektorje, ki so skupaj s središčnim skeletom zadostni za izračun vseh središčnih vektorjev in posledično končnega personaliziranega PageRanka. Ideja je redukcija računanja na središčno množico, ki je veliko manjša kot graf spleta. Seveda velja, da z večjim K dobimo boljšo reprezentacijo matrike Q.
Online korak je sestavljen iz določanja personalizacijskega vektorja

$$\boldsymbol{v}^\sigma = \sum_{k=1}^{K} \alpha_k^\sigma \boldsymbol{e}^{i_k}$$

in pripadajočega PageRank vektorja

$$\boldsymbol{c}_{PR}^{\sigma} \sum_{k=1}^{K} \boldsymbol{c}_{PR}^{\alpha_k^{\sigma} \boldsymbol{e}^{i_k}}.$$

Tudi tu si pomagamo s parcialnimi vektorji in središčnim skeletom.

BlockRank

Ta pristop je Sepandar D. Kamvar [111] z ostalimi iznašel, da bi pospešil računanje PageRanka (4. poglavje). Sestoji iz 3-stopenjskega algoritma, kjer je glavna ideja razgraditi graf spleta glede na domene. To metodo lahko uporabimo tudi za iskanje personaliziranih PageRank vrednosti. Algoritem se glasi:

1. (offline) izberemo K blokov (domen) in definiramo v_i^k kot lokalni PageRank strani i in bloka k, $i = 1, 2, \ldots, n$, $k = 1, 2, \ldots, K$. Izračunamo $\hat{Q}_{\cdot k} = \boldsymbol{c}_{PR}^{\boldsymbol{v}^k}$ (avtorji trdijo, da je mogoče izračunati za $K \geq 10^3$, če izkoristimo strukturo spleta);

2. (online) za iskalni niz σ poiščemo ustrezne uteži domen, da uredimo domene;

3. (online) uporabimo standardni PageRank algoritem za izračun pripadajoče centralnosti. Kot vhod uporabimo lokalne PageRank vrednosti, izračunane v prvem koraku in utežene z utežmi domen iz drugega koraka.

Tako koncept personalizacije iz tega dela kot tudi koncept normalizacije iz prejšnjega razdelka bomo uporabili za definiranje štirih dimenzij centralnih indeksov, ki bodo predstavljene v naslednjem poglavju.

3.3 Štiri dimenzije centralnega indeksa

V tem poglavju bomo predstavili pogled na centralne indekse z vidika štirih dimenzij. Gre za poskus strukturiranja obsežnega področja mer centralnosti in metod za normalizacijo in personalizacijo le-teh.

Sama ideja, ki stoji za tem pristopom, izvira iz spoznanja, da trenutno še ni konsistentne aksiomatske sheme, ki bi zaobjela vse centralne mere, ki smo jih obravnavali doslej. Pomembno pa je poudariti, da ta prispevek k strukturiranju, ki ga bomo spoznali, ne predstavlja niti formalnega niti celostnega pristopa k tej tematiki. Toda ne glede na to lahko služi kot zelo uporabno orodje v praksi.

Analiza centralnih mer je vodila k ideji o delitvi centralnosti v štiri kategorije na podlagi njihovega osnovnega modela izračuna. Vsak model izračuna je predstavljen s t.i. osnovnim izrazom. S tem, ko določimo osnovni izraz, lahko na modelu izračuna uporabimo še kakšen operator izraza (npr. vsoto ali maksimum) in številne metode, namenjene personalizaciji in normalizaciji. V sledečih vrsticah bomo podrobneje razdelali to idejo in na koncu poglavja predstavili še shemo, ki nazorno prikaže sam proces strukturiranja z vidika štirih dimenzij. Z njo si lahko pomagamo, če želimo klasificirati nove centralnostne mere ali pa morda že obstoječe prilagoditi svojim potrebam.

3.3.1 Osnovni izraz

Prvo dimenzijo našega pristopa tvorita klasifikacija centralnih indeksov v štiri kategorije in reprezentacija vsake izmed kategorij z osnovnim izrazom. Pri tem znova opomnimo, da je ta klasifikacija le predlog strukturirajna, ki se je porodil z analizo že obstoječih mer centralnosti, ki so bile obravnavane v prejšnjih poglavjih. Kategorije pa so sledeče:

Dosegljivost

Prva kategorija centralnih mer temelji na pojmu dosegljivosti. Pri tem vozlišče smatramo kot centralno, če doseže veliko drugih vozlišč.

Zgledi centralnih indeksov, ki sodijo v to kategorijo, so sosedska centralnost c_D, ekscentrična centralnost c_E, bližinska centralnost c_C in bližinska centralnost v naključnem sprehodu (centralnost Markova) c_M.

Vse naštete centralnosti temeljijo na konceptu razdalje $d(u, v)$ med dvema vozliščema u in v. Npr. pri merjenju sosedske centralnosti vozlišča štejemo vozlišča, ki so dosegljiva na razdalji 1. Bližino vozlišča u merimo z obratno vrednostjo vsote po razdaljah do vseh drugih vozlišč v. Največjo bližinsko centralnost bo torej imelo tisto vozlišče, ki ima najmanjšo skupno razdaljo do drugih vozlišč. Pri centralnosti, ki temelji na ekscentričnosti, postopamo podobno, le da namesto skupnih razdalj opazujemo maksimalne razdalje. V primeru bližinske centralnosti v naključnem sprehodu pa ekvivalentno, kot smo v prvih treh primerih obravnavali pojem razdalje, tu obravnavamo povprečni čas prvega prehoda od vozlišča u do vseh drugih vozlišč v v naključnem sprehodu - torej povprečni čas, ki je potreben, da iz vozlišča u obiščemo vsa druga vozlišča po naključnih sprehodih.

Količina pretoka

Druga kategorija centralnih indeksov bazira na količini pretoka $f_{st}(x)$ od vozlišča s do vozlišča t, ki gre skozi element x, ki je lahko bodisi vozlišče bodisi povezava. Temu primerno vzamemo za osnovni izraz te kategorije kar količino pretoka $f_{st}(x)$.

V to kategorijo sodijo npr. centralnosti, ki obravnavajo pretoke, ki se v principu obnašajo podobno, kot se pretaka električni tok po električnem omrežju, in naključne sprehode. Tudi mere centralnosti, ki temeljijo na štetju najkrajših poti, sodijo sem. Takšna mera je med drugimi obremenitvena centralnost, ki jo lahko interpretiramo kot mero količine toka, ki gre skozi element x (ki je vozlišče ali povezava), če vsako vozlišče grafa s pošlje vsem drugim vozliščem t po eno enoto toka po vsaki najkrajši poti, ki ju povezuje. Tako ima največji indeks obremenitvene centralnosti tisto vozlišče, skozi katerega poteka največ najkrajših poti. Podobno tudi centralnost posrednika na najkrajši poti sodi v to kategorijo, saj meri pričakovani delež pretoka skozi vozlišče v, če iz vsakega vozlišča s pošljemo po eno enoto toka vsem drugim vozliščem t zaporedoma, pri čemer vsakič pošljemo tok po natanko eni naključno in neodvisno izbrani najkrajši poti, ki povezuje s in t.

Vitalnost

Tretja kategorija centralnih indeksov bazira na definiciji vitalnosti. Centralna vrednost elementa x je definirana kot razlika med vrednostjo neke realne funkcije f na G in vrednostjo f na G brez tega elementa. Pri tem je lahko x vozlišče ali povezava.

Spomnimo se, da smo splošno mero vitalnosti označili kot $\nu(G, x) = f(G) - f(G \backslash \{x\})$. V to kategorijo tako sodi npr. vitalnost posrednika maksimalnega pretoka.

Povratnost

Osnova zadnje kategorije centralnih indeksov je implicitna definicija centralnosti. Centralne indekse iz te kategorije lahko v splošnem predstavimo s formulo

$$c(v_i) = f(c(v_1), c(v_2), \ldots, c(v_n)).$$

Takšna formula nam ne pove nič drugega kot to, da je centralna vrednost vozlišča v_i odvisna od centralnih vrednosti vseh vozlišč $v_1, v_2, \ldots, v_n$.

3.3.2 Operator izraza

Drugo dimenzijo centralnega indeksa predstavlja operator izraza. Osredotočimo se na prve tri kategorije, torej na dosegljivost, količino pretoka in vitalnost. Opazili smo že, da lahko pogosto na osnovnem izrazu uporabimo ustrezno množico operatorjev in dobimo smiselne centralne mere.

Predstavimo to na preprostem zgledu. Če smo za nek primer določili, na kakšen način merimo razdaljo, ki nas zanima, lahko z operatorjem natančneje definiramo, kaj točno naj meri centralni indeks. Centralni indeks lahko tako definiramo kot maksimum po vseh razdaljah od vozlišča u do vseh drugih vozlišč v (kot pri ekscentričnosti) ali kot vsoto po vseh razdaljah (kot v bližinski centralnosti) ali pa kot povprečno razdaljo do vseh drugih vozlišč (kot v normalizirani bližinski centralnosti).

V nekaterih primerih lahko dobimo centralni indeks s popolnoma smiselnim pomenom tudi z uporabo nekoliko bolj posebnih operatorjev, kot je npr. varianca vseh razdalj.

Tako vidimo, da je za centralnosti iz prvih treh kategorij povsem smiselno ločiti izbiro operatorja izraza od izbire osnovnega izraza.

3.3.3 Personalizacija

Tretja dimenzija je dana z metodami, ki pomagajo personalizirati indekse centralnosti. Spoznali smo dva načina, kako se lahko lotimo personalizacije. Prvi način, ki ga označimo s P_v, je mogoče uporabiti na vseh centralnih merah, ki znajo delati z uteženimi povezavami ali vozlišči. Ta personalizacija priredi utežni vektor v množici vozlišč V, množici povezav E ali pa prehodni matriki P, ki je bila predstavljena v modelu slučajnega spletnega uporabnika v poglavju o spletnih centralnostih.

Druga metoda personalizacije, označena s P_R, pa upošteva vnaprej izbrano množico vozlišč, t.i. množico korenov R. Pri določanju centralnosti je tako upoštevana ta množica. Takšen pristop lahko uporabimo na vseh centralnih indeksih, ki temeljijo na razdalji.

Oba načina personalizacije in še številni drugi, ki se jih nismo dotaknili, gradijo tretjo dimenzijo centralnega indeksa.

3.3.4 Normalizacija

Vse centralne indekse, ki smo jih opisali do sedaj, je mogoče normalizirati. Metode, s katerimi lahko normaliziramo centralni indeks, tvorijo četrto dimenzijo. Veliko večino indeksov je mogoče normalizirati tako, da vsako centralno vrednost delimo z maksimalno

centralno vrednostjo. Ker smo normalizacijo že natančneje obravnavali, podrobnosti izpustimo.

3.3.5 Neodvisnost dimenzij

Vse naštete dimenzije, torej osnovni izraz, operator izraza, personalizacija in normalizacija so neodvisne druga od druge. Centralne indekse, ki smo jih do sedaj spoznali, lahko smiselno razčlenimo glede na te dimenzije. Pri tem pa ne trdimo, da je mogoče vse obstoječe centralne indekse smiselno razčleniti na ta način. Ker poleg tega ne obstaja striktna definicija centralnega indeksa, ne moremo zagotoviti, da bo vsaka možna kombinacija v tako definiranih dimenzijah porodila smiselen centralni indeks. Naš cilj je ustvariti model, ki nam bo pomagal predstaviti strukturo konstrukcije ustreznega centralnega indeksa glede na ta štiri-dimenzionalni pristop.

3.3.6 Konstrukcija centralnega indeksa

Slika 3.1: Diagram za enostavnejše izbiranje, prilagajanje ali ustvarjanje centralnega indeksa, ustreznega obravnavanemu problemu.

Zgornji diagram prikazuje pristop, ki demonstrira, kako lahko poiščemo ali adaptiramo kakšno že obstoječo centralnost, ustrezno problemu, ki ga obravnavamo.

Prvi korak pri izbiri ustreznega centralnega indeksa je, da se vprašamo, na katero vprašanje nam mora centralni indeks odgovoriti. S tem, ko postavimo vprašanje, določimo kategorijo, ki nas zanima, in njej pripadajoči osnovni izraz. Kakorkoli, v splošnem osnovni izraz zaobjame le abstraktni koncept, ker lahko npr. razdaljo med dvema vozliščema izmerimo kot povprečni čas prvega prehoda v naključnem sprehodu ali pa klasično izračunamo dolžino najkrajše vmesne poti. Tako moramo določiti konkreten model izračuna za izbrani osnovni izraz.

Ko to storimo, lahko uporabimo prvo personalizacijo. S tem dobimo personaliziran graf z modificiranimi ali dodanimi utežmi na vozliščih ali pa na povezavah. Če pri tem osnovni izraz pripada eni izmed prvih treh kategorij, torej dostopnosti, količini pretoka ali vitalnosti, lahko na tem mestu uporabimo drugo personalizacijo tako, da določimo množico korenov, ki jo je treba upoštevati pri merjenju centralnosti vozlišč.

Če na novo dobljeni izraz ustreza eni izmed kategorij dostopnosti, količini pretoka ali vitalnosti, moramo na tem mestu izbrati operator izraza. Le-tega bomo uporabili na izrazu, pri čemer bomo upoštevali spremenjeni graf, dobljen s personalizacijo. Primera operatorjev izraza sta operator maksimizacije in vsote po vseh izrazih.

Če smo za centralni indeks izbrali centralnost iz kategorije povratnosti, ne moremo vedno personalizirati z določanjem množice korenov. Zato diagram ubere malce drugačno

pot za indekse iz kategorije povratnosti. Za personalizacijo z utežnim vektorjem ubere korak, v katerem določi ustrezen sistem linearnih enačb in ga reši.

Za vse štiri kategorije lahko v zadnjem koraku dobljene centralne vrednosti normaliziramo. Običajno normaliziramo tako, da množimo s skalarjem. S tem dobimo končno obliko centraliziranega indeksa.

Tako je pristop s štirimi dimenzijami sicer nekoliko alternativno, toda zelo praktično in fleksibilno orodje za opisovanje, strukturiranje in konstruiranje centralnih mer. V naslednjem poglavju pa bomo predstavili nekaj bolj klasičnih pristopov, ki jih lahko prav tako uporabimo pri karakterizaciji centralnih indeksov.

3.4 Aksiomatizacija

V tem poglavju bomo govorili o vprašanju ali obstajajo neke splošne lastnosti, katerim naj bi zadoščala centralnost. Najprej bomo obravnavali dve aksiomatizaciji centralnega indeksa, ki temelji na razdalji, v drugem delu pa dve aksiomatizaciji povratne centralnosti.

3.4.1 Aksiomatizacija vozliščne centralnosti glede na razdaljo

Sabidussi je v [92] definiral nekaj aksiomov za vozliščno centralnost neusmerjenega grafa $G = (V(G), E(G))$. Mi si bomo pogledali približek njegovih definicij. Najprej definiramo operaciji na grafih:

- Dodajanje povezave (u, v):
  Naj bosta u in v dva različna vozlišča grafa G, kjer velja $(u, v) \notin E(G)$. Graf $H = (V(G), E(G) \cup \{u, v\})$ dobimo iz grafa G, tako da dodamo povezavo (u, v).

- Premik povezave (u, v):
  Naj bodo u, v in w različna vozlišča v grafu G taka, da $(u, v) \in E(G)$ in $(u, w) \notin E(G)$. Potem graf $H = (V(G), (E(G) \setminus \{(u, v)\}) \cup \{u, w\})$ dobimo s premikom povezave (u, v) v povezavo (u, w). Ko premaknemo povezavo, mora graf ostati povezan.

Naj bo $\mathcal{G}_n$ razred neusmerjenih povezanih grafov z n vozlišči. Naj bo $c : V(G) \to \mathbb{R}_0^+$ funkcija definirana na množici vozlišč grafa $G = (V(G), E(G)) \in \mathcal{G}_n$, ki vsakemu vozlišču grafa priredi nenegatovno realno število. Vpeljemo množico vozlišč grafa G z maksimalno centralnostjo glede na vozliščno centralnost c, označimo jo $\mathcal{S}_c(G) = \{u \in V(G) : \forall v \in V(G) c(u) \geq c(v)\}$.

Definicija 3.1 (Sabadussi [91]) Funkcijo c imenujemo *vozliščna centralnost* na $G \in \mathcal{G}'_n \subseteq \mathcal{G}_n$ ter $\mathcal{G}'_n$ imenujemo *c-dopustna* natanko tedaj ko veljajo naslednji pogoji:

1. $\mathcal{G}'_n$ je zaprta za izomorfizme, t.j., če je $G \in \mathcal{G}'_n$ in je graf H izomorfen grafu G potem je tudi $H \in \mathcal{G}'_n$.

2. Če je $G = (V(G), E(G)) \in \mathcal{G}'_n$, $u \in V(G)$ in H dobimo, tako da prestavimo ali dodamo povezavo v grafu G do vozlišča u, potem velja $H \in \mathcal{G}'_n$, t.j., $\mathcal{G}'_n$ je zaprta za prestavljanje in dodajanje povezav.

3. Če velja $G \cong_\phi H$, potem velja $c_G(u) = c_H(\phi(u))$ za vsak $u \in V(G)$.

4. Naj bo $u \in V(G)$ ter H dobimo iz grafa G, tako da dodamo povezavo do vozlišča u, potem velja $c_G(u) < c_H(u)$ in $c_G(v) \leq c_H(v)$ za vsak $v \in V(G)$.

5. Naj bo $u \in \mathcal{S}_c(G)$ in H tak graf, ki ga dobimo iz G, če bodisi premaknemo bodisi dodamo povezavo do vozlišča u. Potem velja $c_G(u) < g_H(u)$ in $u \in \mathcal{S}_c(H)$.

Potrebno je povdariti, da vsi razredi grafov ne zadoščajo pogoju 2, npr. razred dreves je zaprt za premik povezave ne pa za dodajanje povezav.

Zgled 3.2 Za stopenjsko centralnost $C_D(u) = deg(u)$ je lahko preveriti, da zadošča zgornjim aksiomom. Torej stopenjska centralnost je vozliščna centralnost po Sabadussijevi definiciji.

Zgled 3.3 Pokažimo, da vozliščna centralnost $c_E(u)$, ki temelji na ekscentričnosti $e(u) = \max\{d(u,v); v \in V(G)\}$, ni vozliščna centralnost glede na Sabidussijevo definicijo.

Slika 3.2: Polni graf K_5.

Na sliki imamo grafa G in H ter označene ekscentrične vrednosti za vsako vozlišče posebej. Graf G je pot dolžine 8, kjer imamo eno centralno vozlišče u_5. Ko dodamo povezavo (u_5, u_9), dobimo graf H, kjer je centralično vozlišče u_4. Opazimo, da ko dodamo povezavo se center grafa premakne in tako ne velja pogoj 5 iz zgornje definicije. Tudi pogoj 4 ne velja.

Bližnja centralnost je definirana kot $c_C(u) = s(u)^{-1}$, kjer je $s(u) = \sum_{v \in V(G)} d(u,v)$ vsota vseh razdalj od u do ostalih točk grafa. Kishi je v [91] pokazal, da ta centralnost ni vozliščna centralnost glede na Sabidussijevo definicijo.

Zgled 3.4 Oglejmo si primer bližnje centralnosti.

Na sliki so označene vsote vseh razdalj do vsakega vozlišča. V množici $\mathcal{S}_c(G)$ so vsebovana vozlišča u, u' in u''. Ko dodamo povezavo (u,v) dobimo graf H za katerega velja, da množica $\mathcal{S}_c(H)$ vsebuje vozlišče w. Torej velja $\mathcal{S}_c(G) \cap \mathcal{S}_c(H) = \emptyset$, kar pomeni, da ne velja pogoj 5.

Naj bo c neka realna vrednost funkcije definirane na vozliščih neusmerjenega povezanega grafa $G = (V(G), E(G))$ in naj bosta u in v nesosednji vozlišči grafa G. Ko grafu G dodamo povezavo (u,v), dobimo graf $H = (V(G), E \cup \{(u,v)\})$, kjer razliko centralnih vrednosti izračunamo kot $\Delta_{uv}(w) = c_H(w) - c_G(w)$. Kishi [91] je za definicijo vozliščne centralnosti izhajal iz Sabadussijeve.

Definicija 3.5 (Kishi [91]) Funkcijo c imenujemo *vozliščna centralnost* natanko tedaj ko veljata naslednja dva pogoja

Slika 3.3: Grafa G in H.

1. $\Delta_{uv}(u) > 0$, t.j., $c_G(u) < c_H(u)$.

2. Za vsak par nesosednjih vozlišč u in v velja naslednje:
   Če za $w \in V(G)$ velja $d(u,w) \leq d(v,w)$, velja tudi $\Delta_{uv}(u) \geq \Delta_{uv}(w)$.

Pogoja iz zgornje definicije sta podobna pogojema 4 in 5 iz Sabidussiove definicije. Torej ni presenetljio, da ekscentričnost ni vozliščna centralnost glede na Kishijevo definicijo. Če se vrnemo na primer 3.3, opazimo, da vozlišče u_5 ne usteza pogoju 2 zgornje definicije. Kishi [91] je pokazal tudi, da bližnja centralnost zadostuje njegovi definiciji vozliščne centralnosti.

Zgornji primeri kažejo, da je težko najti minimalne zahteve, ki bi jim zadoščal centralni indeks, ki temelji na razdalji.

3.4.2 Aksiomatizacija povratne centralnosti

Do zdaj smo imeli opravka z aksiomatizacijo centralnosti, ki temeljijo na najkrajši poti ali stopnji vozlišča. V tem delu bomo pogledali aksiomatizacojo, ki vodi do povratne centalnosti.

Pogledali si bomo dva primera aksiomatizacije. Za aproksimacijo obstaja več vrst pristopov.V literaturi je predlaganih veliko lastnosti, ki naj bi jim zadoščala centralnost, ampak te lastnosti so velikokrat odvisne od uporabe, ki jo želi avtor.

Prvi del bo temeljil na članku [101] van den Brinka in Gillesa. Tu bomo iskali povezavo med centralnostjo, ki bazira na stopnji in centralnostjo, ki temelji na povratnosti. Nadaljevali bomo z rezultati Volija in sodelavcev, ki so aksiomatično karakterizirali posebne povratne centralnosti.

Od stopnje do povratnosti

V [101] sta se Van den Brink in Gilles povečala usmerjenim grafom. Mi se bomo ukvarjali z neuteženimi usmerjenimi grafi, vendar se vse rezultate da posplošiti na utežene grafe.

Naš cilj je najti aksiomatsko karakterizacijo centralnosti, t.j., postaviti želimo aksiome, ki izhajajo iz lastnosti centalnosti. *Mera relacijske moči* priredi usmerjenemu omrežju z n vozlišči vektor dolžine n in na i-ti komponenti vektorja je mera relacijske moči oziroma dominantnost za vozlišče i.

Poglejmo si *β-mero* [100]. Naj bo $\mathcal{G}_n$ množica neuteženih usmerjenih grafov na n točkah. Vozlišče i *dominira* vozlišče j, če imamo v grafu usmerjeno povezavo $(i, j) \in E(G)$.

Definicija 3.6 Naj bo dana množica vozlišč V in $|V| = n$. *β-mera* na V je funkcija $\beta : \mathcal{G}_n \to \mathbb{R}^n$ podana s predpisom:

$$\beta_G(i) = \sum_{j \in N_G^+(i)} \frac{1}{d_G^-(j)} \quad \forall i \in V, G \in \mathcal{G}_n,$$

kjer je $d_G^-(j)$ vhodna stopnja in $N_G^+(i)$ množica vozlišč j, za katere obstaja usmerjena povezava $(i, j) \in E(G)$

β-mero si lahko predstavljamo, kot povratno centralnost, saj je vrednost za vozlišče i odvisna od lastnosti sosed v njegovi okolici.

Naslednji sklop štirih aksimov enolično določa β-mero. Naj bo $f : \mathcal{G}_n \to \mathbb{R}^n$ mera relacijske dominantnosti. Potrebujemo še naslednjo definicijo:

Definicija 3.7 *Particija* grafa $G \in \mathcal{G}_n$ je podmožica $\{G_1, \ldots, G_K\} \subseteq \mathcal{G}_n$, za katero velja:

- $\cup_{k=1}^K E_k = E$ in
- $E_k \cap E_i = \emptyset \forall 1 \leq k, l \leq K, k \neq l$.

Particijo imenujemo *neodvisna*, če velja tudi

$$\left|\left\{k \in \{1, \ldots, K\} : d_{G_k}^-(i) > 0\right\}\right| \leq 1 \quad \forall i \in V.$$

Za začetek mero normaliziramo, da lahko primerjamo dominantno vrednost različnih vozlišč, po možnosti v različnih omrežjih. Van den Brink in Gilles sta glede na dominantno strukturo predlagala, da vzamemo število dominiranih vozlišč kot skupno vrednost in jo razdelimo vozliščem glede na njihovo relacijsko moč.

Aksiom 1: *Normalizacija dominantnosti*
Za vsak $G \in \mathcal{G}_n$ velja

$$\sum_{i \in V(G)} f_G(i) = \left|\left\{j \in V(G) : d_G^-(j) > 0\right\}\right|.$$

Naslednji aksimon preprosto govori o tem, da če vozlišče ne dominira nobenega vozlišča, potem nima relacijske moči in zato ima vrednost 0:

Aksiom 2: *Navidezna lastnost vozlišča*
Za vsak $G \in \mathcal{G}_n$ in za $i \in V(G)$ velja $N_G^+ = \emptyset$, potem je $f_G(i) = 0$.

V tretjem aksimomu je formalizirano dejsto, da če imata dve vozlišči enako dominantno strukturo, t.j., da dominirata enako število vozlišč in sta dominirane od enakega števila vozlišč, potem dominantna vrednost enaka za ti dve vozlišči:

Aksiom 3: *Simetričnost*
Za vse $G \in \mathcal{G}_n$ in za $i, j \in V(G)$ velja $d_G^+(i) = d_G^+(j)$ in $d_G^-(i) = d_G^-(j)$, potem je $f_G(i) = f_G(j)$.

Zadnji četrti aksiom govori o sestavljanju usmerjenih grafov. Torej, če kombiniramo nekaj usmerjenih grafov tako, da so točke dominirane v največ enem od grafov, t.j., kombinacijo grafov lahko obravnavamo kot neodvisno particijo, potem je skupna dominantna vrednost vozlišč, enaka kar vsoti dominantnih vrednostih v posameznem usmerjenem grafu.

Aksiom 4: *Aditivnost glede na neodvisne particije*

Za vsak $G \in \mathcal{G}_n$ in vsako neodvisno particijo $\{G_1, \ldots, G_K\}$ grafa G velja

$$\boldsymbol{f}_G = \sum_{k=1}^{K} \boldsymbol{f}_{G_k}.$$

Zgornji aksiomi so deloma povezani tudi s prejšnjim poglavjem o centralnosti, ki temelji na razdalji. Če aksiom o normalizaciji dominantnosti malo spremenimo, potem je z njimi izhodno stopnjska centralnost enolično določena. Avtorji jo imenujejo *rezultatska mera*. Podobni rezultati veljajo tudi za utežene grafe.

Namesto, da vzamemo za skupno vrednost vsa vozlišča, ki so dominirana in to vrednost razdelimo glede na dominanco vsakemu vozlišču, raje za osnovo normalizacije vzamemo skupno vsoto vseh relacijskih moči:

Aksiom 1b: *Rezultatska normalizacija*

Za vsak $G \in \mathcal{G}_n$ velja

$$\sum_{i \in V(G)} f_g(i) = |E(G)|\,.$$

Če aksiom 1 zamenjamo z aksiomom 1b potem je naslednja funkcija enolična mera relacijske moči, ki zadošča aksiomom 2 do 4 in 1b:

$$\sigma_G(i) = d_G^+(i) \quad \forall i \in V(G), \quad G \in \mathcal{G}_n.$$

Zgoraj smo si pogledali množico aksiomov, ki opisujejo neko mero, ki ima nekaj elementov povratne centralnosti in neko zvezo s prejšnjim poglevjem preko rezultatske mere. Sedaj preidemo na povratno centralnost.

Povratna centralnost

Imamo usmerjen graf $G = (V(G), E(G))$ z utežmi ω na povezavah in utežmi α na vozliščih. V jeziku omrežij citiranja je $V(G)$ množica revij in povezava $(i, j) \in E(G)$, če je revija i citirana v reviji j. Utež $\omega(i, j)$ je število citiran revije i v reviji j in utež $\alpha(i)$ je definirana kot število člankov objavljenih v reviji i. Krepko povezani podgafi z dodatno lastnostjo, da ne obstaja pot, ki se začne v vozlišču zunaj podgrafa in konča v vozlišču podgafa (dovoljene so zanke). Palacio-Huerta in Volji sta poimenovala take podgrafe *disciplina*, kjer je disciplina poseben *komunikacijski razred* (krepko povezanih podgrafov), ki se definira kot ekvivalenčni razred glede na ekvivalenčno relacijo komunikacije. Reviji i in j *komunicirata*, če je bodisi $j = i$ bodisi i in j *vplivata* drug na drugega, kjer i vpliva na j, če obstaja zaporedje revij $i = i_0, i_1, \ldots, i_{K-1}, i_K = j$, kjer i_l citira i_{l-1}. To pomeni da obstaja pot od i do j.

Definirajmo $(|V(G)| \times |V(G)|)$-matrike

$$W = (\omega(i, j), \quad D_\omega = diag(\omega(\cdot j)), \text{ kjer je } \omega(\cdot j) = \sum_{i \in V(G)} \omega(i, j)$$

in WD_ω^{-1} je normalizirana uteženа matrika ter $D_\alpha = diag(\alpha(i))$. Potem je *problem razvrščanja* $\langle V, \alpha, W\rangle$ definiran za množico vozlišč $V(G)$ discipline, ki povezuje vozliščno utež α in ustrezno metriko citatov W in razvrščanje (vektor centralnosti) $\mathbf{c}_{PHV} \geq 0$ je normalizirano glede na l_1-normo: $\|\mathbf{c}_{PHV}\|_1 = 1$.

Obstajata dva posebna razreda problemov razvrščanja:

1. problem razvrščanja, kjer so vse vozliščne uteži enake, $\alpha(i) = \alpha(j) \quad \forall i, j \in V(G)$ (problem istega števila člankov) in

2. problem razvrščanja, kjer so vse *referenčne intenzivnosti* enake, $\frac{\omega(\cdot i)}{\alpha(i)} = \frac{\omega(\cdot j)}{\alpha(j)} \quad \forall i, j \in V(G)$ (homogen problem).

Da povežemo majhne in velike probleme, imamo *reduciran problem razvrščanja* R^k za problem razvrščanja $R = \langle V(G), \alpha, W\rangle$ glede na vektor dolžine k, ki je definiran kot $R^k = \left\langle V\setminus\{k\}, (\alpha(i))_{i\in V\setminus\{k\}}, (\omega_k(i,j))_{(i,j)\in V\setminus\{k\}\times V\setminus\{k\}}\right\rangle$, kjer je

$$\omega_k(i,j) = \omega(i,j) + \omega(k,j)\frac{\omega(i,k)}{\sum_{l\in V\setminus\{k\}}\omega(l,k)} \quad \forall i, j \in V\setminus\{k\}.$$

Poglejmo si problem razdelitve vozlišča j pri problemu razvrščanja $R = \langle V(G), \alpha, W\rangle$ v $|T_j|$ množic enakih vozlišč (j, t_j) za $t_j \in T_j$. Za $V' = \{(j, t_j); j \in V, t_j \in T_j\}$ *problem razvrščanja, ki izhaja iz razdelitve* j označimo kot

$$R' = \left\langle V', (\alpha'((j,t_j)))_{j\in J, t_j\in T_j}, (\omega'((i,t_i)(j,t_j)))_{((i,t_i)(j,t_j))\in V'\times V'}\right\rangle,$$

kjer sta

$$\alpha'((j,t_j)) = \frac{\alpha(j)}{|T_j|}, \quad \omega'((i,t_i)(j,t_j)) = \frac{\omega(i,j)}{|T_i|\,|T_j|}.$$

Zgornji dve definiciji posebnih problemov razvrščanja potrebujemo za formalizacijo naslednjih aksiomov.

Metoda razvrščanja Φ vsakemu problemu razvrščanja predpiše centralni vektor in zadoščati mora naslednjim aksiomom (vsaj šibki verziji):

Aksiom 1: *Invariantnost glede na referenčno itenzivnost*

Φ je invariantna glede na referenčno intenzivnost, če

$$\Phi(\langle V(G), \alpha, W\Gamma\rangle) = \Phi(\langle V(G), \alpha, W\rangle)$$

Za vsak problem razvrščanja $\langle V(G), \alpha, W\rangle$ in za vsako matriko $\Gamma = diag(\gamma_j)_{j\in V}$, kjer je $\gamma_j > 0 \quad \forall j \in V(G)$.

Aksiom 2: *(šibka) Homogenost*

(a) Φ zadošča *šibki homogenosti*, če za vsaka dva problema $R = \langle\{i, j\}, \alpha, W\rangle$, ki sta homogena in sta problema istega števila člankov velja

$$\frac{\Phi_i(R)}{\Phi_j(R)} = \frac{\omega(i,j)}{\omega(j,i)}.$$

(b) Φ zadošča *homogenosti*, če zgornja enakost pod točko (a) velja za vse homogene probleme.

Aksiom 3: *(šibka) Usklajenost*

(a) Φ zadošča *šibki usklajenost*, če za vse probleme istega števila člankov in homogene probleme $R = \langle V, \alpha, W \rangle$, kjer $|V(G)| = 2$ in $\forall k \in V(G)$

$$\frac{\Phi_i(R)}{\Phi_j(R)} = \frac{\Phi_i(R^k)}{\Phi_j(R^k)} \quad \forall i, j \in V(G) \backslash \{k\}\,.$$

(b) Φ zadošča *usklajenosti*, če zgornja enakost iz točke (a) velja za vse homoegene probleme.

Aksiom 4: *Invariantnost gelde na razcep revij*

Φ je inveraintna za razdelitev revij, t.j. za vsak problem rangiranja R in za vse razdelitve R' problema R velja

$$\frac{\Phi_i(R)}{\Phi_j(R)} = \frac{\Phi_{(i,t_i)}(R')}{\Phi_{(j,t_j)}(R')} \quad \forall i, j \in V(G), \quad \forall i \in T_i, \quad \forall j \in T_j.$$

Palacios-Huerta in Volijo sta pokazala, da je metoda razvrščanja za centralnost Pinski-Narina $\mathbf{c}_{PN}$, ki je podana kot rešitev enačbe

$$D_\alpha^{-1} W D_W^{-1} D_\alpha \mathbf{c} = \mathbf{c}$$

edina metoda rangiranja ki zdošča

- invariantnosti glede na referenčno itenzivnost (aksiom 1),
- šibki homogenosti (aksiom 2a),
- šibki usklajenosti (aksiom 3a) in
- invariantnosti gelde na razdelitev revij (aksiom 4).

Povezava z normalizacijo

V tem delu bomo opisali Ruhnaujino [102] raziskovanje normalizacije centralnosti. Njena ideja temelji na intuitivnem razumevanju centralnosti, ki jo je že formaliziru Freeman leta 1979 [103]:

> *"Človek, ki se postavi v center zvezde domneva, da je strukturno bolj centalen, kot katerikoli človek v katerikoli poziciji v kateremkoli omrežju podobne velikosti."*

To je ona formalizirala v definiciji vozliščne centralnosti za neusmerjene povezane grafe $G = (V(G), E(G))$.

Definicija 3.8 (Ruhnaujina aksioma vozliščne centralnosti) Naj bo $G = (V(G), E(G))$ neusmerjen in povezan graf z $|V| = n$ in naj bo $c_V : V \to \mathbb{R}$. c_V imenujemo *vozliščna centralnost*, če velja

1. $c_V(i) \in [0,1]$ za vse $i \in V$ in

2. $c_V(i) = 1$ natanko tedaj ko je G zvezda na n vozliščih in i je centralno vozlišče zvezde.

Vozliščna centralnost je uporabna, če želimo primerjati vozlišča različnih grafov. Poglejmo si primerjavo centralnega vozlišča zvezde velikosti n in kateregakoli vozlišča v polnem grafu K_n. Oba imata stopnjo $n-1$, ampak intuitivno ima center zvezde bolj pomembno vlogo v grafu kot katerokoli vozlišče v K_n.

Freeman [104] je pokazal, da centralnost posrednika na najkrajši poti zadošča zgornji definiciji. Glede na dejstvo, da je lastni vektor centralnosti normaliziran v evklidski normi ima lastnost, da je maksimalna dosegljiva vrednost $\frac{1}{\sqrt{2}}$ (neodvisno od n) in da je dosežena ravno v centru zvezde (v [105]). To je tudi vozlišča centralnost (pomnoženo z $\sqrt{2}$).

3.5 Stabilnost in občutljivost

Recimo, da smo omrežje malo spremenili, npr. dodali novo spletno povezavo ali stran (vozlišče) v primeru spletnega grafa. V takšnih situacijah nas pogosto zanima stabilnost, torej če spremembi navkljub izračunane centralnostne vrednosti ostanejo ustrezne ali pa morda postanejo popolnoma neustrezne.

V prvem podpoglavju bomo pod drobnogled vzeli stabilnost pri centralnostih, ki temeljijo na razdalji, natančneje si bomo ogledali, kako je z ekscentrično in bližinsko centralnostjo. Predstavili bomo tudi pojem stabilnega, kvazi-stabilnega in nestabilnega grafa in podali nekaj pogojev za obstoj le-teh. V drugem podpoglavju pa se bomo posvetili spletnim centralnostim in predstavili rezultate o njihovi numerični stabilnosti in o njihovi stabilnosti razvrstitve centralnostih vrednosti.

3.5.1 Stabilnost centralnosti, ki obravnavajo razdaljo

Osredotočili se bomo na stabilnost centra $\mathcal{S}_c(G) = \{u \in V : \forall v \in V \mid c(u) \geq c(v)\}$ glede na operacijo dodajanja povezave (u,v) med dve različni nesosedni vozlišči v grafu $G = (V,E)$.

Naj bo $u \in \mathcal{S}_c(G)$ centralno vozlišče glede na centralnost c in $(u,v) \notin G$. Z dodajanjem povezave (u,v) grafu G dobimo graf $H = (V, E \cup (u,v))$. Za center novega grafa H lahko velja bodisi $\mathcal{S}_c(H) \subseteq \mathcal{S}_c(G) \cup \{v\}$ bodisi $\mathcal{S}_c(H) \not\subseteq \mathcal{S}_c(G) \cup \{v\}$ za vsako vozlišče $v \in V$.

Kishi je graf, za katerega se po dodani povezavi zgodi drugo, tj.

$$\mathcal{S}_c(H) \not\subseteq \mathcal{S}_c(G) \cup \{v\},$$

v delu [91] poimenoval nestabilen graf glede na centralnost c. Sliki iz poglavja o aksiomatizaciji prikazujeta nestabilna grafa glede na ekscentrično in bližinsko centralnost.

Vrnimo se k možnim situacijam, ki se lahko zgodijo, ko grafu dodamo novo povezavo. Za analizo nam je ostala še prva možnost, torej ko $\mathcal{S}_c(H) \subseteq \mathcal{S}_c(G) \cup \{v\}$. Ta slučaj ločimo pri obravnavi grafa še na dve možnosti. V prvi možnosti veljata pogoja $\mathcal{S}_c(H) \subseteq \mathcal{S}_c(G)$ in $u \in \mathcal{S}_c(H)$ - če se to zgodi, obravnavani graf klasificiramo kot stabilen graf. Ostane nam še druga možnost, v kateri se zgodi vsaj ena od možnosti $\mathcal{S}_c(H) \not\subseteq \mathcal{S}_c(G)$ in $u \notin \mathcal{S}_c(H)$, pripadajočemu grafu pa rečemo kvazi-stabilen graf.

Tako definirana stabilnost grafa glede na centralnost c motivira Sabidussijevo zahtevo v definiciji vozliščne centralnosti, ki pravi, da naj povezava, ki jo dodamo centralnemu vozlišču $u \in \mathcal{S}_c(G)$, okrepi njegovo centralnost. Zahtevo najdemo v [92].

Na tem mestu bi bil dobrodošel kakšen zgled kvazi-stabilnega grafa, zato si v ta namen oglejmo primer slednjega za bližinsko centralnost.

Slika 3.4: Kvazi-stabilen graf glede na bližinsko centralnost. Označene vrednosti predstavljajo skupne razdalje $s(u)$. Z vstavljanjem povezave (u, v) postane nova mediana vozlišče v.

Na grafu 3.4 je ob vsakem vozlišču u označena statusna vrednost $s(u) = \sum_{v \in V} d(u, v)$. Vozlišče z največjo bližinsko centralnostjo c_C je tisto, ki mu pripada najmanjša statusna vrednost. Na levi strani, ki prikazuje začetni graf, je takšno vozlišče u, torej $u \in \mathcal{S}_c(G)$. Na desni pa s tem, ko dodamo povezavo (u, v), postane centralno vozlišče v, torej $v \in \mathcal{S}_c(H)$. Od tod vidimo, da je graf res kvazi-stabilen glede na bližinsko centralnost. Res, saj najprej opazimo, da velja $\mathcal{S}_c(H) = \{v\} \subseteq \{u, v\} = \mathcal{S}_c(G) \cup \{v\}$, kar nam zaenkrat pove le to, da graf ni nestabilen. Če klasificiramo še naprej, pa vidimo, da veljata tako pogoj $\mathcal{S}_c(H) = \{v\} \not\subseteq \{u\} = \mathcal{S}_c(G)$ kot tudi $u \notin \{v\} = \mathcal{S}_c(H)$, torej je graf res kvazi-stabilen za bližinsko centralnost.

Kishi je v [91] predstavil bolj posplošeno obliko bližinske centralnosti. Centralnostno vrednost $c_{GenC}(u)$ vozlišča $u \in V$ je definiral kot

$$c_{GenC}(u) = \frac{1}{\sum_{k=1}^{\infty} a_k n_k(u)}.$$

Pri tem $n_k(u)$ označuje število vozlišč, ki so na razdalji k od vozlišča u, vsak člen a_k pa predstavlja neko realno konstanto. Če vzamemo $a_k = k$, dobimo definicijo običajne bližinske centralnosti, torej

$$\frac{1}{\sum_{k=1}^{\infty} a_k n_k(u)} = \frac{1}{\sum_{v \in V} d(u, v)} = c_C(u).$$

Kishi in Takeuchi sta v [93] pokazala, pod katerimi pogoji obstaja stabilen, kvazi-stabilen in nestabilen graf za posplošene centralne funkcije c_{GenC}, ki smo jih definirali zgoraj.

Izrek 3.9 *Za vsako posplošeno vozliščno centralnost c_{GenC}, definirano s*

$$c_{GenC}(u) = \frac{1}{\sum_{k=1}^{\infty} a_k n_k(u)}$$

za $u \in V$, velja:

1. *če je $a_2 < a_3$, obstaja kvazi-stabilen graf, in*
2. *če je $a_3 < a_4$, obstaja nestabilen graf.*

Izrek 3.10 *Vsak povezan neusmerjen graf G je stabilen, natanko tedaj ko posplošena vozliščna centralnost c_{GenC} iz izreka 3.9 zadošča pogoju $a_2 = a_3$. Še več, G ni nestabilen, natanko tedaj ko c_{GenC} zadošča $a_3 = a_4$.*

Sabidussi je v [92] pokazal še, da so grafi iz razreda neusmerjenih dreves stabilni grafi glede na bližinsko centralnost c_C.

Izrek 3.11 *Če je neusmerjen graf G drevo, je G stabilen za bližinsko centralnost.*

3.5.2 Stabilnost in občutljivost spletnih centralnosti

Najprej bomo obravnavali stabilnost glede na centralne vrednosti, čemur pravimo numerična stabilnost, kasneje pa bomo podali še nekaj izsledkov raziskav glede stabilnosti razvrstitve centralnih vrednosti, kar je poznano pod imenom stabilnost rangiranja (razvrščanja).

Numerična stabilnost

Langville in Meyer sta v [94] opazila, da npr. pri opazovanju stabilnosti PageRank-a ni smiselno obravnavati formulacije v obliki linearnega sistema in pripadajoče občutljivosti matrike sistema, tj. $\kappa(A) = \|A\|\|A^{-1}\|$ za obrnljivo matriko A, ker se lahko zgodi, da se rešitveni vektor linearnega sistema občutno spremeni, medtem ko pa normalizirani rešitveni vektor ostane praktično enak. Tako moramo namesto občutljivosti linearnega sistema obravnavati stabilnost problema lastnih vektorjev, ki je sam po sebi osnova za številne spletne centralnosti.

Ng je skupaj s soavtorji v [95] predstavil zgled, ki nam pokaže, da se lahko lastni vektor občutno spremeni, čeprav se pripadajoče omrežje le malo spremeni. Obravnaval je množico spletnih strani, pri čemer je 100 strani vsebovalo povezavo na *algore.com*, drugih 103 strani pa na *georgewbush.com*. Na levem grafu slike 3.5 sta narisana pripadajoča prva dva lastna vektorja (oziroma projekciji na njuni neničelni komponenti). Desni graf pa prikazuje, kako se spremeni situacija, če dodamo pet novih strani, ki vsebujejo povezavi na obe spletni strani, tako na *algore.com* kot tudi na *georgewbush.com*. Opazimo, da se perturbacija na grafu odraža kot močan zamik lastnih vektorjev.

Slika 3.5: Zgornji primer kaže na nestabilnost, ki jo povzroči majhna perturbacija grafa.

Isti avtorji so objavili še zgled o stabilnosti algoritma Hubs & Authorities, ki je ponazorjen na spodnji sliki in pod njo podrobneje razložen.

Slika 3.6: Nestabilnost kot posledica različnih vrzeli med lastnimi vrednostmi. Na levi sliki se zavoljo majhne vrzeli položaj lastnih vektorjev občutno spremeni.

Med izvajanjem Hubs & Authorities algoritma se izračuna največji lastni vektor matrike $S = A^T A$. Na zgornji sliki je na levem grafu s polno sklenjeno krivuljo predstavljen rob kvadratične forme $x^T S_1 x$ za matriko S_1, s črtkano krivuljo pa kvadratična forma za malo perturbirano matriko. Narisani so tudi pripadajoči lastni vektorji. Vektorja, narisana s polnima črtama, pripadata matriki S_1, vektorja, narisana s črtkanima črtama, pa malo perturbirani matriki S_1. Analogna razlaga velja tudi za desni graf za matriko S_2. Pri tem sta obe matriki S_1 in S_2 perturbirani ekvivalentno. Lahko opazimo, da se v levem primeru lastni vektorji močno premaknejo, v drugem primeru pa je sprememba praktično neopazna. Razlog za drugačen odziv na perturbacijo v obeh primerih tiči v različnih vrzelih med lastnimi vrednostmi. Vrzel med lastnimi vrednostmi je definirana kot razlika med prvo in drugo največjo lastno vrednostjo in jo označimo z δ. V našem primeru ima matrika S_1 vrzel blizu 0, tj. $\delta_1 \approx 0$, za matriko S_2 pa velja $\delta_2 = 2$. Vidimo torej, da je lahko algoritem Hubs & Authorities v primeru, ko je razlika med največjima lastnima vrednostima blizu 0, zelo občutljiv glede na majhne spremembe v matriki, medtem ko je

občutljivost majhna, če je razlika med lastnima vrednostima velika. Ng je to obnašanje utemeljil tudi teoretično:

Izrek 3.12 *Za dano matriko $S = A^T A$ naj bo c_{HA-A} dominantni lastni vektor in δ razlika med največjima dvema lastnima vrednostima matrike S. Naj bo $d^+(i) \leq d$ za vsak $i \in V$ in $\varepsilon > 0$. Če graf spleta perturbiramo tako, da eni spletni strani dodamo ali odstranimo največ k povezav za $k < (\sqrt{d+\alpha} - \sqrt{d})^2$, pri čemer je $\alpha = \frac{\varepsilon\delta}{4+\sqrt{2}\varepsilon}$, za perturbirani dominantni lastni vektor $\tilde{c}_{HA-A}$ perturbirane matrike $\tilde{S}$ velja $\|c_{HA-A} - \tilde{c}_{HA-A}\|_2 \leq \varepsilon$.*

V smislu zgornjega primera nam izrek res pove, da bo pri večji razliki med največjima dvema lastnima vrednostima δ sprememba lastnih vektorjev majhna. Res, za večji δ bo α večji, torej bo tudi meja za k, pri kateri se lastni vektorji le malo spremenijo, višja. To nam ne pove nič drugega kot to, da lahko za večji δ z večjo motnjo na matriki lastne vektorje le malo spremenimo, torej da je problem malo občutljiv.

Izrek 3.13 *Če je S simetrična matrika z razliko med največjima dvema lastnima vrednostima δ, obstaja perturbirana verzija $\tilde{S}$ matrike S, za katero velja $\|S - \tilde{S}\|_F = \mathcal{O}(\delta)$, ki povzroči veliko spremembo (reda $\Omega(1)$) na dominantnem lastnem vektorju. Pri tem je $\|X\|_F = \left(\sum_i \sum_j (x_{ij}^2)\right)^{1/2}$ Frobeniusova norma matrike X.*

Če vzamemo pod drobnogled algoritem PageRank, opazimo, da za Markovsko verigo s prehodno matriko P velja, da je občutljivost dominantnega lastnega vektorja določena z razliko med drugo največjo lastno vrednostjo in 1. Haveliwala in Kamvar sta v [96] pokazala, da za matriko PageRank-a, za katero ima P vsaj dve ireducibilni zaprti podmatriki, velja $\lambda_2 = d$. To velja tudi v primeru, ko je vektor $\mathbf{1}_n$ iz formule $\boldsymbol{c}_{PR} = dP\boldsymbol{c}_{PR} + (1-d)\mathbf{1}_n$ (d je faktor dušenja) zamenjan s poljubnim stohastičnim vektorjem $\boldsymbol{v}$, ki mu pravimo vektor personalizacije. Zato da faktor dušenja $d = 0.85$, ki so ga predlagali ustanovitelji Googla, v splošnem precej stabilnejše rezultate kot $d = 0.99$, ki bi si ga želeli izbrati, če bi si bila originalni in perturbirani graf spleta poljubno blizu.

Ng je v [95] s soavtorji dokazal še sledeči izrek:

Izrek 3.14 *Naj bo $U \subseteq V$ množica spletnih strani, na katerih spremenimo izhodne povezave, $\boldsymbol{c}_{PR}$ prvotna vrednost PageRank-a in $\boldsymbol{c}_{PR}^U$ nova vrednost PageRank-a za perturbirano situacijo. Potem velja*

$$\|\boldsymbol{c}_{PR} - \boldsymbol{c}_{PR}^U\|_1 \leq \frac{2}{1-d} \sum_{i \in U} c_{PR}(i).$$

Bianchini, Gori in Scarselli so v [97] uspeli še dodatno zmanjšati zgornjo mejo izreka iz 3.14. Pokazali so, da velja:

Izrek 3.15 *Naj veljajo pogoji iz izreka 3.14. Potem velja*

$$\|\boldsymbol{c}_{PR} - \boldsymbol{c}_{PR}^U\|_1 \leq \frac{2d}{1-d} \sum_{i \in U} c_{PR}(i).$$

Pri tem je $d < 1$.

3.5.3 Stabilnost razvrščanja

Pri spletnih centralnostih so rezultati v splošnem vrnjeni v obliki seznama spletnih strani, ki ustrezajo iskalni poizvedbi. Vrednosti, ki jih dosežejo spletne strani, običajno niso prikazane in tako se na tem mestu porodi vprašanje, če numerična stabilnost implicira tudi stabilnost glede na razvrstitev v seznamu, torej t.i. stabilnost razvrščanja. Lempel in Moran sta v [98] raziskovala stabilnost razvrščanja pri treh glavnih predstavnikih algoritmov, ki imajo opravka s spletnimi centralnostmi.

Izkaže se, da iz numerične stabilnosti ne sledi nujno stabilnost razvrščanja. To lahko vidimo na primeru grafa $G = (V, E)$ s spodnje slike.

Slika 3.7: Graf G, na katerem opazujemo stabilnost razvrščanja algoritma PageRank. Graf G_a dobimo z dodajanjem usmerjene povezave od vozlišča y do h_a, graf G_b pa z dodajanjem usmerjene povezave od y do h_b.

Vsaka neusmerjena povezava $[u, v]$ na G predstavlja dve usmerjeni povezavi (u, v) in (v, u). Iz G naredimo dva različna grafa $G_a = (V, E \cup \{(y, h_a)\})$ in $G_b = (V, E \cup \{(y, h_b)\})$. PageRank vektor $\boldsymbol{c}^a_{PR}$, ki pripada G_a, zadošča

$$0 < c^a_{PR}(x_a) = c^a_{PR}(y) = c^a_{PR}(x_b),$$

in zato $c^a_{PR}(h_a) > c^a_{PR}(h_b)$.

Analogno v grafu G_b velja

$$0 < c^b_{PR}(x_a) = c^b_{PR}(y) = c^b_{PR}(x_b),$$

in tako $c^b_{PR}(h_a) < c^b_{PR}(h_b)$.

Če povzamemo, vidimo, da s tem, ko premaknemo eno samo izhodno povezavo iz vozlišča y s slabo uvrstitvijo na razporedu vozlišč glede na centralni indeks, povzročimo popolno spremembo v celotni razvrstitvi, natančneje

$$c^a_{PR}(a_i) > c^a_{PR}(b_i) \;\text{ in }\; c^b_{PR}(a_i) < c^b_{PR}(b_i) \qquad \forall i.$$

Da bomo lahko določili, če je algoritem stabilen glede na razvrščanje, moramo za začetek natančno definirati stabilnost razvrščanja. Upoštevajmo takšno definicijo, kot si jo je zamislil Borodin s soavtorji v delih [99] in [98].

Definicija 3.16 Naj bo $\mathcal{G}$ množica usmerjenih grafov in $\mathcal{G}_n$ podmnožica $\mathcal{G}$, v kateri imajo vsi usmerjeni grafi n vozlišč.

1. Za vektorja razvrščanja $\boldsymbol{r}^1$ in $\boldsymbol{r}^2$, ki pripadata množici vozlišč moči n, definiramo *razdaljo razvrščanja* med njima kot

$$d_r(\boldsymbol{r}^1, \boldsymbol{r}^2) = \frac{1}{n^2} \sum_{i,j=1}^{n} l_{i,j}^{\boldsymbol{r}^1,\boldsymbol{r}^2},$$

pri čemer je

$$l_{i,j}^{\boldsymbol{r}^1,\boldsymbol{r}^2} = \begin{cases} 1, & r_i^1 < r_j^1 \text{ in } r_i^2 > r_j^2 \\ 0, & \text{sicer} \end{cases}.$$

2. Za algoritem $\mathcal{A}$ rečemo, da je *stabilen glede na razvrščanje* za $\mathcal{G}$, če za vsak fiksen k velja

$$\lim_{n\to\infty} \max_{\substack{G_1,G_2\in\mathcal{G}_n \\ d_e(G_1,G_2)\leq k}} d_r(\mathcal{A}(G_1), \mathcal{A}(G_2)) \longrightarrow 0,$$

kjer je

$$d_e(G_1, G_2) = |(E_1 \cup E_2)\backslash(E_1 \cap E_2)|.$$

Tako je algoritem $\mathcal{A}$ stabilen glede na razvrščanje za grafe $\mathcal{G}$, če za vsak k spremembe, ki se zgodijo v razvrstitvi vozlišč zaradi spreminjanja k povezav, izginejo s tem, ko gre število vozlišč grafa v limiti proti neskončnosti.

Borodin je s soavtorji pokazal, da niti Hubs & authorities algoritem niti metoda SALSA nista stabilna glede na razvrščanje za množico vseh usmerjenih grafov $\bar{\mathcal{G}}$.

Pozitiven rezultat pa je dosegel, ko je opazoval stabilnost za posebno podmnožico $\bar{\mathcal{G}}$, konkretneje za množico 'authority povezanih' usmerjenih grafov $\mathcal{G}^{ac}$:

Definicija 3.17

1. Vozliščema $p, q \in V$ pravimo *ko-citirana*, če obstaja vozlišče $r \in V$, da velja $(r, p), (r, q) \in E$.

2. p in q sta povezana s *ko-citirano potjo*, če obstajajo vozlišča $p = v_o, v_1, \ldots, v_{k-1}, v_k = q$, tako da je par (v_{i-1}, v_i) ko-citiran za vsak $i = 1, 2, \ldots, k$.

3. Usmerjeni graf $G = (V, E)$ imenujemo *'authority povezan'*, če za vsak par vozlišč p, q, ki zadošča $d^-(p), d^-(q) > 0$, obstaja ko-citirana pot.

Lempel in Moran sta mnenja, da se je razumno pri obravnavanju stabilnosti omejiti le na to podmnožico usmerjenih grafov zavoljo spodnjih opazk:

- če sta p in q ko-citirana, pokrivata isto temo,
- relevantnost p in q bi morali meriti pod enakimi pogoji,
- nikogar ne zanimajo odgovori na vprašanja, kot so: „Ali je p boljši geografski vir kot je q 'authority' glede na šport?“.

Za 'authority povezane' podgrafe velja:

- SALSA je stabilen glede na razvrščanje za $\mathcal{G}^{ac}$,
- Hubs & Authorities algoritem ni stabilen glede na razvrščanje za $\mathcal{G}^{ac}$,

- Page-Rank ni stabilen glede na razvrščanje za $\mathcal{G}^{ac}$.

Do zadnjih dveh rezultatov sta prišla Lempel in Moran v [98]. S tem smo v grobem pokrili vse pomembnejše izsledke o občutljivosti in stabilnosti spletnih centralnosti.

Poglavje 4

Lokalna gostost

Maruša Povhe, Nev a Port, Matej Pirnat, Špela Podgoršek

Udeleženci v omrežjih ponavadi ne delujejo sami. S selektivnim procesom vzpostavljanja odnosov z drugimi udeleženci oblikujejo skupine oziroma grupe. Grupe so pogosto ustanovljene s skupnim ciljem, interesom, s skupnimi preferencami ali drugimi podobnostmi. Standardni primeri vključujejo odnose osebnega poznanstva, odnose sodelavcev različnih družbenih področij ter koalicijske ali pogodbene odnose na trgih. Kohezija oziroma povezanost znotraj teh grup jim omogoča, da vplivajo na delovanje celotnega omrežja. Odkrivanje povezanih grup je ključen element v analizi omrežij. Za računsko obdelavo potrebujemo formalne koncepte, ki kažejo na nek intuitivni pomen povezanosti. Na splošni ravni so sledeče lastnosti pripisane povezani grupi.

***vzajemnost*:** Člani grupe izbirajo med seboj, kdo bo vključen v grupo. V smislu teorije grafov to pomeni, da so sosednji.

***kompaktnost*:** Člani grupe so dobro dosegljivi med sabo, čeprav ni nujno, da so sosednji. V smislu teorije grafov - biti dobro dosegljiv je mogoče razlagati, kot imeti kratko razdaljo ali visoko povezanost.

***gostost*:** Člani grupe imajo veliko stikov med seboj. V smislu teorije grafov - člani grupe imajo veliko sosedov znotraj grupe.

***ločenost*:** Člani grupe imajo več stikov znotraj grupe kot zunaj nje.

Odvisno od omrežja, ki nas zanima, so različni koncepti lahko uporabljeni z vključevanjem lastnosti povezanosti z različnimi poudarki. Pojmi, pri katerih je gostota dominantni vidik, so še posebej pomembni.

Gostota ima izjemen pomen v socialnih omrežjih. Po eni strani so z nedavnimi študiji odkrili, da socialna omrežja kažejo na asortativno mešanje točk, tj. ponavadi imajo lastnost, da imajo sosedi točke z visoko stopnjo tudi sami visoko stopnjo. Asortativno mešanje je izraz tipične ugotovitve, da so socialna omrežja sestavljena po grupah glede na

visoko gostoto. Po drugi strani pa obstaja več matematičnih rezultatov, ki dokazujejo, da velika gostota pomeni tudi druge značilnosti povezanosti. Na primer, en klasičen rezultat pravi, da če bi imel vsak član grupe vezi z vsaj *1/k* drugimi člani grupe, potem je razdalja vezi med grupami kvečjemu *k*. Tukaj odvisnost od gostote ni tako močna kot v primeru oddaljenosti.
V tem poglavju bomo pregledali računske pristope in rešitve za odkrivanje lokalno gostih grup. V teoriji grafov je grupna lastnost lokalna, če je opredeljena v podgrafih, ki jih lahko povzročijo samo grupe. Lokalnost ne ustreza zgoraj omenjenim lastnostim povezanosti, odkar zanemarja grupe zunaj omrežja. Dejstvo je, da ima večina pojmov, ki so bili opredeljeni za kritje povezanosti, maksimalno stanje. To pomeni, da je za grupo, ki je povezana glede na nekatere lastnosti, potrebno tudi, da ni vsebovana v kateri koli večji grupi omrežja. Maksimalnost ni lokalna. Pojme prikazujemo na podlagi njihovih temeljnih lastnosti teorije grafov in brez dodatnih zahtev maksimalnosti. Namesto tega se maksimalnost pojavlja v povezavi z nekaterimi računskimi problemi, ki izhajajo iz teh pogojev. To poudarja, da je lokalnost pomemben skriti vidik povezanih grup: biti nespremenjen v okviru omrežnih sprememb zunaj grupe. Notranja trdnost in stabilnost je prirojena kakovost grup.
Primer polne grupe je klika. Od uvedbe v sociologiji leta 1949, so bila številna prizadevanja v kombinatorični optimizaciji in algoritmih posvečena reševanju računskih problemov za klike, zato si obravnava algoritmov in strogost rezultatov za klike zasluži velik del tega poglavja. V poglavju 4.1 vam predstavimo nekaj podrobnih orientacijskih rezultatov. Vsi ostali pojmi, o katerih razpravljamo, so milejši za koncept klik. Razlikujemo med strukturno in statistično omilitvijo. Značilnost strukturnih gostost je, da morajo vsi člani grupe izpolnjevati zahteve za članstvo v grupi. Pojmi (plex, jedro) priznajo močne trditve o strukturi znotraj grup. Strukture gostih grup so obravnavane v poglavju 4.2. V statistično gostih grupah mora biti lastnost, ki definira članstvo skupine, izpolnjena šele v povprečju (ali pričakovanju) vseh članov grupe. Na splošno, statistično goste grupe razkrivajo le nekaj vpogledov v sestavo grupe. Kakorkoli, te grupe se uporabljajo v negotovosti informacij, ki so obravnavani v poglavju 4.3. Kratek seznam pogosto uporabljenih nelokalnih pojmov je obravnavan v poglavju 4.4.
Vsi algoritmi so predstavljeni izključno v primeru neuteženih, neusmerjenih enostavnih grafov. Večina se jih lahko zlahka prevede za usmerjene ali utežene grafe. V nekaterih primerih, kjer so potrebne nove ideje, to omenimo eksplicitno.

4.1 Popolnoma goste grupe - klike

Klika je graf s popolno povezanostjo.

Definicija 4.1 Naj bo $G = (V, E)$ neusmerjen graf. Podmnožici $U \subseteq V$ pravimo *klika* natanko tedaj, ko je $G[U]$ poln graf.

V kliki je vsaka točka povezana z ostalimi. Klika U je *lokalno maksimalna klika* v grafu $G = (V, E)$ natanko tedaj, ko v G ne obstaja klika U', za katero velja $U \subset U'$. Klika je *globalno maksimalna klika* v grafu G natanko tedaj, ko ima maksimalno moč med vsemi klikami v G.

Strukturne lastnosti klik:

1. Klike so popolnoma goste, to pomeni, če je U klika velikosti k, potem je $\delta(G[U]) = \bar{d}(G[U]) = \Delta(G[U]) = k-1$. Višja stopnja ni mogoča.

2. Klike so popolnoma kompaktne, kar pomeni, da je $\text{diam}(G[U]) = 1$. Krajša razdalja med katerimakoli dvema točkama ni mogoča.

3. Klike so popolnoma povezane, torej, če je U klika velikosti k, potem je $(k-1)$-točkovno povezana in $(k-1)$-povezana. Višja povezanost ni mogoča.

Sledeči izrek nam da zadostne pogoje za obstoj klik določenih velikosti, z upoštevanjem velikosti celotnega omrežja.

Izrek 4.2 (Turán,1941) *Naj bo $G = (V, E)$ neusmerjen graf, $|V| = n$ in $|E| = m$. Če je $m > \frac{n^2}{2} \cdot \frac{k-2}{k-1}$, potem obstaja klika velikosti k znotraj G.*

Posledica izreka je, da mora biti omrežje samo po sebi gosto, da zagotovo vsebuje veliko kliko. Socialna omrežja so ponavadi redka, tako da nimamo a priori dokaza o obstoju klike in iskanje klik postane algoritmična naloga. Tudi, če bi vedeli, da obstaja klika določene velikosti v omrežju, je ne bi bili zmožni najti v razumnem času, kot bomo videli v nadaljevanju.
Lokalno maksimalne klike vedno obstajajo v grafu. Pravzaprav jih je veliko in težijo k temu, da se prekrivajo. To pomeni, da imamo v splošnem lahko primer, da lokalno maksimalni kliki U_1, U_2 obstajata in zadoščata pogojema: $U_1 \neq U_2$ in $U_1 \cap U_2$ neprazen [56].

Izrek 4.3 (Moon and Moser, 1965) *Vsak neusmerjen graf G z n točkami ima največ $3^{\lceil \frac{n}{3} \rceil}$ lokalno maksimalnih klik.*

Pričakovano število lokalno maksimalnih klik vodi k resnemu problemu prepoznavanja bolj pomembnih med njimi. Na voljo je le nekaj algoritmičnih tehnik, ki zagotavljajo uporabno razlago lokalno maksimalnih klik.

Družina vseh klik določenega grafa nam kaže neko strukturo:

1. Klike so zaprte za izključitev, to pomeni, če je U klika v G in $v \in U$, potem je tudi $U - \{v\}$ klika.

2. Klike so ugnezdene, to pomeni, da vsaka klika velikosti n vsebuje kliko velikosti $n-1$ (celo n klik velikosti $n-1$). Kljub temu, da je to posledica zaprtja za izključitev, je tudi dokazana lastnost za sorodne pojme, ki pa niso zaprti za izključitev.

Klike osnovane na razdalji. Obstaja več pristopov za posploševanje pojma klike, ki so pomembni za teorijo socialnih omrežij. Naj bo $G = (V, E)$ neusmerjen graf, U podmnožica točk in $N > 0$ neko naravno število.

1. U pravimo *N-klika* natanko tedaj, ko je za vsak $u, v \in U$, $d_G(u, v) \leq N$.

2. U pravimo *N-club* natanko tedaj, ko je $\text{diam}(G[U]) \leq N$.

3. U pravimo *N-clan* natanko tedaj, ko je U lokalno maksimalna N-klika in je $\text{diam}(G[U]) \leq N$.

Razdalja med točkama u in v v N-kliki je merjena ob upoštevanju grafa G in ne $G[U]$. Posledično N-klike niso nujno povezane za $N > 1$. Pojavile so se kritike klik osnovanih na razdalji. Prvi razlog je, da imajo v mnogo primerih omrežja globalno majhen premer, tako da je razdalja precej groba mera za iskanje pomembnih podstruktur. Drugi razlog pa je, da klike osnovane na razdalji, v splošnem niso niti zaprte za izključitev, niti ugnezdene.

4.1.1 Reševanje problemov

Od tu naprej privzemimo, da je n število točk grafa G in m število povezav.

V mnogih primerih so klike preprosti objekti, ki jih lahko obvladujemo s pomočjo algoritmov. Imamo algoritme s časovno zahtevnostjo $O(n+m)$, ki nam pomagajo pri reševanju problemov:

1. Določijo, če je dana množica točk $U \subseteq V$ klika v G. Preprosto testiramo ali je vsak par točk iz U povezan v G. To je do $\binom{n}{2}$ parov, vendar tudi če imamo manj povezav, smo po testiranju m parov gotovi.

2. Določijo, če je dana klika $U \subseteq V$ lokalno maksimalna v G. Preprosto testiramo ali obstaja točka v $V - U$, ki je sosednja vsem točkam v U. Zopet smo v najslabšem primeru konec po m testiranjih.

Predpostavimo, da je množica točk grafa $G = (V, E)$ urejena. Pravimo, da je množica $U \subseteq V$ leksikografsko manjša kot množica $U' \subseteq V$ natanko tedaj, ko prva točka, ki ni hkrati v U in U', pripada U. Sledi:

3. Poiščejo leksikografsko najmanjšo lokalno maksimalno kliko, ki vsebuje neko kliko U'. Na začetku določimo $U := U'$ in ponavljamo za vse $v \in V - U$, v naraščajočem vrstnem redu ter testiramo za vsak v ali je $U \subseteq N(v)$. Če to velja, potem dodamo točko v v U. Ko končamo je U lokalno maksimalna klika, ki vsebuje U'. Časovna zahtevnost algoritma je $O(n+m)$.

Težave v algoritmom nastopijo le, kadar želimo najti klike določenih velikosti, ozirima globalno maksimalne klike. Za take primere ne poznamo algoritmov s časovno zahtevnostjo z enakim redom velikosti kot zgoraj in najverjetneje sploh ne obstajajo.

4.1.2 Iskanje globalno maksimalnih klik

Kliko maksimalne velikosti je lažje poiskati, če nas ne zanima, koliko časa porabimo za to. Očiten pristop, ki ga uporabimo je *exhaustive search*. V algoritmu preprosto naštejemo vse možne kandidate množic $U \subseteq V$ in preverimo, če je U klika. Algoritem vrne največjo najdeno kliko. Preprosta ocena zgornje meje časovne zahtevnosti je v najslabšem primeru $O(n^2 \cdot 2^n)$.

Računska zahtevnost. Zanima nas, ali lahko izboljšamo časovno zahtevnost exhaustive search algoritma. Na žalost to verjetno ne bo šlo. Računsko je iskanje globalno maksimalne klike težek problem. Zapišimo problem:

Problem: KLIKA
Vhodni podatki: Graf G, parameter $k \in \mathbb{N}$
Vprašanje: Ali obstaja klika velikosti vsaj k znotraj G?

Naj $\omega(G)$ označuje velikost globalno maksimalne klike grafa G. Če imamo algoritem, ki določa problem iskanja klike v času $T(n)$ z iskanjem na podlagi binarnega drevesa, potem lahko izračunamo $\omega(G)$ s časovno zahtevnostjo $O(T(n) \cdot \log n)$. Obratno nam da vsak $T(n)$ algoritem za računanje $\omega(G)$, $T(n)$ algoritem za določanje problema iskanja klike. Če imamo torej algoritem s polinomsko časovno zahtevnostjo za problem iskanja klike, potem bomo imeli algoritem s polinomsko časovno zahtevnosto za računanje velikosti globalno maksimalne klike in obratno.

Izrek 4.4 *Problem iskanja klike je $\mathcal{NP}$-poln problem.*

Dokaz. Dokaz je podrobneje opisan v [57], mi pa si poglejmo skico dokaza. Testiranje, ali je neka množica klika, je možno s polinomsko časovno zahtevnostjo. Predpostavimo, da imamo dano *Booleanovo formulo* H v zvezni normalni obliki, sestavljeno iz m klavzul $C_1, \ldots, C_k$. Za H konstruiramo k-delni graf G_H, kjer so točke konstante v H, označene glede na njihove klavzule in kjer so konstante povezane tako, da niso negacije druga druge. Bolj natančno, definirajmo $G_H = (V_H, E_H)$ kjer je

$$V_H =_{\text{def}} \{(L, i) \mid i \in \{1, \ldots, k\} \text{ in } L \text{ konstanta v klavzuli } C_i\}$$

$$E_H =_{\text{def}} \{(L', j)\}\{(L, i) \mid i \neq j \text{ in } L \neq \neg L'\}$$

Graf G_H lahko dobimo s polinomsko časovno zahtevnostjo z uporabo formule H. Pokažimo, da je formula H dobra natanko tedaj, ko graf G_H vsebuje kliko velikosti k.

Predpostavimo, da je formula H dobra. Potem obstaja neka vloga spremenljivk $x_1, \ldots, x_n$, tako da je v vsaki klavzuli ena konstanta prava. Naj bodo $L_1, \ldots, L_k$ take konstante. Potem mora držati, da je $L_i \neq \neg L'_j$ za $i \neq j$. Tako dobimo, da je množica $\{(L_1, 1), \ldots, (L_k, k)\}$ klika velikosti k v grafu G_H.

Sedaj pa predpostavimo, da je $U \subseteq V_H$ klika velikosti k v grafu G_H. Ker je G_H k-delen, U vsebuje natanko eno točko iz vsakega dela V_H. Po definiciji množice V_H velja, da je za vse točke (L, i) in (L', j) iz U, $L \neq \neg L'$ kadarkoli je $i \neq j$. Zato lahko določimo vrednosti spremenljivkam, tako da vse konstante, vsebovane v U zadoščajo in dobimo, da je formula H dobra.

□

Če ne drži $\mathcal{P} = \mathcal{NP}$, ne obstaja algoritem s polinomsko časovno zahtevnostjo v n za reševanje problema iskanja klike neke velikosti, oziroma globalno maksimalne klike. Po drugi strani pa ne bi mogli najti klike velikosti k v grafu G s polinomsko časovno zahtevnostjo, tudi če bi vedeli, da obstaja.

Posledica 4.5 *Če ne drži $\mathcal{P} = \mathcal{NP}$, ne obstaja algoritem s polinomsko časovno zahtevnostjo za iskanje klike velikosti k v grafu, kjer vemo, da klika velikosti k obstaja.*

Dokaz. Predpostavimo, da imamo algoritem A s polinomsko časovno zahtevnostjo za vsak vhodni podatek (G, k), ki nam vrne kliko velikosti k, če ta obstaja. A lahko spremenimo v algoritem A', ki določa problem iskanja klike v polinomskem času. Poženemo algoritem A in če nam ta ne vrne rezultata, zavrnemo primer. Če pa vrne množico U, potem testiramo ali je U klika. Ta postopek ima zagotovo polinomski čas.

□

Težavnost iskanja klike ni odvisna od njene velikosti. Celo velikih klik (velikosti $(1-\varepsilon)n$ za $\varepsilon > 0$) se ne da poiskati, če ne drži $\mathcal{P} = \mathcal{NP}$ [89, 88]. Situacija se izboljša, če vzamemo poljubno izbrane grafe, kjer se vsaka povezava pojavi z verjetnostjo $\frac{1}{2}$. Predpostavimo, da postavimo poljubno kliko velikosti k v tak graf velikosti n. Vprašanje je, kako hitro lahko najdemo to kliko. Če je $k = \Omega(\sqrt{n \log n})$, potem kliko zagotovo sestavlja k točk z največjo stopnjo. To nam da algoritem s časovno zahtevnostjo $O((n+m)\log n)$. Za $k = \Omega(\sqrt{n})$ najdejo klike velikosti k v polinomskem času algoritmi, ki bazirajo na spektralnih tehnikah. Kakorkoli, veliko algoritmičnih tehnik ne najde klik velikosti $k = o(\sqrt{n})$[52].

Boljši algoritmi z eksponentno časovno zahtevnostjo. Kljub temu, da najverjetneje ne bomo nikoli uporabili algoritma s polinomsko časovno zahtevnostjo za iskanje globalno maksimalnih klik, lahko poskusimo oblikovati hitre algoritme s super-polinomsko časovno zahtevnostjo. Exhaustive search algoritem nam da zgornjo mejo časovne zahtevnosti $O(n^2 \cdot 2^n)$, oziroma $O^*(2^n)$, kadar opustimo polinomske faktorje. Naš cilj je oblikovati algoritme s časovno zahtevnostjo $O^*(\beta^n)$, s čim manjšo možno β.

Izrek 4.6 *Obstaja algoritem za iskanje globalno maksimalne klike s časovno zahtevnostjo* $O^*(1.3803^n)$.

Dokaz. Uporabimo tehniko za odstranjevanje vozlišč grafa v obratnem vrstnem redu. Naj bo G graf z n točkami in m povezavami in naj bo $v \in V$ točka z minimalno stopnjo. Če je $\delta(G) \geq n-3$, potem grafu manjkajo paroma disjunktni cikli in poti za polnost. V tem primeru je lahko najti globalno maksimalno kliko s časovno zahtevnostjo $O(n+m)$. Predpostavimo, da obstaja točka s stopnjo $d_G \leq n-4$. Vsaka maksimalna klika bodisi vsebuje v, bodisi ne in ustrezno je globalno maksimalna klika v G bodisi $\{v\}$ povezana z globalno maksimalno kliko dobljenega podgrafa $G[N(v)]$, bodisi z globalno maksimalno kliko podgrafa $G[V - \{v\}]$. Rekurzivno dobimo globalno maksimalno kliko v obeh podgrafih in iz njiju izpeljemo rešitev za G. Čas je v najslabšem primeru odvisen od neenačbe:

$$T(n) \leq T(n-4) + T(n-1) + c \cdot (n+m) \quad \text{za nek} \quad c > 0$$

Z uporabo standardnih tehnik, ki so osnovane na rodovnih funkcijah, izračunamo, da je $T(n)$ znotraj polinomskega faktorja β^n, kjer je $\beta \approx 1.3803$ največja realna ničla polinoma $\beta^4 - \beta^3 - 1$.

□

Algoritem v izreku zajame bistvo vrste hitrih algoritmov z eksponentno časovno zahtevnostjo za problem iskanja globalno maksimalne klike. Začelo se je z algoritmom s časovno zahtevnostjo $O^*(1.286^n)$, ki v bistvu sledi ideji zgornjega algoritma. Algoritem je bil pozneje izpopolnjen do časovne zahtevnosti $O^*(1.2599^n)$ [53] z uporabo analize primera okolice točk z nizko stopnjo. Časovna zahtevnost algoritma je bila še nadaljnje izboljšana v $O^*(1.2108^n)$ [54], vendar slednji algoritem potrebuje eksponentni prostor. Temu pa se lahko izognemo z algoritmom v polinomskem prostoru, z malce šibkejšo časovno zahtevnostjo $O^*(1.2227^n)$ [55].

4.1.3 Aproksimiranje velikosti maksimalnih klik

Ker ne znamo najti globalno maksimalnih klik v zmernem času, se lahko vprašamo, do katere velikosti lahko prepoznamo kliko v opravičljivem času. Velikost največje klike v grafu G označimo z $\omega(G)$. Pravimo, da algoritem aproksimira $\omega(G)$ s pomočjo faktorja $f(n)$ natanko tedaj, ko algoritem pri danem vhodnem grafu G vrne kliko U iz G tako, da je $\omega(G) \leq f(n) \cdot |U|$. Globalno maksimalno kliko sestavlja največ n točk, tako da jo lahko s faktorjem $O(n)$ aproksimiramo tako, da vrnemo povezavo, če ta obstaja v grafu.

Izrek 4.7 *Obstaja algoritem s polinomsko časovno zahtevnostjo, ki vrne za graf G z n točkami kliko s faktorjem $O\left(\frac{n}{\log n^2}\right)$ za velikost $\omega(G)$.*

Izrek 4.8 *Če ne drži $\mathcal{NP} = \mathcal{ZPP}$ ne obstaja algoritem s polinomsko časovno zahtevnostjo, ki za graf G z n točkami vrne kliko s faktorjem $n^{1-\varepsilon}$ za velikost $\omega(G)$ za vsak $\varepsilon > 0$.*

Teoretična predpostavka o kompleksnosti uporabljena v izreku, je skoraj tako močna kot $\mathcal{P} = \mathcal{NP}$. Neaproksimativen rezultat se je poostril najprej v $O(\frac{1}{\sqrt{\log \log n}})$ in kasneje v $O(\frac{1}{(\log n)^\gamma})$ za neko $\gamma > 0$. Ti rezultati so osnovani na močnejših predpostavkah o polnosti, v bistvu, da je lahko $\mathcal{NP}$-poln problem rešljiv z randomiziranimi algoritmi s kvazi-polinomsko časovno zahtevnostjo $2^{(\log n)^{O(1)}}$. Razmerje $\frac{n}{(\log n)^2}$ lahko izrazimo z $\Omega(\frac{\log \log n}{\log n})$. Spodnja in zgornja meja za aproksimacijo sta si zelo blizu.

V poljubnem grafu vemo, da je $\omega(G)$ bodisi $(2 + o(1)) \log n$ zaokrožen navzgor, bodisi navzdol za graf velikosti n. Več algoritmov s polinomsko časovno zahtevnostjo tvori klike velikosti $(1 + o(1)) \log n$, to pomeni da dosegajo aproksimativno razmerje približno dva. Domneva je, da ne obstaja algoritem s polinomsko časovno zahtevnostjo, ki ne bi vrnil klike velikosti vsaj $(1 + \varepsilon) \log n$ za $\varepsilon > 0$ [52].

4.1.4 Iskanje klik s fiksno velikostjo

V mnogo primerih bi bilo bolje iskati klike z omejeno velikostjo. Tehnično to pomeni, da velikost klike ni del vhodnih podatkov. Exhaustive search algoritem ima naprimer časovno zahtevnost $\Theta(n^k)$, kadar je velikost klike k fiksna. S trikom pridobimo algoritem za iskanje klik velikosti tri, ki ima hitrejšo časovno zahtevnost kot $O(n^3)$.

Izrek 4.9 *Obstaja algoritem za iskanje trikotnikov v grafu s časovno zahtevnostjo $O(n^{2.376})$.*

Dokaz. Naj bo G graf z n točkami. Z $A(G)$ označimo matriko sosednosti grafa G. To pomeni, da je element a_{ij} matrike $A(G)$ enak ena, če sta točki v_i in v_j sosednji in če nista je enak nič. Matrika $A(G)^2 = A(G) \cdot A(G)$ je dobljena s standardnim matričnim množenjem. Element b_{ij} matrike $A(G)^2$ je ravno število sprehodov dolžine dva med v_i in v_j. Predpostavimo, da obstaja $b_{ij} > 1$. To pomeni, da obstaja vsaj ena točka $u \in V$ različna od v_i in v_j, ki je sosednja z obema. Če ima graf G povezavo $\{v_i, v_j\}$, potem vemo, da vsebuje trikotnik $\{v_i, v_j, u\}$. Algoritem za iskanje trikotnikov preprosto izračuna $A(G)^2$ in preveri ali obstaja povezava v_i in v_j za nek neničeln element b_{ij} iz $A(G)^2$. Ker ima matrično množenje za pridobitev kvadratne matrike časovno zahtevnost $O(n^\alpha)$, kjer je $\alpha < 2.376$ [58], je časovna zahtevnost tega algoritma enaka $O(n^{2.376})$.

□

Za redke grafe obstaja celo algoritem za iskanje trikotnikov z uporabo iste tehnike, ki je hitrejši in ima časovno zahtevnost $O(m^{\frac{2\alpha}{\alpha+1}}) = O(m^{1.41})$.

Sedaj bi radi uporabili tehnike z matričnim množenjem in poiskali algoritem za iskanje klik z večjo velikostjo kot tri.

Izrek 4.10 *Za vsak $k \geq 3$ obstaja algoritem za iskanje klike velikosti k v grafu z n točkami s časovno zahtevnostjo $O(n^{\beta(k)})$, kjer je $\beta(k) = (\lfloor \frac{k}{3} \rfloor, \lceil \frac{(k-1)}{3} \rceil, \lceil \frac{k}{3} \rceil)$ in kjer ima množenje $n^r \times n^s$ matrike z $n^s \times n^t$ matriko časovno zahtevnost $O(n^{\alpha(r,s,t)})$.*

Dokaz. S k_1 označimo $\lfloor \frac{k}{3} \rfloor$, s k_2 označimo $\lceil \frac{(k-1)}{3} \rceil$ in s k_3 označimo $\lceil \frac{k}{3} \rceil$ in velja, da je $k = k_1 + k_2 + k_3$. G naj bo graf z n točkami in m povezavami. Najprej skonstruiramo trodelen pomožni graf $\bar{G}$, tako da množico V razdelimo na tri podmnožice $\bar{V_1}, \bar{V_2}$ in $\bar{V_3}$, kjer je $\bar{V_i}$ sestavljena iz vseh klik velikosti k_i v G. Definiramo dve točki $U \in \bar{V_i}$ in $U' \in \bar{V_j}$, ki sta sosednji v $\bar{G}$ natanko tedaj, ko je $i \neq j$ in je $U \cup U'$ klika velikosti $k_i + k_j$ v grafu G. Algoritem sedaj išče, če je v pomožnem grafu $\bar{G}$ kaj trikotnikov in če je v pomožnem grafu trikotnik $\{U_1, U_2, U_3\}$, potem zaradi sestave $\bar{G}$ sledi, da je $U_1 \cup U_2 \cup U_3$ klika velikosti k v grafu G. Iskanje trikotnikov v grafu $\bar{G}$ poteka s pomočjo matričnega množenja, opisanega v izreku 4.9. Potem moramo množiti $n^{k_1} \times n^{k_2}$ matriko sosednosti, ki predstavlja povezave med $\bar{V_1}$ in $\bar{V_2}$, z $n^{k_2} \times n^{k_3}$ matriko sosednosti, ki predstavlja povezave med $\bar{V_2}$ in $\bar{V_3}$. Ta korak ima časovno zahtevnost $O(n^{\beta(k)})$. Tvorjenje treh matrik ima v najslabšem primeru časovno zahtevnost $O(n^{\max k_1+k_2, k_1+k_3, k_2+k_3}) = O(n^{\lceil \frac{2k}{3} \rceil})$, ki jo asimptotsko dominira čas za hitro pravokotno matrično množenje.

□

Tabela nam kaže kaj pridobimo z uporabo matričnega množenja.

Velikost klike	**Exhaustive search**	**Matrično množenje**
3	$O(n^3)$	$O(n^{2.376})$
4	$O(n^4)$	$O(n^{3.376})$
5	$O(n^5)$	$O(n^{4.220})$
6	$O(n^6)$	$O(n^{4.751})$
7	$O(n^7)$	$O(n^{5.751})$
8	$O(n^8)$	$O(n^{6.595})$

Izrek 4.11 *Za vsak $k \geq 3$ obstaja algoritem, ki šteje koliko je klik velikosti k katerim pripada vsaka točka iz grafa na n točkah, s časovno zahtevnostjo $O(n^{\beta(k)})$, kjer je $\beta(k)$ ista funkcija, kot v izreku 4.10.*

Dokaz. Izrek je osnovan na opazovanju, da za primer $k = 3$, ni le lahko preveriti ali dve točki v_i in v_j pripadata istemu trikotniku v G, temveč tudi prešteti v koliko trikotnikih ležita: če povezava $\{v_i, v_j\}$ obstaja v G, potem je število kar element b_{ij} v kvadratu matrike sosednosti $A(G)$. V splošnem uporabimo to opazovanje na pomožnem grafu $\bar{G}$. Za vsako točko $v \in V$, naj $C_k(v)$ označuje število različnih klik v G velikosti k, ki

vsebujejo v. Podobno naj $\bar{C}_3(U)$ označuje število trikotnikov, katerim pripada vožlišče U iz G. Opazimo, da je U klika v G, ki je manjša od k. Očitno imajo klike grafa G več oblik v grafu $\bar{G}$. Točno število je $\binom{k}{k_1,k_2,k_3}$, kjer so k_1, k_2, k_3 definirane tako, kot v prejšnjem dokazu. Brez škode za splošnost naj bo k_1 najmanjši od teh treh parametrov. Naj bo $\mathcal{U}(v)$ množica vseh takih klik iz G velikosti k_1, da je $v \in U$. Dobimo naslednjo enačbo:

$$\sum_{U \in \mathcal{U}(v)} \bar{C}_3(U) = \binom{(k-1)}{(k_1-1), k_2, k_3} \cdot C_k(v)$$

Z uporabo Izreka 6.1.10 je leva stran enačbe izračunana s časovno zahtenostjo $O(n^{\beta(k)})$. Sedaj lahko iz enačbe izračunamo $C_k(v)$.

□

4.1.5 Štetje maksimalnih klik

Tradicija algoritmov za iskanje maksimalnih klik v danem grafu G je precej dolga, saj obstaja veliko ralzičnih algoritmov za štetje maksimalnih klik. Prvi algoritmi so se pojavili že leta 1957. Želimo si, da bi bil naš algoritem čim bolj učinkovit. Tipična zahteva, da je nek algoritem za štetje učinkovit, je *skupni polinomski čas*. To pomeni, da algoritem vrne vseh C možnih konfiguracij v času, ki je omejen s polinomom v spremenljivki C, velikost vhodnih podatkov pa je enaka n. Prej omenjeni algoritem "Exhaustive search"ni algoritem s polinomsko časovno zahtevnostjo. Algoritem za štetje maksimalnih klik v danem grafu G s polinomsko časovno zahtevnostjo ne obstaja, razen v primeru, ko velja $\mathcal{P} = \mathcal{NP}$ [71]. V nadaljevanju bo predstavljen eden izmed algoritmov za štetje maksimalnih klik v danem grafu G s skupnim polinomskim časom, ki ima še nekaj drugih lepih lastnosti.

Ena izmed takih lastnosti je *Polynomial delay - Polinomska zakasnitev*. Algoritem, ki izpolnjuje to lastnost generira konfiguracije eno za drugo v nekem vrstnem redu tako, da je časovni zamik od pričetka delovanja algoritma do prve konfiguracije, časovni zamik med katerimakoli dvema nadaljnjima konfiguracijama in časovni zamik med zadnjo vrnjeno konfiguracijo algoritma do časa, ko se algoritem ustavi, omejen s polinomom. Za štetje maksimalnih klik danega grafa G obstajajo algoritmi, ki izpolnjujejo zgornjo lastnost in zavzamejo linearno količino prostora [84]. O tem govori tudi naslednji izrek.

Izrek 4.12 *Za štetje vseh maksimalnih klik danega grafa G obstaja algoritem, ki ima lastnost polinomske zakasnitve $O(n^3)$ in zavzame zgolj $O(n+m)$ prostora.*

Dokaz. Konstruirajmo binarno drevo z n nivoji. To drevo naj ima liste samo na zadnjem n-tem nivoju. Vsak nivo je povezan s točko in sicer na i-tem nivoju bomo to točko označili z v_i. Vozlišča drevesa na i-tem nivoju so maksimalne klike grafa $G[\{v_1, \ldots, v_i\}]$. Od tod sledi, da so listi na n-tem nivoju maksimalne klike grafa G. Fiksirajmo nivo i in maksimalno kliko U grafa $G[\{v_1, \ldots, v_i\}]$. Poskusimo ugotoviti, kaj so sinovi točk iz množice U na nivoju $i+1$. Ločimo dva glavna primera:

1. Recimo, da so vse točke iz maksimalne klike U sosednje s točko v_{i+1} v grafu G. Potem je $U \cup v_{i+1}$ maksimalna klika grafa $G[\{v_1, \ldots, v_i, v_{i+1}\}]$. To je edini način, na katerega dobimo maksimalno kliko grafa $G[\{v_1, \ldots, v_i, v_{i+1}\}]$ tako, da ta vsebuje množico točk U. V tem primeru ima U samo enega sina v tem binarnem drevesu.

2. Recimo, da nobena točka iz množice točk U ni povezana s točko v_{i+1}. V tem primeru množica U ostane maksimalna klika grafa $G[\{v_1, v_2, \ldots, v_i, v_{i+1}\}]$ in množica U nima nobenega sina v tem binarnem drevesu.

3. Recimo, da obstaja točka v maksimalni kliki U, ki ni sosednja s točko v_{i+1} grafa G. V tem primeru lahko maksimalno kliko grafa $G[\{v_1, \ldots, v_i, v_{i+1}\}]$ dobimo na dva različna načina:

   - Maksimalna klika grafa $G[\{v_1, \ldots, v_i, v_{i+1}\}]$ je kar množica U sama.
   - Druga klika pa je enaka $\big(U - \overline{N}(v_{i+1})\big) \cup \{v_{i+1}\}$, kjer je $\overline{N}(v_{i+1})$ množica vseh točk grafa G, ki niso sosednje s točko v_{i+1}.

   V primeru, da je druga množica maksimalna klika, potem ima U dva sinova. Množica točk $\big(U - \overline{N}(v_{i+1})\big) \cup \{v_{i+1}\}$ je potencialno lahko sin različnih množic. V primeru, da je ta množica maksimalna bomo rekli, da je sin leksikografsko najmanjše množice U.

Glede na zgornjo ugotovitev imamo drevo, pri katerem imajo vsa notranja vozlišča enega ali dva sinova. Zato se drevo tudi imenuje binarno drevo, vsi listi pa se nahajajo na n-tem nivoju.
Naš algoritem štetja se tako poenostavi v pregled grafa v globino, ki kot rezultat vrne liste na n-tem nivoju. Za dano množico vozlišč U binarnega drevesa na nivoju i moramo tako premisliti samo še naslednje:

- *Oče (U,i).* Glede na definicijo binarnega drevesa, očetje množice točk U tvorijo leksikografsko najmanjšo maksimalno kliko grafa $G[\{v_1, \ldots, v_{i-1}\}]$, ki vsebuje kliko $U - \{v_i\}$. Ta izračun je učinkovit, saj je množica dobljena v času $O(n + m)$.

- *Levi sin (U,i).* Če je $U \subseteq N(v_{i+1})$, potem je množica levih sinov enaka $U \cup \{v_{i+1}\}$. Če pa velja $U \nsubseteq N(v_{i+1})$, potem je množica levih sinov enaka kar U. Preverjanje, ali gre za prvi ali za drugi primer, vzame $O(n + m)$ časa.

- *Desni sin (U,i).* Če je $U \subseteq N(v_{i+1})$, potem množica desnih sinov ni definirana. Če pa velja $U \nsubseteq N(v_{i+1})$, potem je množica desnih sinov za U enaka $\big(U - \overline{N}(v_{i+1})\big) \cup \{v_{i+1}\}$, ampak samo v primeru, če je ta množica maksimalna klika. V tem primeru velja tudi
$$U = \text{Oče}\Big(\big(U - \overline{N}(v_{i+1})\big) \bigcup \{v_{i+1}\}, i + 1\Big) .$$
   Če pa množica $\big(U - \overline{N}(v_{i+1})\big) \cup \{v_{i+1}\}$ ni maksimalna klika, potem pa množica desnih sosedov ni definirana. Tudi v tem primeru algoritem potrebuje zgolj $O(n+m)$ časa.

Najdaljšo pot med katerimakoli dvema listoma drevesa sestavlja $2n-2$ povezav in $2n-1$ vozlišč. Za vsako vozlišče potrebujemo $O(n + m)$ časa. Ker ima vsako poddrevo našega drevesa list na nivoju n pomeni, da je zamik med dvema zaporednima rezultatoma enak $O(n^3)$. Opazimo, da mora algoritem pri pregledovanju vozlišč, množice U in nivoja i določiti in shraniti zgolj to, ali gre za levega ali desnega sina. Za to pa potrebuje $O(n+m)$ prostora.

□

V dokazu smo omenili leksikografsko urejeno množico. Definicija te urejenosti je naslednja.

Definicija 4.13 Naj bo $(M, \preceq)$ linearna urejenost. Leksikografsko urejenost $\leq_{lex}$ na množici $M \times M$ definiramo takole

$$(m_1, n_1) \preceq_{lex} (m_2, n_2) \equiv m_1 \prec m_2 \vee (m_1 = m_2 \wedge n_1 \preceq n_2) \ .$$

Predpisan vrstni red

Nekoliko težji problem je generiranje maksimalnih klik danega grafa G v določenem vrstnem redu, kot je na primer leksikografsko zaporedje. Če vztrajamo, da mora imeti algoritem polinomsko časovno zahtevnost, potem to ni neka dodatna omejitev, saj moramo le zbrati vse rezultate, ki jih algoritem vrne in jih razvrstiti v leksikografskem vrstnem redu. Vključevanje vrstnega reda je smiselno samo pri algoritmih, ki izpolnjujejo prej omenjeno lastnost "polynomial delay". Algoritem s to lastnostjo, ki temelji na DFS-algoritmu (Depth First Search - Pregled grafa v globino [72]) iz izreka 7.16, ne vrne rezultatov v leksikografskem vrstnem redu.

Izrek 4.14 *Naj bo G poljuben graf in U maksimalna klika tega grafa. Ugotavljanje, ali v danem grafu obstaja maksimalna klika U' , ki je leksikografsko večja od maksimalne klike U, je $\mathcal{NP}$- poln problem.*

Posledica 4.15

1. *Če ne drži $\mathcal{P} = \mathcal{NP}$, ne obstaja algoritem, ki bi za dani graf G in poljubno maksimalno kliko U tega grafa, generiral v leksikografskem vrstnem redu naslednjo maksimalno kliko v polinomskem času.*

2. *Če ne drži $\mathcal{P} = \mathcal{NP}$, ne obstaja algoritem, ki bi za poljuben graf G generiral vse maksimalne klike v obratnem leksikografskem vrstnem redu s polinomsko časovno zahtevnostjo.*

Mogoče se zdi presenetljivo, da sploh obstajajo algoritmi, ki generirajo vse maksimalne klike danega grafa G v leksikografskem vrstnem redu s polinomsko časovno zahtevnostjo. Ideja teh algoritmov je naslednja: med izračunavanjem trenutnega rezultata, dodaten čas vlagamo v iskanje leksikografsko večje maksimalne klike. Te klike shranjujemo v tako imenovano prednostno vrsto Q. Ta prednostna vrsta Q lahko potencialno vsebuje eksponentno število klik in zasede eksponento količino prostora. V nadaljevanju bomo predstavili algoritem za štetje maksimalnih klik danega grafa G v leksikografskem vrstnem redu, ki pri delovanju uporablja drevesno strukutro, ki je bila opisana v izreku 7.16. Algoritem je natančno predstavljen v [73].

Izrek 4.16 *Algoritem 5 prešteje vse maksimalne klike danega grafa G z n točkami v leksikografskem vrstnem redu. Časovna zahtevnost algoritma je enaka $O(n^3)$.*

Dokaz. Najprej opazimo, da je množica T dodana v prednostno vrsto Q, ko je leksikografsko večja od množice U. Tako v prednostno vrsto Q shranimo zgolj tiste množice, ki jih mora algoritem vrniti za množico U. Tako je zaporedje maksimalnih klik, ki ga

Algorithm 1 Leksikografsko štetje maksimalnih klik [73]

Vhod: Graf $G = (V, E)$
Izhod: Zaporedje maksimalnih klik grafa G v leksikografskem vrstnem redu.
Naj bo U_0 leksikografsko prva maksimalna klika
Dodaj U_0 v prednostno vrsto Q;
while Q ni prazna množica **do**
U:= minimum(Q);
output U;
for točko v_j grafa G, ki ni sosednja s točko $v_i \in U$ tako, da je $i < j$ **do**
$U_j := U \cup \{v_1, \ldots, v_j\}$;

if $\left(U_j - \overline{N}(v_j)\right) \cup \{v_j\}$ je maksimalna klika grafa $G[\{v_1, \ldots, v_j\}]$ **then**
Naj bo T leksikografsko najmanjša maksimalna klika, ki vsebuje $\left(U_j - \overline{N}(v_j)\right) \cup \{v_j\}$;

Dodaj T v Q
end if
end for
end while

dobimo na koncu, res leksikografsko naraščajoče. Pokazati moramo, da so v dobljenem zaporedju res vse maksimalne klike. To bomo dokazali z indukcijo: če je U leksikografsko prva maksimalna klika, ki je algoritem še ni vrnil, potem se množica U nahaja v prednostni vrsti Q.

Baza indukcije: Recimo, da je $U = U_0$. Potem zgornja trditev očitno drži.
Indukcijski korak: Recimo, da je U leksikografsko večja od množice U_0. Naj bo j največji indeks, za katerega velja, da $U_j = U \cap \{v_1, \ldots, v_j\}$ ni maksimalna klika v grafu skrčenem na točke $v_1, \ldots, v_j$. Tak indeks mora obstajati, saj bi sicer veljalo $U = U_0$. V resnici velja še več. In sicer $j < n$, ker je U maksimalna klika celotnega grafa G. Zaradi maksimalnosti indeksa j mora veljati naslednje: $v_{j+1} \in U$. Potem obstaja neprazna množica S tako, da je $U_j \cup S$ maksimalna klika grafa $G[\{v_1, \ldots, v_j\}]$. Ponovno zaradi maksimalnosti indeksa j velja tudi, da točka v_{j+1} ni sosednja z vsemi točkami iz množice S. Sklepamo, da obstaja maksimalna klika U', ki vsebuje $U_j \cup S$, ampak ne vsebuje točke v_{j+1}. Opazimo, da je U' leksikografsko manjša kot množica U, saj se razlikujeta na množici S. Po indukcijski predpostavki je algoritem množico U' že vrnil. V trenutku, ko algoritem vrne množico U' ugotovimo, da točka v_{j+1} ni sosednja z neko točko $v_i \in U'$ za indeks $i < j + 1$. Jasno je, da v tem primeru velja $\left(U'_{j+1} - \overline{N}(v_{j+1})\right) \cup \{v_{j+1}\} = U_{j+1}$ in U_{j+1} je maksimalna klika grafa $G[\{v_1, \ldots, v_{j+1}\}]$. Tako je leksikografsko prva maksimalna klika T, ki vsebuje U_{j+1} shranjena v prednostno vrsto Q. Ponovno zaradi maksimalnosti indeksa j velja, da se množici U in T ujemata na prvih $j+1$ točkah. Predpostavimo, da $U \neq T$. Naj bo k prvi tak indeks, za katerega velja, da se množici U in T razlikujeta v točki v_k. Od tod sledi, da je $k > j + 1$. Ker je T leksikografsko manjša od U velja: $v_k \in T$ in $v_k \notin U$. Zato U_k ni maksimalna klika grafa $G[\{v_1, \ldots, v_k\}]$, kar je v protislovju z maksimalnostjo indeksa j. Posledično mora veljati $U = T$ in zato $U \in Q$. To dokaže indukcijski korak.
Časovna zahtevnost algoritma: Iskanje leksikografsko najmanjše maksimalne klike v prednostni vrsti Q nas stane $O(n \log C)$ operacij. Potem sledi n izračunov ali maksimalna

klika vsebuje dano množico, za kar potrebujemo $O(n+m)$ operacij za vsako množico. Na koncu moramo upoštevati še shranjevanje maksimalne klike v prednostno vrsto Q, kar zahteva $O(n\ logC)$ za vsako kliko. Ker je $C \leq 3^{\lceil \frac{n}{3} \rceil}$, je celotna časovna zahtevnost algoritma v najslabšem primeru enaka $O(n^3)$.

□

Zahtevnost štetja maksimalnih klik v danem grafu G
Ta razdelek bomo zaključili z nekaj opombami, ki se nanašajo na zahtevnost štetja maksimalnih klik v danem grafu. Vse maksimalne klike danega grafa G dobimo tako, da za štetje uporabimo enega izmed zgoraj omenjenih algoritmov. Spremenljivko, ki nam šteje maksimalne klike, pa ob vsaki kliki, ki jo algoritem vrne, povečamo za ena. Ta postopek nam vzame eksponentno količino časa. Seveda se vprašamo, ali je število vseh maksimalnih klik danega grafa G mogoče dobiti bolj direktno in v polinomskem času. Če želimo odgovoriti na to vprašanje, moramo definirati *razred* $\#\mathcal{P}$ [82]. To je oznaka za razred vseh funkcij, ki štejejo rešitve $\mathcal{NP}$-problemov. Da se pokazati, da je problem štetja maksimalnih klik danega grafa G $\mathcal{P}$-poln problem. Posledica tega je, da če obstaja algoritem za štetje maksimalnih klik grafa s polinomsko časovno zahtevnostjo, potem je ta problem iskanja maksimalnih klik danega grafa $\mathcal{P}$-poln problem in posledično velja $\mathcal{P} = \mathcal{NP}$.

4.2 Strukturno goste množice

V tem razdelku si bomo ogledali dve variaciji problema iskanja maksimalnih klik, ki temeljita na principu minimalnih stopenj. Obe variaciji sta strukturne narave, dodatne splošne omejitve pa se nanašajo na posamezne točke v teh množicah [75, 76, 77].

4.2.1 Plexi

V tem primeru bomo posplošili koncept klik danega grafa G tako, da bomo enostavno izbrisali nekaj povezav znotraj goste množice. Vendar pa bomo lahko v grafu G odstranili največ $N \geq 1$ povezav, saj je stopnja vsake točke navzdol omejena. Tako dobljeno gosto množico bomo poimenovali N-plex danega grafa G [76, 83].

Definicija 4.17 Naj bo $G = (V, E)$ poljuben neusmerjen graf in $N \in \{1, \ldots, n-1\}$ naravno število. Podmnožici $U \subseteq V$ pravimo N-plex danega grafa G natanko tedaj, ko velja $\delta\big(G[U]\big) \geq |U| - N$.

Jasno je, da je klika 1-plex in da je N-plex grafa G hkrati tudi $(N+1)$-plex. Pravimo, da je podmnožica $U \subseteq V$ *lokalno maksimalni N-plex* danega grafa G natanko tedaj, ko je U N-plex in ko U zagotovo ni vsebovana v nobenem večjem N-plexu danega grafa G. Podmnožica $U \subseteq V$ je *globalno maksimalni N-plex* grafa G natanko tedaj, ko U vsebuje največje število točk med vsemi N-plexi grafa G.
Opazimo lahko tudi, da je vsak podgraf N-plexa prav tako N-plex, kar pomeni, da je množica N-plexov zaprta za izključevanje točk. Poznamo naslednjo zvezo med velikostjo N-plexa in njegovim premerom [76, 78, 79].

Trditev 4.18 *Naj bo $N \in \{1, \ldots, n-1\}$naravno število in naj bo $G = (V, E)$ poljuben neusmerjen graf na n-točkah.*

1. *Če je V N-plex, kjer je $N < \frac{n+2}{2}$, potem je* $\mathrm{diam}(G) \leq 2$. *V primeru, da velja tudi $n \geq 4$, potem je G 2-povezan graf.*

2. *Če je V N-plex, kjer je $N \geq \frac{n+2}{2}$ in je G povezan graf, potem je* $\mathrm{diam}(G) \leq 2N - n + 2$.

Dokaz.

1. Recimo, da je $N < \frac{n+2}{2}$. Naj bosta $u, v \in V$ dve taki točki, za kateri velja $u \neq v$. Če sta u in v sosednji točki, potem je razdalja med njima enaka ena. Sedaj predpostavimo, da u in v nista sosednji točki. Privzemimo, da je razdalja med točko u in točko v vsaj tri. To pomeni, da za sosednje točke u in v velja $N(u) \cap N(v) = \emptyset$. Dobimo naslednjo zvezo:

$$n - 2 \geq \Big|N(u) \cup N(v)\Big| \geq 2\delta(G) \geq 2(n - N) > 2\left(n - \frac{n+2}{2}\right) = n - 2 \ ,$$

kar je protislovje. Od tod sledi, da je razdalja med u in v največ dva. Zato velja $\mathrm{diam}(G) \leq 2$. Sedaj moramo pokazati še, da je za $n \geq 4$ graf G 2-povezan. Privzemimo nasprotno. Recimo, da obstaja most. To je takšna povezava e grafa G, da če to povezavo odstranimo, je graf $G - \{e\}$ sestavljen iz dveh povezanih komponent V_1 in V_2. Očitno je, da mora vsaka najkrajša pot od točke v množici V_1 do točke v množici V_2 vsebovati to povezavo e. Ker velja $\mathrm{diam}(G) \leq 2$ mora biti ena izmed komponent V_1, V_2 singelton. Od tod sledi, da ima točka v tej komponenti stopnjo ena. Glede na trditev 4.18 velja, da je V N-plex in $n \geq 4$. Za stopnjo vsake točke iz množice V velja nalsednja zveza:

$$n - N > n - \frac{n+2}{2} = \frac{n-2}{2} \geq 1 \ ,$$

kar je protislovje. Od tod sledi, da takšna povezava e v grafu G ne more obstajati.

2. Recimo, da je $N \geq \frac{n+2}{2}$. Naj bo $\{v_0, v_1, \ldots, v_r\}$ najdaljša najkrajša pot grafa G. To je pot, ki realizira premer r. Predpostavimo lahko, da je $r \geq 4$. Ker ne obstaja krajša pot med točko v_0 in točko v_r velja, da točka v_i ni sosednja s točkami $v_0, \ldots, v_{i-2}, v_{i+2}, \ldots, v_r$ za vsak $i \in \{0, \ldots r\}$. Poleg tega ne more obstajati točka, ki bi bila sosednja z obema točkama v_0 in v_3. Zato velja naslednja zveza:

$$\{v_0\} \cup \{v_2, v_3, \ldots, v_r\} \cup \Big(N(v_3) - \{v_2, v_4\}\Big) \subseteq \overline{N}(v_0) \ .$$

Opazimo, da imamo na levi strani disjunktno unijo množic. Tako dobimo naslednjo neenakost:

$$1 + (r - 1) + d_G(v_3) - 2 \leq N \ .$$

Od tod sledi $r + n - N - 2 \leq N$. Zato velja $\mathrm{diam}(G) = r \leq 2N - n + 2$.

□

Iz računskega vidika ni iskanje globalno maksimalnega plexa nič bolj enostavno kot iskanje maksimalne klike v danem grafu G. Videli smo, da je problem iskanja maksimalne klike v danem grafu G $\mathcal{NP}$-poln problem. Sedaj bomo poskusili ugotoviti, kakšna je zahtevnost

iskanja N-plexa določene velikosti za fiksno število N. Za poljubno naravno število $N > 0$ definiramo naslednji problem:

Problem: N-plex
Vhodni podatki: graf G, parameter $k \in \mathbb{N}$
Vprašanje: Ali obstaja N-plex velikosti najmanj k v grafu G?

Ker je 1-plex v danem grafu G kar klika grafa G velja, da je iskanje 1-plexa $\mathcal{NP}$-poln problem. Od tod sledi, da je iskanje globalno maksimalnega N-plexa $\mathcal{NP}$-poln problem za vsak $N > 0$.

Izrek 4.19 *Iskanje N-plexa v danem grafu G je NP-poln problem za vsa naravna števila $N > 0$.*

Dokaz. Dovolj je, če izrek 4.19 dokažemo za primer, ko je $N > 1$. Obstaja generičen dokaz izreka 4.19, ki temelji na dejstvu, da je N-plex grafa G "dedna"lastnost danega grafa [80]. Mi bomo tukaj podali direkten dokaz izreka 4.19 zato, da bomo tako pokazali strukturno podobnost med klikami in plexi. Opisali bomo polinomsko transformacijo problema iskanja klik v danem grafu G v problem iskanja N-plexa v tem grafu. Naj bo (G, k) poljuben problem iskanja klike v grafu G velikosti k. Konstruiramo nov graf G' po naslednjem postopku: vzamemo $N - 1$ kopij vsake točke grafa G in jih povežemo med sabo. Prav tako povežemo vse nove točke z obstoječimi točkami v grafu G razen z izvorno točko. Bolj natančno, naj bo $G' = (V', E')$ graf definiran kot:

$$\begin{aligned} V' &=_{\text{def}} V \times \{0, 1, \ldots, N-1\} \\ E' &=_{\text{def}} \Big\{ \{(u,0),(v,0)\} | \{u,v\} \in E \Big\} \cup \\ &\quad \cup \Big\{ \{(u,i),(v,j)\} | u, v \in V \text{ in } i, j > 0 \Big\} \cup \\ &\quad \cup \Big\{ \{(u,0),(v,i) | u, v \in V, \text{ kjer je } u \neq v \text{ in } i > 0\} \Big\} \ . \end{aligned}$$

Graf G' lahko zagotovo konstruiramo v polinomskem času. Ključna ugotovitev je, da so kopirane točke, to so točke v množici $V \times \{1, \ldots, N-1\}$, sosednje z vsemi točkami v množici V', razen z eno. Pokazali bomo, da graf G vsebuje kliko velikosti k natanko tedaj, ko graf G' vsebuje N-plex velikosti $k + (N-1)n$.
Predpostavimo, da obstaja klika $U \subseteq V$ velikosti natanko k v grafu G. Naj bo U' množica točk v grafu G', ki vsebuje vse originalne točke množice U in vse kopije točk množice V, to je $U' = U \times \{0\} \cup V \times \{1, \ldots, N-1\}$. Opazimo, da je moč množice U' enaka $k + (N-1)n$. Vsaka točka z oznako $i \in \{1, \ldots, N-1\}$ je direktno povezana z vsako drugo točko v množici U' razen z eno točko, ki ima oznako 0, zato ima ta točka stopnjo enako $|U'| - 2 = k + (N-1)n - 2$. Vsaka točka $(u, 0)$ je sosednja z vsemi točkami v množici U' razen s točko (u, i), kjer je $i > 0$. To pomeni, da ima $(u, 0)$ stopnjo $k + (N-1)n - 1 - (N-1)$. Sledi, da je U' N-plex.
Predpostavimo, da ne obstaja klika velikosti k v grafu G. Tako ima vsak induciran podgraf grafa G s k' točkami, kjer je $k' \geq k$, minimalno stopnjo enako največ $k' - 2$. Naj bo $U \subseteq V'$ poljubna množica točk s $k + (N-1)n$ točkami. Potem obstaja še druga množica $U' \subseteq V'$ s $k + (N-1)n$ točkami tako, da velja $\delta\big(G'[U']\big) \geq \delta\big(G'[U]\big)$ in U' vsebuje vse kopije točk grafa G', to je $U' \supseteq V \times \{1, \ldots, N-1\}$. To sledi iz dejstva, da vedno obstaja ena točka

v množici $U_0 = U \cap (V \times \{0\})$, ki ni sosednja z neko drugo točko v množici U_0 (sicer bi U_0 inducirala kliko velikosti $|U_0| \geq k$ v grafu G). Glede na zgornje ugotovitve lahko sedaj rekurzivno zamenjujemo te točke s točkami iz množice $V \times \{1, \ldots, N-1\}$ dokler je to mogoče, brez da bi pri tem zmanjšali minimalno stopnjo. Na koncu dobimo željeno množico $U' \subseteq V'$. Ker nimamo podane velikosti k-klike v grafu G, lahko zaključimo, da velja

$$\delta\big(G'[U]\big) \leq \delta\big(G'[U']\big) \leq k + (N-1)n - 2 - (N-1) \; .$$

Sledi, da v grafu G' ni N-plexa.

□

4.2.2 Jedra

Struktura, ki je podobna plexom v danem grafu G, so jedra tega grafa. V tem primeru se ne sprašujemo, koliko povezav manjka v podgrafu danega grafa G, da bi ta bil poln, ampak enostavno fiksiramo kolikšna mora biti minimalna stopnja vsake posamezne točke v podmnožici točk. Najpomembnejša ugotovitev tega razdelka bo, da obstajajo algoritmi s polinomsko časovno zahtevnostjo, ki v danem grafu G poiščejo globalno maksimalna jedra. Več o jedrih je predstavljeno v [75].

Definicija 4.20 Naj bo $G = (V, E)$ poljuben neusmerjen graf. Pravimo, da je podmnožica $U \subseteq V$ N-jedro natanko tedaj, ko velja $\delta\big(G[U]\big) \geq N$.

Parameter N pri N-jedru je red jedra. Podmnožica $U \subseteq V$ je *lokalno maksimalno N-jedro* natanko tedaj, ko je množica U N-jedro in ni vsebovana v nobenem večjem N-jedru grafa G. Podmnožica $U \subseteq V$ je *globalno maksimalno N-jedro* natanko tedaj, ko vsebuje največje število točk med vsemi N-jedri grafa G. Globalno maksimalna N-jedra se imenujejo tudi *glavna jedra*.

Vsako $(N+1)$-jedro je tudi N-jedro in vsako N-jedro je $(n-N)$-plex. Velja še več, če sta U in U' N-jedri, potem je tudi $U \cup U'$ N-jedro. To pomeni, da je glavno N-jedro eno samo. Množica N-jeder ni zaprta za izključevanje točk, prav tako pa N-jedra v splošnem niso ugnezdena. Na primer cikel je zagotovo 2-jedro. Ampak vsak pravilni podgraf ima vsaj eno točko, ki ima stopnjo manjšo od dva. N-jedra niso nujno povezana. Naslednja trditev se nanaša na lokalno maksimalna povezana N-jedra.

Trditev 4.21 *Naj bo $G = (V, E)$ poljuben neusmerjen graf in naj bo $N > 0$ poljubno naravno število. U in U' naj bosta lokalno maksimalni, povezani N-jedri v grafu G tako, da velja $U \neq U'$. Potem v grafu G ne obstaja povezava med U in U'.*

Dokaz. Privzemimo, da obstaja povezava $\{u, v\}$, kjer je $u \in U$ in $v \in U'$. Potem sledi, da je $U \cup U'$ N-jedro, ki vsebuje obe množici U in U'. Še več, to je celo povezano N-jedro, ker sta U in U' povezana.

□

Sedaj si poglejmo nekaj neposrednih posledic trditve 4.21:

- Edino globalno maksimalno N-jedro danega grafa G je unija vseh njegovih lokalno maksimalnih povezanih N-jeder.

- Globalno maksimalno 2-jedro povezanega grafa je povezano (opazimo, da imajo notranje točke poti stopnjo dva).
- Graf je gozd natanko tedaj, ko ne vsebuje nobenega 2-jedra.

Naslednja trditev opisuje pomembno algoritmično lastnost N-jeder.

Trditev 4.22 *Naj bo $G = (V, E)$ poljuben neusmerjen graf in naj bo $N > 0$ poljubno naravno število. Če rekurzivno odstranjujemo vse točke s stopnjo strogo manjšo od N in vse povezave, ki so povezane z njimi, potem preostala množica točk U tvori globalno maksimalno N-jedro.*

Dokaz. Jasno je, da je U N-jedro. Pokazati moramo, da je globalno maksimalno. Privzemimo nasprotno. Recimo, da N-jedro U, ki ga dobimo, ni globalno maksimalno. Potem obstaja neprazna množica $T \subseteq V$ tako, da je množica $U \cup T$ globalno maksimalno N-jedro. Ampak točke iz množice T so bile odstranjene. Recimo, da je točka t prva točka iz množice T, ki je bila odstranjena. V tem trenutku je morala biti stopnja točke t strogo manjša od N. Kakorkoli, če ima točka t vsaj N sosedov v množici $U \cup T$ in se vse ostale točke še vedno nahajajo v grafu G, ko je točka t odstranjena, dobimo protislovje.

□

Na osnovi lastnosti opisane v trditvi 4.22 lahko konstruiramo algoritem, ki nam bo za dani graf G poiskal N-jedra. Točki $v \in V$ določimo številko jedra tako, da je ta enaka največjemu redu N med vsemi globalno maksimalnimi N-jedri, ki jim točka v pripada. To pomeni, da velja naslednje:

$$\xi_G(v) =_{\text{def}} \max\{N| \text{ obstaja } N\text{-jedro } U \text{ v grafu } G \text{ tako, da velja } v \in U\} .$$

Metoda, ki izračuna vsa jedrna števila, je predstavljena v spodnjem algoritmu. Algoritem je pravilen zaradi naslednjih razlogov:

- Poljuben graf G je zagotovo $\delta(G)$-jedro.
- Vsaka sosednja točka točke v ima nižjo stopnjo kot točka v, od koder sledi potencialno jedrno število za točko v.

Najbolj enostavne izvedbe algoritma imajo v najslabšem primeru časovno zahtevnost enako $O(mn \log(n))$. Največ korakov porabimo pri sortiranju točk glede na njihove stopnje. Spretnejše različice algoritma imajo linearno časovno zahtevnost [81].

Izrek 4.23 *Obstaja izvedba algoritma 6, ki nam izračuna jedrna števila vseh točk v danem grafu $G = (V, E)$, ki ima n točk in m povezav, v času $O(n + m)$.*

Dokaz. Če želimo zmanjšati čas delovanja algoritma, moramo pospešiti operacije, ki so povezane s sortiranjem točk v algoritmu. To lahko dosežemo na dva načina:

Algorithm 2 Izračun jedrnih števil [81]

Vhod: Graf $G = (V, E)$
Izhod: Vektor ξ_G, ki vsebuje jedrna števila vseh točk grafa G.
Izračunamo stopnje vseh točk in jih shranimo v D;
Množico točk V sortiramo v naraščajočem vrstnem redu stopenj iz množice D;
for $v \in V$ v sortiranem vrstnem redu **do**
$\xi_G(v)$:=D[v];
for točko u, ki je sosednja s točko v **do**
if $D[u] > D[v]$ **then**
$D[u] := D[u] - 1$;
Presortiramo množico V v naraščajočem vrstnem redu stopenj iz množice D.
end if
end for
end for

1. Ker stopnja točke leži na območju $\{0, \ldots, n-1\}$, pri sortiranju porabimo n predalčkov, enega za vsako stopnjo točke. Tako porabimo $O(n)$ časa za začetno sortiranje množice točk V.

2. Lahko pa v celoti shranjujemo ta presortiranja množice točk V. To naredimo tako, da poskušamo ohranjati informacije o tem, kje v območju V, ki je sestavljeno iz točk v naraščajočem vrstnem redu njihovih stopenj, se prične nova regija z višjimi stopnjami. Bolj natančno, mi vzdržujemo neko območje J, kjer je vstop $J[i]$ enak minimalnemu indeksu j, za katerega velja, da ima za vsak $r \geq j$, točka $V[r]$ stopnjo enako vsaj i. Sedaj lahko **if** stavek v algoritmu 6 nadomestimo z naslednjim ukazom:

   if $u \neq$ točka w na poziciji $J[D[u]+1]$ **then** zamenjamo točki u in w v množici V;
   Povečamo $J[D[u]+1]$;

   Sortiranje množice V, z namenom ohraniti naraščajoči vrstni red stopenj točk, sedaj zavzame $O(1)$ časa. Opazimo lahko, da se da začetno območje J izračunati v času $O(n)$.

Za celotno delovanje algoritma 6 sedaj potrebujemo naslednje število korakov: $O(n)$ korakov porabimo za inicializacijo in sortiranje, $O(m)$ korakov porabimo za glavni del algoritma (ker vsako povezavo obravnavamo največ dvakrat). Tako smo dokazali, da časovna zahtevnost algoritma znaša $O(n+m)$.

□

Posledica 4.24 *Za vsak $N > 0$ lahko globalno maksimalno N-jedro za graf z n točkami in m povezavami izračunamo v času $O(n+m)$, kar je neodvisno od N.*

4.3 Statistično goste grupe

Statistične mere v sploĽ˘nem ne vpeljujejo kakšnih univerzalnih strukturnih zahtev. To

jih dela bolj fleksibilne od strukturnih mer, hkrati pa je s statističnimi običajno grafe težje analizirati. Vseeno se bomo posvetili prav slednjim.

4.3.1 Gosti podgrafi

Naravna oznaka za gostoto grafa je naslednja. Naj bo $G = (V, E)$ nek neusmerjen graf z n točkami in m povezavami. Gostota grafa G, $\varrho(G)$, predstavlja razmerje, definirano kot:

$$\varrho(G) =_{def} \frac{m}{\binom{n}{2}}$$

Gostota grafa je torej odstotek števila povezav. Število obstoječih, ulomljeno s številom vseh možnih. Zanimali nas bodo predvsem podgrafi z določenimi gostotami.

Definicija 4.25 Naj bo $G = (V, E)$ neusmerjen graf in naj bo $0 \leq \eta \leq 1$ realno število. Podmnožica $U \subseteq V$ naj predstavlja η-gost podgraf, natanko tedaj, ko $\varrho(G[U]) \geq \eta$.

V η-gostem podgrafu sta poljubni dve točki povezani z verjetnostjo najmanj η. Kljub temu pa je možno, da imajo celo grafi s precej veliko gostoto izolirane točke.

Klika, podgraf z veliko gostoto, je 1-gost podgraf. N-plex ima gostoto $1 - \frac{N-1}{n-1}$. Sledi, da se gostota N-plexa bliža 1, ko gre n proti neskončno. Malce natančneje, za vsak $N > 0$ in za vsak $0 \leq \eta \leq 1$, je $N - plex$ velikosti najmanj $\frac{N-\eta}{n-\eta}$ je η-gost podgraf. Toda očitno je, da ni vsak $(1 - \frac{N-1}{n-1})$-gost podgraf (kadar dovolimo nekonstantne gostote) hkrati tudi N-plex. $N - jedro$ je $\frac{N}{n-1}$-gost podgraf, ki pa ima lahko gostoto poljubno blizu 0 za velike n.

Trditev 4.26 *Naj bo $0 \leq \eta \leq 1$ realno število. η-gost graf G velikosti k vsebuje η-gost podgraf velikosti $k - 1$.*

Dokaz. Naj bo U nek η-gost podgraf grafa G, $|U| = k$. Naj m_U označuje število povezav v $G[U]$. Naj bo v točka z minimalno stopnjo v grafu $G[U]$. Zapišimo, da: $\varrho(G[U]) \leq \bar{d}(G[U]) = \frac{2m_U}{k} = \varrho(G[U])(k-1)$. Z odstranitvijo točke v iz množice točk U dobimo podmnožico U'. Označimo z $m_{U'}$ število povezav U'. Dobimo:

$$m_{U'} = m_U - \varrho(G[U]) \geq \varrho(G[U])\binom{k}{2} - \varrho(G[U])(k-1) = \varrho(G[U])\frac{k-1}{2}.$$

□

Zaradi tega: $\varrho(G[U']) \geq \varrho(G[U]) \geq \eta$. Zato je U' η-gost podgraf.

Trditev nam torej predlaga požresen pristop za iskanje η-gostih grafov: rekurzivno odstranjevanje točk z minimalno stopnjo, dokler ne ostane le še η-gost podgraf. Kakor koli, ta postopek nam lahko tudi zelo spodleti. O tem bomo razpravljali v nadaljevanju.

Sprehodi. Povprečne gostote na povezavah podgrafov. Povezava je sprehod dolžine 1.

Posplošitev gostote lahko vključuje tudi sprehode večjih dolžin. Za večjo natančnost vpeljemo nekaj oznak. Naj bo $G = (V, E)$ nek neusmerjen graf z n točkami. Naj bo $l \in \mathbb{N}$ dolžina sprehoda. Za točko $v \in V$ definiramo njeno stopnjo reda l v G kot število sprehodov dolžine l, ki se začnejo v točki v. Naj bo $d^l_G(v)$ stopnja točke reda l v grafu G. Določimo $d^0_G(v) = 1$ za vse $v \in V$. Jasno je, da je $d^1_G(v)$ stopnja točke v v grafu G. Število sprehodov dolžine l v grafu G je označeno z $W_l(G)$. Dobimo sledečo relacijo med stopnjo višjega reda in številom sprehodov v grafu.

Trditev 4.27 *Naj bo $G = (V, E)$ neusmerjen graf. Za vse $l \in \mathbb{N}$ in vse $r \in \{0, 1, ..., l\}$, $W_l(G) = \sum_{v \in V} d^r_G(v) \cdot d^{l-r}_G(v)$.*

Dokaz. Vsak sprehod dolžine l vsebuje točke $v_0, v_1, ..., v_l$. Določimo poljuben $r \in \{0, 1, ..., l\}$. Obravnavamo točko v_r. Potem sprehod $v_0, v_1, ..., v_r$ prispeva stopnjo reda r točke v_r in sprehod $v_r, v_{r+1}, ..., v_l$ prispeva stopnjo reda $l-r$ točke v_r. Zatorej dobimo $d^r_G(v_r) \cdot d^{l-r}_G(v_r)$ sprehode dolžine l, ki vsebujejo točko v_r na poziciji r. Če preštejemo vse možne izbire za točko na poziciji r, lahko potrdimo, da trditev res velja.

□

Jasno je, da je največje število sprehodov dolžine l v grafu z n točkami $n(n-1)^l$. Zato lahko definiramo gostoto reda l grafa G na naslednji način:

$$\varrho_l(G) =_{def} \frac{W_l(G)}{n(n-1)^l}$$

Zapišimo, da $\varrho_1(G) = \varrho(G)$ kot v $W_1(G)$, kjer vsako povezavo štejemo dvakrat. Sedaj enostavno zaključimo naslednjo trditev.

Trditev 4.28 *Drži, da $\varrho_l(G) \leq \varrho_{l-1}(G)$ za vsak graf G in vsa naravna števila $l \geq 2$.*

Dokaz. Naj bo $G = (V, E)$ neusmerjen graf z n točkami. S trditvijo 4.27,

$$W_l(G) = \sum_{v \in V} d^1_G(v) \cdot d^{l-1}_G(v) \leq (n-1) \sum_{v \in V} d^{l-1}_G(v) = (n-1) W_{l-1}(G).$$

Sedaj neenačba enastavno sledi iz tega.

□

Za graf $G = (V, E)$ lahko definiramo podmnožico $U \subseteq V$, ki predstavlja η-gost podgraf reda l, če in samo če $\varrho_l(G[U]) \geq \eta$. Iz zgornje trditve sledi, da je vsak η-gost podgraf reda l prav tako η-gost podgraf reda $l-1$. η-gost podgraf reda $l \geq 2$ prevzame lastnost vgnezdenosti od η-gostih podgrafov. Če določimo gostoto in upoštevamo gostoto podgrafaov naraščajočega reda, potem lahko opazimo, da postajajo le-ti vse bolj podobni klikam. Formalni argument je naslednji. Definiramo gostoto neskončnega reda grafa G na naslednji način:

$$\varrho_\infty(G) =_{def} \lim_{l \to \infty} \varrho_l(G.)$$

Gostota neskončnega reda nas pripelje do diskretne funkcije gostote zaradi naslednjega $0-1$ izreka [59].

Izrek 4.29 *Naj bo $G = (V, E)$ neusmerjen graf.*

1. *Drži, da $\varrho_\infty(G)$ je ali 0 ali 1,*

2. *V je klika natanko tedaj, ko je $\varrho_\infty(G) = 1$.*

Izrek pravi, da je edina podgrupa, ki je η-gosta za nek $\eta > 0$ in vse rede, klika. Na nek način red funkcij gostot dovoljuje skaliranje tega, kako pomembna je kompaktnost grup v povezavi z gostoto.

Povprečna stopnja. Lahko bi preprosto prevedli gostoto grafa z n točkami v njeno povprečno stopnjo (kot smo to naredili v dokazu trditve 4.26): $\bar{d}(G) = \varrho(G)(n-1)$. Tehnično, gostota in povprečna stopnja sta zamenljivi (s primernimi popravki, spremembami). Zato lahko definiramo gostote podgrafov alternativno s pojmom povprečnih stopenj. Naj bo $N > 0$ racionalno število. N-gost podgraf grafa $G = (V, E)$ in $U \subseteq V$ taka, da $\bar{d}(G[U]) \geq N$. Jasno je, da je η-gost podgraf (z upoštevanjem gostot v odstotkih) velikosti k $\eta(k-1)$-gost podgraf (z upoštevanjem povprečnih stopenj) in N-gost podgraf (z upoštevanjem povprečnih stopenj) velikosti k $\frac{N}{k-1}$-gost podgraf (z upoštevanjem gostot v odstotkih). Vsako N-jedro je N-gost podgraf. N-gosti podgrafi niso niti zaprti pod izključitvijo, niti niso vgnezdeni. To lahko enastavno opazimo z upoštevanjem N-regularnih grafov (za $N \in \mathbb{N}$). Odstranjevanje nekaterih točk zmanjšuje povprečno stopnjo strogo pod N. Kakorkoli, povprečne stopnje dovoljujejo več podrobno-razdrobljenih (angl. fine-grained) analiz omrežnih struktur. Ko je število povezav kvadratno v številu točk, je za vsak graf potrebno, da postane še gostejši od nekaterih danih mejnih odstotkov, zato imajo manjši grafi prednost. Povprečne stopnje preprečujejo take nevarnosti.

Ekstremni grafi. Na temelju Turan-ovega izreka (glej Izrek 4.2) se je razvilo celotno novo območje v teoriji grafov, ki se imenuje teorija ekstremnih grafov [60]. To področje preučuje vprašanja kot je slednje: koliko povezav naj ima graf tako, da nekatere podmnožice podgrafov ne bodo vsebovane v grafu? Jasno, če imamo več povezav v grafu, potem morajo biti ti podgrafi vsebovani v njej. To je bilo uvedeno tudi za goste podgrafe. Slednji klasični izrek Dirac-a [61] je direktni podkrepljeni Turan-ov izrek.

Izrek 4.30 *(Dirac, 1963). Naj bo $G = (V, E)$ nek neusmerjen graf. Če $m > \frac{n^2}{2} \cdot \frac{k-2}{k-1}$, potem G vsebuje podgrafe velikosti $k + r$ s povprečno stopnjo najmanj $k + r - 1 - \frac{r}{k+r}$ za vse $r \in \{0, 1, ..., k-2\}$ in $n \geq k + r$.*

Upoštevajmo, da v primeru, ko $r = 0$, ustreza obstoju klik velikosti k kot izražava Turanovega izreka. V veliko primerih so možne le asimptotične cenilke. Na primer, lahko bi pokazali, za graf $G = (V, E)$ na n točkah in m povezavah, če $m = \omega(n^{2-\sqrt{\frac{2}{d \cdot k}}})$, potem ima graf G podgraf s k točkami in povprečno stopnjo d [62] [63].

4.3.2 Najgostejši podgrafi

Pregledali bomo rešitev za izračun najgostejšega podgrafa glede na povprečne stopnje.

Naj bo $\gamma^*(G)$ maksimalna povprečna stopnja nekega nepraznega induciranega podgrafa grafa G, t.j.

$$\gamma * (G) =_{def} \max\{\bar{d}(G[U])|U \subseteq V \text{ in } U \neq \emptyset\}.$$

Kot v primeru $N - jeder$, je maksimalni podgraf z realizacijo $\gamma^*(G)$ enolično določen. Ob upoštevanju tega, imamo naslednji problem:

Problem: Najgostejši podgraf
Vhodni podatki: Graf G
Izhodni podatki: Množica točk grafa G, ki realizira $\gamma^*(G)$

Ta problem je lahko rešen v polinomskem času z uporabo tehnik pretokov (angl. flow techniques) [64] [65] [66].

Izrek 4.31 *Obstaja algoritem za reševanje problema najgostejši podgraf na grafih z n točkami in m povezavami in je rešljiv v času $\mathcal{O}(mn(\log n)(\log \frac{n^2}{m}))$.*

Dokaz. Formuliramo *najgostejši podgraf* kot problem največjega pretoka v odvisnosti od nekega parametra $\gamma \in \mathbb{Q}^+$. Naj bo $G = (V, E)$ nek neusmerjen graf z n točkami in m povezavami. Upoštevajmo, da pretok omrežja vsebuje graf $G' = (V', E')$ in funkcija kapacitete $u_\gamma : E' \to \mathbb{Q}^+$ dana na tak način. V V dodamo izvor s in ponor t; zamenjamo vsako povezavo grafa G (neusmerjen) z dvema usmerjenima povezavama, vsaka s kapaciteto 1; povežemo izvor z vsemi točkami V s povezavami kapacitete m; in povežemo vsako točko $v \in V$ s ponorom s povezavami kapacitete $m + \gamma - d_G(v)$. Bolj natančno, omrežje je definirano na naslednji način:

$$V' =_{def} V \cup \{s, t\}$$

$$E' =_{def} \{(v, w)|\{v, w \in E\} \cup \{(s, v)|v \in V\} \cup \{(v, t)|v \in V\}$$

in za $v, w \in V'$ je funkcija kapacitete u_γ definirana na naslednji način:

$$u_\gamma(v, w) =_{def} \begin{cases} 1, & \text{if } \{v, w\} \in E \\ m, & \text{if } v = s \\ m + \gamma - d_G(v), & \text{if } w = t \\ 0, & \text{if } (v, w) \notin E'. \end{cases}$$

Upoštevajmo kapacitete prerezov v omrežju. Naj bosta S in T neka dela V', dve disjunktni množici točk z $s \in S$ in $t \in T$, $S_+ = S - \{s\}$ in $T_+ = T - t$. Vemo $S_+ \cup T_+ = V$. Če $S+ = \emptyset$, potem je kapaciteta prereza $c(S, \bar{S}) = m|V|$. Drugače dobimo:

$$c(S, T) = \sum_{v \in S, w \in T} u_\gamma(v, w)$$

$$= \sum w \in T_+ u_\gamma(s, w) + \sum_{v \in S_+} u_\gamma v, t + \sum_{v \in S_+, w \in T_+} u_\gamma(v, w)$$

$$= m|T_+| + (m|S_+| + \gamma|S_+| - \sum v \in S_+ d_G(v)) + \sum_{v\in S_+, w\in T_+, \{v,w\}\in E} 1$$

$$= m|V| + |S_+|(\gamma - \frac{1}{|S_+|}(\sum_{v\in S_+} d_G(v) - \sum_{v\in S_+, w\in T_+, \{v,w\}} 1))$$

$$= m|V| + |S_+|(\gamma - \bar{d}G[S_+]). \tag{4.1}$$

Iz enačbe sledi, da je γ naša ocenjena največja povprečna stopnja grafa G. Potrebujemo vedeti, kako lahko ugotovimo, ali je γ prevelika ali premajhna. Dokažemo naslednjo trditev.

Trditev 4.32 *Naj bosta S in T množici, ki realizirata minimalno kapaciteto prereza, z upoštevanjem γ. Potem velja naslednje:*

1. *Če $S_+ \neq \emptyset$, potem $\gamma \leq \gamma^*(G)$.*

2. *Če $S_+ = \emptyset$, potem $\gamma \geq \gamma^*(G)$.*

Trditev je dokazana z naslednjimi arguemnti.

1. Predpostavimo $S_+ \neq \emptyset$. Če $c(\{s\}, V' - \{s\}) = m|V| \geq c(S,T))$, tedaj $|S_+|(\gamma - \bar{d}(G[S_+])) \leq 0$. Iz tega sledi, $\gamma \leq \bar{d}(G[S_+]) \leq \gamma^*(G)$.

2. Predpostavimo $S_+ = \emptyset$. Nadalje predpostavimo nasprotno, da $\gamma < \gamma^*(G)$. Naj bo $U \subseteq V$ ne-prazna množica točk, ki zadošča $\bar{d}(G[U]) = \gamma^*(G)$. Z enačbo 4.1, dobimo

$$c(U \cup \{s\}, \bar{U} \cup \{t\}) = m|V| + |U|(\gamma - \gamma^*(G)) < m|V| = c(S,T),$$

protislovje minimalnosti kapacitete prereza $c(S,T)$. Zatorej, $\gamma \geq \gamma^*(G)$.

Trditev predlaga algoritem za iskanje prave ocene za γ z binarnim iskanjem. Vemo, da ima lahko $\gamma^*(G)$ le končno število vrednosti, t.j.

$$\gamma^*(G) \in \{\frac{2i}{j} | i \in \{0, ..., m\} \text{ in } j \in \{1, ..., n\}\}.$$

Lahko je videti, da je najmanjša možna razdalja med dvema točkama v množici enaka $\frac{1}{n(n-1)}$. Postopek binarnega iskanja za iskanje maksimalne povprečne stopnje podgrafa je zapisan v Algoritmu 3.

Za časovni okvir lahko povemo, da iteracijo izvršimo $\lceil \log((m+1)n(n-1)) \rceil = \mathcal{O}(\log n)$-krat. Znotraj vsake iteracije zaženemo algoritem, ki najde prerez najmanjše kapacitete. Če uporabimo, npr., algoritem ponovnega označevanja (angl. push-relabel algorithm) [67] za izračun največjega pretoka, le-to lahko naredimo v času $\mathcal{O}(nm\log \frac{n^2}{m})$ v omrežju z n točkami in m povezavami. Zunanje omrežje ima $n+2$ točk in $2m+2n$ povezav. To ne spremeni zahtevnosti algoritma največjega pretoka asimptotsko. Iz tega sledi, da je skupna časovna zahtevnost enaka $\mathcal{O}(nm(\log n)(\log \frac{n^2}{m}))$.

Algorithm 3 Najgostejši podgraf z minimalnim prerezom in binarnim iskanjem

Vhod: Graf $G = (V, E)$
Izhod: Množica k točk grafa G
  Določimo $l := 0, r := m$, in $U := \{\}$;
  while $r - l \geq \frac{1}{n(n-1)}$ **do**
    $\gamma := \frac{l+r}{2}$
    Konstruitaj pretočno omrežje (V', E', u_γ)
    Poišči minimalni prerez S in T pretočnega omrežja
    if $S = \{s\}$ **then**
      $r := \gamma$
    else
      $l := \gamma$ in $U := S - \{s\}$
    end if
  end while
  Vrni U

□

Algoritem maksimalnega pretoka [67] je bil uporabljen za izboljšanje časovne zahtevnosti na $\mathcal{O}(nm\log \frac{n^2}{m})$ [67]. Problem *najgostejši podgraf* je bil rešen z linearnim programiranjem [69]. To nam nedvomno da slabšo zgornjo mejo časovne zahtevnosti, toda vsebuje pa nekatere razširitve v primeru usmerjenih grafov.

Usmerjeni grafi. Ne obstaja očitna pot, kako defirati idejo o gostoti v usmerjenih grafih. Dokler so povprečne vhodne stopnje in povprečne izhodne stopnje v usmerjenih grafih enake, sta obe meri neobčutljivi na orientiranost. En pristop temelji na upoštevanju dveh množic točk S in T, ki nista nujno disjunktni, je pa namenjen upoštevanju orientiranosti. Za vsak usmerjen graf $G = (V, E)$ in ne-prazni množici $S, T \subseteq V$, naj E(S, T) označuje množico povezav, ki gredo iz S v T, t.j., $E(S, T) = \{(u, v)|u \in S \text{ in } v \in T\}$. Definiramo povprečno stopnjo para (S, T) v grafu kot:

$$\bar{d}_G(S, T) =_{def} \frac{|E(S, T)|}{\sqrt{|S| \cdot |T|}}.$$

Ta oznaka je bila vpeljana za mero povezanosti med središči (angl. hubs) in avtoritetami (angl. authorities) v spletnih grafih. Množica S je interpretirana kot množica središč in množica T kot množica avtoritet [74], ali kot občudovalci in centri. Če $S = T$ potem je $\bar{d}_G(S, T)$ natančna povprečna stopnja $G[S]$ (t.j., vsota povprečne vhodne in povprečne izhodnje stopnje $G[S]$). Maksimalna povprečna stopnja usmerjenega grafa $G = (V, E)$ je definirana kot:

$$\gamma^*(G) =_{def} \max\{\bar{d}_G(S, T)|S, T \subseteq V in S \neq \emptyset, T \neq \emptyset\}.$$

Najgostejši podgraf je na usmerjenem grafu lahko rešljiv v polinomskem času z linearnim programiranjem [69]. Da to izvedemo, obravnavajmo naslednjo LP poenostavitev LP_γ, kjer γ zavzame vse možne vrednosti $|S|/|T|$:

$$\max \sum_{(u,v)\in E} x_{(u,v)}$$

$$\text{tako, da } x_{(u,v)} \leq s_u \text{ za vse } (u,v) \in E$$

$$x_{(u,v)} \leq t_v \text{ za vse } (u,v) \in E$$

$$\sum_{u \in V} s_u \leq \sqrt{\gamma}$$

$$\sum_{v \in V} t_v \leq \frac{1}{\sqrt{\gamma}}$$

$$x_{(u,v)},\ s_u, t_v \geq 0 \text{ za vse } u, v \in V \text{ in } (u,v) \in E$$

Lahko je pokazati, da je največja povprečna stopnja za graf G maksimum optimalnih rešitev za LP_γ po vseh γ. Vsak linearni program je rešljiv v polinomskem času. Če je $\mathcal{O}(n^2)$ mnogo razmerij za $|S|/|T|$ in posledično za γ, lahko izračunamo maksimalno povprečno stopnjo za G (in enako za podgrafe) v polinomskem času z binarnim iskanjem.

4.3.3 Najgostejši podgrafi dane velikosti

Najgostejši podgraf grafa je zelo krhek (ang. highly fragile), kot graf z neko povprečno stopnjo za katerega ne potrebujemo imeti podgrafa z enako povprečno stopnjo. Torej iz rešitve najgostejših podgrafov ne moremo enostavno sklepati o obstoju podgrafov z določenimi povprečnimi stopnjami in določenimi velikostmi. Za neusmerjen graf $G = G(V,E)$ in parameter $k \in \mathbb{N}$ naj $\gamma^*(G,k)$ označuje maksimalno vrednost povprečne stopnje za vse inducirane podgrafe G, ki imajo k točk, tj.

$$\gamma^*(G,k) =_{\text{def}} \max(\bar{d}(G[U]) | U \subseteq V \text{ in } |U| = k).$$

Sledeči optimizacijski problem je bil uveden v [85]:

Problem: Gost k-podgraf.
Vhodni podatki: Graf G, parameter $k \in \mathbb{N}$.
Izhodni podatki: Množica točk grafa G, ki realizira $\gamma^*(G,k)$.

V nasprotju z najgostejšimi grafi je ta problem računsko težak. Jasno je, da je gost k-podgraf $\mathcal{NP}$-težak (upoštevamo, da na primer $(G, k, k-1)$, pri ustrezni določitvi problema, pomeni iskanje klika velikosti k v G). Največ, na kar lahko upamo, je algoritem s polinomsko zahtevnostjo z zmernim aproksimacijskim razmerjem. Naravni pristop k aproksimaciji $\gamma^*(G,k)$ temelji na požrešni metodi. Primer požrešne metode [85]:

Algorithm 4 Požrešna metoda

Vhod: Graf $G = (V,E)$ in parameter $k \in \mathbb{N}$ (z $|V| \geq k$)
Izhod: Množica k točk grafa G
Točke razvrsti po padajočem vrstnem redu njihovih stopenj;
Naj bo H množica $k/2$ točk z najvišjimi stopnjami;
Izračunaj $N_H(v) = |N(v) \cap H|$, za vse točke $v \in V - H$;
Razvrsti točke $V - H$ po padajočem vrstnem redu N_H vrednosti;
Naj bo R množica $k/2$ točk $V - H$ z najvišjimi N_H vrednostmi;
Vrni $H \cup R$

Izrek 4.33 *Naj bo G graf z n točkami in naj bo $k \in \mathbb{N}$, $k \leq n$. Naj $A(G,k)$ označuje povprečno stopnjo podgrafov grafa G, ki nastanejo zaradi množice točk, ki so izhodi podatki algoritma 4. Potem je*

$$\gamma^*(G,k) \leq \frac{2n}{k} \cdot A(G,k).$$

Dokaz. Za podmnožico $U, U' \subseteq V$ naj $E(U,U')$ označuje množico povezav ene točke iz U in ene točke iz U'. Naj m_U označuje glavno množico povezav $E(G[U])$. Naj d_H označuje povprečno stopnjo $k/2$ točk v G z najvišjo stopnjo glede na G. Vemo, da je $\gamma^*(G,k) \leq d_H$. Dobimo

$$|E(H, V-H)| = d_H \cdot |H| - 2m_H \geq \frac{d_H \cdot k}{2} - 2m_H \geq 0.$$

S požrešno metodo je bil vsaj

$$\frac{|R|}{|V-H|} = \frac{k}{2n-k} > \frac{k}{2n}$$

teh povezav izbran v $G(H \cup R)$. Tako je skupno število povezav v $G(H \cup R)$ vsaj

$$\left(\frac{d_H \cdot k}{2} - 2m_H\right) \cdot \frac{k}{2n} + m_H \geq \frac{d_H \cdot k^2}{4n}.$$

To dokazuje neenakost za povprečno stopnjo.

□

Požrešna metoda je boljša čim večji je k v primerjavi z n. To je primerna izbira, če želimo najti veliko gostih območij v grafu. Za zelo majhen parameter, $k = \mathcal{O}(1)$, je pa to skoraj tako slabo, kot kateri koli trivialni postopek. Aproksimacijsko razmerje $\mathcal{O}(\frac{n}{k})$ je bilo dobljeno iz številnih drugih aproksimacijskih metod, npr. s požrešno metodo, ki temelji na rekurzivnem brisanju točk z minimalno stopnjo ali s semidefiniranim programiranjem. Kakorkoli, da premagamo povezavo med k in n potrebujemo dodaten algoritem, ki deluje za majhne vrednosti k. Glede na naslednjih izrek se zdi mogoče do $\mathcal{O}(n^{\frac{2}{3}})$ [85].

Izrek 4.34 *Gost k-podgraf je lahko aproksimiran v polinomskem času z razmerjem $\mathcal{O}(n^{\frac{1}{3}-\varepsilon})$ za nek $\varepsilon > 0$.*

Znana ni nobena boljša meja, za ta splošni problem. V posebnem razredu grafov je pa mogoče aproksimacijo narediti z boljšim razmerjem. Na primer, za družino gostih grafov, tj. grafi z $\Omega(n^2)$ povezavami, obstaja algoritem, rešen v polinomskem času, z aproksimacijskem razmerjem poljubno blizu 1. Slaba stran tega je, da je večina socialnih omrežij redka, ne gosta. Kar se tiče nižjih meja v aproksimacijskem razmerju, je bilo pred kratkim dokazano, da aproksimacijskega razmerja $1+\varepsilon$ za vsak $\varepsilon > 0$ ni mogoče doseči, razen če so lahko vsi $\mathcal{NP}$ problemi simulirani s slučajnim algoritmom z obojestranskimi napakami in sub-eksponentnim časom trajanja (bolj specifično, v času $\mathcal{O}(n^{\varepsilon})$ za vsak $\varepsilon > 0$). Še več, domnevajo, da ne obstaja algoritem rešen v polinomskem času z aproksimacijskem razmerjem $\mathcal{O}(n^{\varepsilon})$ za vsak $\varepsilon > 0$ [85].

4.3.4 Parametrizirana gostota

Kot smo že povedali, je odločitvena verzija gostega k-podgrafa $\mathcal{NP}$-polna. V nasprotju s to spremenljivko problema odločanja (upoštevajoč da je parameter gostote del vhodnih podatkov), nas sedaj zanima študij verzije fiksnega parametra. Funkcija $\gamma : \mathbb{N} \to \mathbb{Q}_+$ je *začetna gostota* natanko tedaj, ko je γ izračunana v polinomskem času in je $\gamma(k) \leq k-1$ za vsak $k \in \mathbb{N}$. Za vse začetne gostote γ, je γ-gost podgraf grafa $G = (V, E)$ vsaka podmnožica $U \subseteq V$, za katero je $\bar{d}(G[U]) \geq \gamma(|U|)$. Premislimo naslednji problem:

Problem: γ-gost podgraf.
Vhodni podatki: Graf G, parameter $k \in \mathbb{N}$.
Izhodni podatki: Ali obstaja γ-gost podgraf velikosti k v grafu G?

Po eni strani, če izberemo $\gamma(k) = k-1$ za vsak $k \in \mathbb{N}$, dobimo γ-gost podgraf = klika, ki je $\mathcal{NP}$-poln problem. Po drugi strani pa, če izberemo $\gamma(k) = 0$, vsaka izbira k točk povzroči γ-gost podgraf in ta je rešen v polinomskem času. Vprašanje je, katera izbira γ še vedno dopusti algoritem s polinomsko zahtevnostjo in za katero γ ta problem postane $\mathcal{NP}$-poln? Ta problem je študiralo več avtorjev [87, 88, 89]. Naslednji izrek da ostro mejo, kar kaže, da se zahteven preskok pojavi zelo zgodaj [89].

Izrek 4.35 *Naj bo γ začetna gostota.*

1. *Če $\gamma = 2 + \mathcal{O}(\frac{1}{k})$, potem je γ-gost podgraf rešen v polinomskem času.*
2. *Če $\gamma = 2 + \Omega(\frac{1}{k^{q-\varepsilon}})$, za $\varepsilon > 0$, potem je γ-gost podgraf $\mathcal{NP}$-poln.*

Neposredna uporaba izreka da naslednji rezultat za primer konstantne funkcije gostote.

Posledica 4.36 *Iskanje k-točk podgrafa, s povprečno stopnjo vsaj 2, je lahko končano v polinomskem času. Kakorkoli, tu ni algoritma za iskanje k-točk podgrafa s povprečno stopnjo vsaj 2+ε, za vsak $\varepsilon > 0$, razen če $\mathcal{P} = \mathcal{NP}$.*

Ta rezultat bi moral biti v nasprotju z ustreznim rezultatom za N-jeder, kjer je iskanje N-jedra velikosti k lahko končano v linearnem času, celo za vse $N > 0$. To kaže na izrazite računske razlike med statistično in strukturno gostoto.
Rezultat, enak izreku 4.35, je bil dokazan za primer posebnih razredov omrežja z značilnostmi resničnega sveta, zlasti za power-low grafe in splošne redke grafe.

4.4 Poglavje opomb

V tem poglavju bomo preučili računske vidike pojmov lokalnih gostot, tj. pojmov gostot, definiranih samo v induciranih podgrafih, ki posledično odpravljajo omrežne strukture izven podgrup. Upoštevali bomo strukture (N-plex, N-jedra) in statistične omilitve (η-goste podgrafe) koncepta klika, ki je popolno povezana podgrupa. Čeprav je veliko algoritmov za pojme lokalnih gostot računsko težkih (ne poznamo algoritmov s polinomsko zahtevnostjo, ki bi hitro rešili problem), obstaja nekaj primerov algoritmov, ki hitro vrnejo zaželeno informacijo o gostoti, ki temelji na strukturi povezanosti omrežja. Na

primer, število manjših klikov v grafu, število jeder ali maksimalna povprečna stopnja, ki je dosegljiva iz podgrupe v usmerjenem in neusmerjenem omrežju.

Iz predstavljenih rezultatov opazimo, da je (na videz) težak kompromis med matematičnimi pravili in smiselnostmi teh pojmov ter njihovo algoritmično ubogljivostjo. To je razvidno iz spodnje tabele, ki povzema lastnosti naših glavnih pojmov.

Podskupina	Zaprta za izključitev	Ugnezdena	Ubogljiva
Klika	+	+	-
N-plex ($N \in \mathbb{N}$)	+	+	-
N-jedro ($N \in \mathbb{N}$)	-	-	+
η-gost podgraf	-	+	-

Tukaj vidimo, da ugnezdenost, kot pomembna struktura znotraj grupe, izključuje hitre algoritme za računanje podgrup določene velikosti. Ta izključitev je podedovana z nekaj omilitvami. Vendar pa nimamo strogih dokazov za ta opazovanja v primeru splošne lokalno določene podgrupe. Po drugi strani, podobna relacija je dokazano pravilna za zaprtje pod izključitvijo in učinkovito odkrivanje podgrup dane velikosti: ne moremo doseči obeh z ustreznim pojmom gostosti.

To poglavje bomo zaključili s kratko razpravo o izbiri nelokalnih konceptov povezanosti podgrup, ki so pritegnile zanimanje za analizo socialnih omrežij. Odkar nelokalnost poudarja pomembnost povezanih podgrup, ločenih od preostalega omrežja, imajo ti pojmi pomembno vlogo pri modelih za strukturo jedra ali obrobja.

LS množica (Luccio-Sami množica). LS množice je mogoče razumeti kot regije omrežja, kjer so notranje vezi pomembnejše od zunanjih. Natančneje, za graf $G = (V, E)$ je podmnožica točk $U \subseteq V$ LS množica natanko tedaj, ko za vse neprazne podmnožice $U' \subset U$ velja

$$|E(U', V - U)| > |E(U, V - U)|.$$

Trivialno, V je LS množica. Tudi množice singeltonov $\{v\}$ so LS množice v G za vsak $v \in V$. Struktura LS množice ima nekaj lepih lastnosti. Na primer, da ne ne-trivialno pokrivajo, tj. če sta U_1 in U_2 LS množici, $U_1 \cap U_2 \neq \emptyset$, potem je ali $U_1 \subseteq U_2$ ali $U_2 \subseteq U_1$. Poleg tega so LS množice precej goste: minimalna stopnja ne-trivialnosti LS množice je vsaj polovica od števila izhodnih povezav [86]. Upoštevamo, da je struktura moči LS množic zelo odvisna od splošne zahteve, da imajo vse ustrezne podmnožice več vezi z zunanjim omrežjem kot pa množica U [86].

Lambda množica. Pojem, tesno povezan z LS množico, je lambda množica. Naj bo $G = (V, E)$ neusmerjen graf. Za točki $u, v \in V$ naj $\lambda(u, v)$ označuje število robnih disjunktnih poti med u in v v G, tj. $\lambda(u, v)$ meri robne povezanosti u in v v G. Podmnožica $U \subseteq V$ je lambda množica natanko tedaj, ko

$$\min_{u,v \in U} \lambda(u, v) > \max_{u \in U, v \in V - U} \lambda(u, v).$$

V lambda množici imajo njihovi člani več povezanih robnih disjunktnih poti kot nečlani. Vsaka LS množica je lambda množica [86]. Lambda množica ne meri neposredno gostote

podmnožic. Ima pa določen pomen, saj omogoča algoritem za njihov izračun s polinomskmo zahtevnostjo. Algoritem je v bistvu sestavljen iz dveh delov, iz izračuna matrike robne povezanosti točk množice V (kar je mogoče storiti s tokovnim algoritmom v $\mathcal{O}(n^4)$) ter iz združevanja točk na level-wise način, tj. točki u in v pripadata isti lambda množici (na ravni N) $\Leftrightarrow \lambda(u, v) \geq N$. Algoritem je tudi mogoče enostavno razširiti na izračun LS množic.

Normalna množica. V [90] je normalnost za omrežne podgrupe definirana v statističnem načinu preko slučajnih sprehodov na grafu. Eden od najpomembnejših razlogov za obravnavo slučajnih sprehodov je, da so ponavadi posledice algoritmov enostavne, hitre in splošne. Slučajni sprehod je stohastični proces s katerim gremo čez graf, tako da naključno izberemo naslednjo točko, ki jo bomo obiskali, med sosedi sedanje točke. Slučajni sprehod lahko uporabimo za "ulov"pojma kakovosti povezane podgrupe. Intuicija je, da bolj kot je grupa povezana, večja je verjetnost, da slučajni sprehod, z začetkom na nekem članu te grupe, ne zapusti grupe.

Naj bo $G = (V, E)$ neusmerjen graf. Za $d \in \mathbb{N}$ in $\alpha \in \mathbb{R}_+$ je podmnožica $U \subseteq V$ (d, α)-normalna natanko tedaj, ko je za vsako točko $u, v \in U$, za katero je $d_G(u, v) \leq d$, verjetnost, da bo slučajni sprehod, ki je začel v u, prej obiskal v kot pa $w \in V - U$, enaka vsaj α. Čeprav je ta pojem precej intuitiven, ne vemo kako izračunati normalno množico ali pripadajoče omrežje za normalno množico. Namesto tega so bili nekateri hevristični algoritmi, ki rešijo problem v linearnem času (vsaj na grafih z omejeno stopnjo), razviti za proizvodnjo razpada v duhu normalnosti [90].

Poglavje 5

Povezanost

Miha Eržen, Zala Herga, Nika Šušterič, Nina Zupančič

Uvod

V seminarski nalogi smo se osredotočili predvsem na moči povezav med točkami, z ozirom na število po povezavah oz. točkah disjunktnih poti (*vertex- or edge-disjoint paths*). Kot bomo videli v nadaljevanju, je to ekvivalentno vprašanju, *koliko vozlišč (nodes) oziroma povezav (edges) moramo odstraniti iz grafa, da uničimo vse poti med dvema točkama (vertices).*

Osnovna definicija povezanosti pravi, da je graf $G = (V, E)$ *povezan*, če za vsak par točk $u, v \in V$ obstaja pot v G med njima. Največjemu povezanemu delu grafa pravimo *komponenta*. Točko v grafa G imenujemo *prerezna točka*, če ima graf $G - v$ več komponent kot G. Podobno imenujemo povezavo e grafa G *prerezna povezava* oziroma *most*, če z odstranitvijo te povezave dobimo graf z več komponentami, kot jih je imel prvotni graf G. Graf G je *(po točkah) k-povezan* ($k \in \mathbb{N}, |V(G)| > k$), če je $G - S$ povezan za vsako *množico točk* $S \subseteq V$, kjer je $|S| < k$; *k-povezan* graf ima vsaj $k + 1$ točk. Če je graf k-povezan, je tudi j-povezan za vsako število $j \leq k$. Največje število k, za katero je graf G k-povezan, imenujemo *povezanost* grafa G in ga označimo s $\kappa(G)$. Graf G je *po povezavah l-povezan*, če je za vsako (*prerezno množico povezav*) $F \subseteq E$ $G - F$ povezan, pri čemer je $|F| < l$. Z $\lambda(G)$ označimo največji l, za katerega je G še po povezavah l-povezan. Povezanost po povezavah lahko definiramo tudi takole: $\lambda(G)$ grafa G je najmanjše število povezav, brez katerih postane graf G nepovezan. *Blok* grafa G je maksimalen povezan podgraf brez prereznih točk. Blok grafa je bodisi izolirana točka v G, bodisi most v G ali pa maksimalen *2-povezan* podgraf v G. Zato imata dva različna bloka največ eno skupno točko, ki je vedno prerezna točka. Torej vsaka povezava leži v enem bloku in je graf unija blokov.

Najprej bomo predstavili nekaj splošnih definicij in izrekov o osnovnih lastnostih povezanosti, ki bodo kasneje zadostovali za razumevanje obravnavanih algoritmov. Pokazali bomo algoritme, ki

- preverijo k-točkovno (k-povezavno) povezanost,
- izračunajo točkovno (povezavno) povezanost in

- izračunajo maksimum k-povezanih komponent danega grafa.

Kot zgoraj, označimo povezanost grafa G s $\kappa(G)$ in povezanost po povezavah z $\lambda(G)$. Definirajmo še lokalno povezanost $\kappa_G(s,t)$ za dve določeni točki s in t, kot minimalno število točk, ki morajo biti odstranjene, da prekinemo vse poti med s in t. V primeru, da obstaja povezava iz s v t, definiramo $\kappa_G(s,t) = n-1$, saj v nasprotnem primeru κ_G ne more preseči $n-2$ (če sta s in t povezana s povezavo, ju ni mogoče ločiti le z odstranitvijo točke). Podobno sledi, da je $\lambda_G(s,t)$ najmanjše število povezav, ki jih moramo odstraniti, da med s in t ni več nobene poti. Pri neusmerjenih grafih očitno velja $\kappa_G(s,t) = \kappa_G(t,s)$ in $\lambda_G(s,t) = \lambda_G(t,s)$, medtem ko pri usmerjenih grafih te funkcije v splošnem niso simetrične.

Izrazi v literaturi se med seboj malo razlikujejo, zato velja še enkrat poudariti imena naših osnovnih izrazov, ki jih bomo uporabljaji v nadaljevanju (razlaga le-teh zgoraj): *prerezna točka, prerezna povezava oz. most, komponenta* in *blok*.

Bločni graf $B(G)$ grafa G sestavljajo po ena točka iz vsakega bloka grafa G. Dve točki bločnega grafa sta sosednji, če in samo, če si pripadajoča bloka delita skupno točko (tj. prerezno točko). Tako imenovani *prerezni graf (cutpoint-graph)* $C(G)$ grafa G je sestavljen iz točke za vsako prerezno točko iz G, kjer so si točke sosednje samo, če je pripadajoča prerezna točka del istega bloka grafa G. Za bločni in prerezni graf veljajo naslednje enakosti, in sicer: $B(B(G)) = C(G)$ ter $B(C(G)) = C(B(G))$. *Bločno-prerezni graf* grafa G je dvodelen graf, sestavljen iz množice prereznih točk grafa G in množice točk, ki predstavljajo bloke grafa G. Prerezna točka je sosednja bločni-točki, natanko takrat, ko prerezna točka pripada prilegajočemu se bloku. Bločno-prerezni graf povezanega grafa je drevo. Maksimalen k-povezan (oz. po povezavah k-povezan) podgraf imenujemo k-komponentni podgraf (oz. po povezavah k-komponentni podgraf). Po povezavah k-komponentni graf, ki ne vsebuje nobene od $(k+1)$-komponent se imenuje *segment* oz. *gruča (cluster)*.

5.1 Osnovni izreki

Izrek 5.1 *Za vse netrivialne grafe G velja:*

$$\kappa(G) \leq \lambda(G) \leq \delta(G) \ .$$

Dokaz. Naj ima točka v stopnjo $\delta(G)$ ter naj bo F množica vseh incidenčnih povezav točke v. Graf $G-F$ je nepovezan. Torej velja $\lambda(G) \leq \delta(G)$.

Pokažimo drugo neenakost. Naj bo S prerezna množica povezav moči $\lambda(G)$. Če torej odstranimo te povezave, postane graf $G-S$ nepovezan. Te povezave pa lahko odstranimo tudi tako, da odstranimo po eno krajišče vsake od teh povezav - pri odstranitvi točk pazimo, da ne odstranimo vseh točk bodisi v levi bodisi v desni množici. Takšnih točk je največ $\lambda(G)$, zato velja $\kappa(G) \leq \lambda(G)$.

□

Naslednji izrek je v svojem delu o splošni teoriji krivulj objavil Karl Menger [135].

Izrek 5.2 (Menger, 1927) *Naj bo $G = (V,E)$ neusmerjen graf in $P, Q \subseteq V$ podmnožici točk grafa. Potem je največje število po točkah disjunktnih poti med P in Q (oz. (P,Q)-poti) enako najmanjšemu številu točk, ki ločijo množici P in Q tj. najmanjši moči (P,Q)-prereza v G.*

Od tod sledi eden pomembnejših izrekov v teoriji grafov:

Posledica 5.3 (Mengerjev izrek) *Naj bosta $s, t \in V$ $(s \neq t)$ nesosednji točki neusmerjenega grafa $G = (V, E)$. Potem je največje število po točkah disjunktnih (s, t)-poti v G enako najmanjši moči množice, ki loči točki s in t.*

Analogen pomen ima naslednji izrek.

Izrek 5.4 *Največje število po poteh disjunktnih (s, t)-poti v G je enako najmanjšemu številu povezav, ki ločijo s in t v G.*

Zgornjemu izreku pogosto pravimo *Mengerjev izrek za povezave*, čeprav je bil prvič objavljen šele tri desetletja po Mengerjevemu izreku (za točke). Z njim je močno povezan tudi *Max-Flow-Min-Cut izrek* Forda in Fulkersona [139], ki pravi, da je v vsakem omrežju vrednost maksimalnega (s, t)-pretoka enaka kapaciteti minimalnega (s, t)-prereza. Mengerjev izrek za povezave si lahko interpretiramo tudi kot poseben primer Max-Flow-Min-Cut izreka, kjer imajo kapacitete vseh povezav enotno vrednost.

Tako imenovano *Globalno verzijo Mengerjevega izreka* je leta 1932 objavil Hassler Whitney [140], zato mu ponavadi pravimo kar *Whitneyev izrek*.

Izrek 5.5 (Whitney, 1932) *Naj bo $G = (V, E)$ (netrivialen) graf in k, l pozitivni števili.*

- *Graf G je k-povezan natanko takrat, ko vsebuje k po točkah disjunktnih poti med poljubnima dvema točkama.*
- *Graf G je po povezavah l-povezan natanko takrat, ko vsebuje l po povezavah disjunktnih poti med poljubnima dvema točkama.*

Beineke in Harary sta prišla do podobne ugotovitve za sestavljeno (torej po povezavah in točkah) povezanost (glej [141]). Predpostavila sta takšne *pare povezanosti* (k, l), da obstaja množica k točk in l povezav, za katere velja, da je graf nepovezan, če jih odstranimo.

Izrek 5.6 (Beineke & Harary, 1967) *Naj bo (k, l) par povezanosti za točki s in t v grafu G. Potem obstaja $k + l$ po povezavah disjunktnih poti med s in t, med katerimi se jih k zagotovo ne seka.*

Sledeč izrek govori o mejah povezanosti po točkah in po povezavah [142].

Izrek 5.7 *Maksimum povezanosti (po točkah/povezavah) na grafu z n točkami in m povezavami je*

$$\begin{cases} \lfloor \frac{2m}{n} \rfloor , & \text{če } m \geq n-1 \\ 0 , & \text{sicer.} \end{cases}$$

Minimum povezanosti (po točkah/povezavah) na grafu z n točkami in m povezavami je

$$\begin{cases} m - \binom{n-1}{2} , & \text{če } \binom{n-1}{2} < m \leq \binom{n}{2} \\ 0 , & \text{sicer.} \end{cases}$$

Naslednja izjava se nanaša na poseben primer povezanosti po povezavah, z njo pa se je ukvarjal Chartrand [143]:

Izrek 5.8 *Za vse grafe $G = (V, E)$, z minimalno stopnjo $\delta(G) \geq \lfloor |V|/\ 2 \rfloor$, je povezanost po povezavah enaka minimalni stopnji grafa: $\lambda(G) = \delta(G)$.*

Izreki v nadaljevanju se navezujejo na *k-točkovne/povezavne komponente* grafa. Dokaj očitni sta dejstvi, da dve različni komponenti grafa nimata nobene skupne točke in si dva različna bloka delita največ eno skupno točko. O tem sta v svojem delu o n-povezanih grafih pisala Harary in Kodama [136].

Izrek 5.9 *Dve različni k-(točkovni-)komponenti imata največ $k - 1$ skupnih točk.*

Velja omeniti, da *k-povezavne-komponente* ne morejo sovpadati, medtem ko *k-točkovne-komponente* lahko.

Izrek 5.10 (Matula, 1968) *Za katerokoli fiksno naravno število $k \geq 1$ so k-povezavne-komponente grafa po točkah disjunktne [137].*

Dokaz. Predpostavimo (sovpadajočo) dekompozicijo $\tilde{G} = G_1 \cup G_2 \cup \cdots \cup G_t$ povezanega podgrafa $\tilde{G}$ grafa G. Naj bo $C = (A, \bar{A})$ prerez grafa $\tilde{G}$ po povezavah na nepovezana dela A in $\bar{A}$. Predpostavimo še, da ima $\tilde{G}$ vsaj 2 točki (v izogib dokazovanju trivialnega primera). Za vsak podgraf G_i, ki vsebuje določeno povezavo $e \in C$ iz minimalnega prereza, velja, da prerez vsebuje tudi prerez za G_i (v nasprotnem bi bili dve točki povezani v $G_i \setminus C$ in v $\tilde{G} \setminus C$, kar bi bilo v protislovju s predpostavko, da je C minimalni prerez). Sledi torej, da je G_i tak, da $\lambda(\tilde{G}) = |C| \geq \lambda(G_i)$, od tod pa $\lambda(\tilde{G}) \geq \min_{1 \leq i \leq t}\{\lambda(G_i)\}$, s čimer smo dokazali izrek (glej [137]).

□

Čeprav iz Izreka 5.1 vidimo, da iz k-točkovne/povezavne-povezanosti sledi, da je minimalna stopnja vsaj k, obratno ne velja. V primeru visoke minimalne stopnje, obstaja tudi visoko povezan podgraf.

Izrek 5.11 (Mader, 1972) *Vsak graf s povprečno stopnjo vsaj $4k$ ima k-povezan podgraf.*

Na temo povezanosti usmerjenih grafov je bilo narejenih kar nekaj študij - ena izmed njih se nanaša na usmerjena vpeta drevesa *(vpeto drevo v G je vsak podgraf, ki je drevo)*, s korenom v vozlišču r, ki jim v angleščini pravimo *r-branchings*.

Izrek 5.12 (Edmondov izrek) *V usmerjenem multigrafu $G = (V, E)$, ki vsebuje točko r, je maksimalna vrednost paroma po povezavah disjunktnih vpetih dreves s korenom v vozlišču r enaka $\kappa_G(r)$, kjer $\kappa_G(r)$ označuje minumum (med vsemi množicami točk $S \subset V$, ki vsebujejo r) števila povezav, ki gre iz S.*

Povezavo med maksimalnim številom usmerjenih po povezavah disjunktnih poti in vhodno- ter izhodno- stopnjo točk, nam da Lovász-ev izrek [138].

Izrek 5.13 (Lovász, 1973) *Naj bo $v \in V$ točka grafa $G = (V, E)$. Če $\lambda_G(v, w) \leq \lambda_G(w, v)$ za vse točke $w \in V$, potem $d^+(v) \leq d^-(v)$.*

Ta izrek lahko uporabimo kot dokaz Kotzig-ove domneve.

Izrek 5.14 (Kotzigov izrek) *Za usmerjen graf $G = (V, E)$, je $\lambda_G(v, w)$ enaka $\lambda_G(w, v)$ za vse $v, w \in V$ samo v primeru, ko je graf pseudo-simetričen, to pomeni, da v vseh točkah velja, da je vhodna stopnja enaka izhodni: $d^+(v) = d^-(v)$.*

5.2 Uvod v minimalne prereze

V neusmerjenem uteženem grafu je vsota uteži na povezavah s krajišči v disjunktnih množicah točk X in Y definirana z $w(X, Y)$. Za usmerjene grafe je $w(X, Y)$ definirana podobno, le da tu štejejo le uteži povezav, ki vodijo iz X v Y. Prerez uteženega grafa $G = (V, E)$ je množica točk $\emptyset \subset S \subset V$, njegova teža pa je $w(S, V \setminus S)$. V neuteženem grafu je teža prereza enaka številu povezav iz S v $V \setminus S$.

Definicija 5.15 Minimalen prerez je tak prerez S, da za vse ostale prereze T velja

$$w(S, V \setminus S) \leq w(T, V \setminus T) \ .$$

Problem algoritma, ki izračuna vse *minimalne prereze* je, da mora vse prereze tudi shraniti, vendar za to porabi veliko prostora. Predlog poceni algoritma je leta 1976 predstavil Dinitz (glej [144]), in sicer kot podatkovno strukturo imenovano *kaktus*. Ta združuje vse minimalne prereze neusmerjenega (uteženega) grafa. Velikost kaktusa je linearna s številom vhodnih točk, kaktus pa omogoča tudi računanje prereza v času, lineranem z velikostjo prereza.

Karzanov in Timofeev sta predstavila prvi način za konstruiranje kaktusa za neutežene, neusmerjene grafe (glej [145]). Njun algoritem sestavljata 2 dela: v podanem grafu G se v prvem delu poišče niz vseh minimalnih prerezov grafa G, v drugem delu pa se skonstruira kaktus C_G, sestavljen iz tega niza. Algoritem deluje tudi na uteženih grafih, vendar le v primeru pozitivnih uteži.

Če so dovoljene negativne uteži, imamo opravka z $\mathcal{NP}$-polnim problemom. Poleg tega, ni poznana niti posprošitev iskanja minimalnega prereza za usmerjene grafe. Neutežen graf lahko preoblikujemo v uteženega z dodajanjem teže velikosti 1, na vsako izmed povezav, od koder sledi, da bomo tak graf obravnavali kot v primeru neusmerjenega povezanega grafa s pozitivnimi utežmi.

Predpostavimo omrežje N, definirano z usmerjenim grafom $G = (V, E)$, kapacitetno funkcijo u_N, izvorom s, ponorom t in pretokom f. *Residualno omrežje* R_f sestavljajo povezave, ki lahko nosijo še dodaten tok, poleg tistega, ki ga že nosijo v okviru pretoka f. R_f je definiran z grafom $G_{R_f} = (V, \{(u, v) \mid ((u, v) \in E \text{ ali } (v, u) \in E) \text{ in } u_{R_f}((u, v)) > 0\})$, enakim izvorom s in ponorom t ter sledečo kapacitetno funkcijo:

$$u_{R_f}((a, b)) = \begin{cases} c(a, b) - f(a, b) + f(b, a) \ , & \text{če } (a, b) \in E \text{ in } (b, a) \in E \\ c(a, b) - f(a, b) \ , & \text{če } (a, b) \in E \text{ in } (b, a) \notin E \\ f(b, a) \ , & \text{če } (a, b) \notin E \text{ in } (b, a) \in E. \end{cases}$$

Naj bo $R_{f_{max}}$ residualno omrežje omrežja N in f_{max} maksimalen (s, t)-pretok v N. Maksimalen (s, t)-pretok je enak minimalnemu (s, t)-prerezu, od koder sledi, da je vsaka množica točk $S \subseteq V \setminus t$ minimalen (s, t)-prerez, če je $s \in S$ in nobena povezava v omrežju $R_{f_{max}}$ ne gre iz S.

5.3 Minimalni prerezi vseh parov točk

Problem izračunavanja minimalnih prerezov med vsemi pari točk je lahko opravljen z izračunom $n(n-1)/2$ pretočnih problemov. Gomory in Hu [146] sta dokazala, da izračun $n - 1$ maksimalnih pretokov zadostuje za določitev vrednosti maksimalnega pretoka oziroma minimalnega prereza za vse pare točk. Rezultat prikažemo v tako imenovanem

ekvivalentnem pretočnem drevesu . To je uteženo drevo na n točkah, minimalna teža katerekoli povezave na (s,t)-poti pa je enaka maksimalnemu pretoku iz s v t. Poleg tega sta Gomory in Hu dokazala, da vedno obstaja ekvivalentno pretočno drevo, kjer komponente, ki jih dobimo z odstranjevanjem minimalne teže povezav iz (s,t)-poti, predstavljajo minimalen (s,t)-prerez. To drevo imenujemo *Gomory-Hu prerezno drevo*.

Z izračunavanjem minimalnega prereza se je ukvarjalo (in se še vedno) kar nekaj matematikov. Med drugimi je poenostavljen izračun leta 1990 predstavil Gusfield [147], v nadaljevanju pa bo podrobneje opisan algoritem, ki sta ga leta 1994 objavila Stoer in Wagner.

5.4 Lastnosti minimalnega prereza v neusmerjenih grafih

Imamo $2^{|V|}$ množic in vsaka od teh bi bila lahko minimalni prerez, kljub temu pa je število minimalnih prerezov v fiksnem neusmerjenem grafu polinomsko v $|V|$. Da pridemo do tega, si moramo najprej pogledati nekaj že dobro znanih dejstev o minimalnih prerezih. Ta dejstva nam prav tako pomagajo definirati podatkovno strukturo, ki ji pravimo *kaktus*. Kaktus lahko predstavlja vse minimalne prereze in za to potrebuje samo prostor, ki je linearen v $|V|$.

Na kratko, za graf G bo v tem poglavju λ_G vedno označevala težo minimalnega prereza. Če bo obravnavani graf G jasen iz konteksta, bo indeks G iz λ_G izpuščen.

Lema 5.16 *Naj bo S minimalni prerez v $G=(V,E)$. Potem za vse $\emptyset \neq T \subset S$ velja:*

$$w(T, S \setminus T) \geq \frac{\lambda}{2}$$

.

Dokaz. Predpostavimo, da $w(T, S \setminus T) \leq \frac{\lambda}{2}$. Ker je $w(T, V \setminus S) + w(S \setminus T, V \setminus S) = \lambda$ lahko brez škode za splošnost vzamemo: $w(T, V \setminus S) \leq \frac{\lambda}{2}$ (sicer definiramo T kot $S \setminus T$). Potem $w(T, V \setminus T) = w(T, S \setminus T) + w(T, V \setminus S) \leq \lambda$, to je pa protislovje.

□

Lema 5.17 *Naj bosta $A \neq B$ dva taka minimalna prereza, da je tudi $T := A \cup B$ minimalni prerez. Potem*

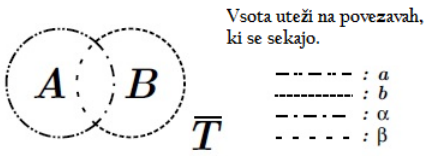
$$w(A, \overline{T}) = w(B, \overline{T}) = w(A \setminus B, B) = w(A, B \setminus A) = \frac{\lambda}{2}$$

.

Dokaz. Kot v sliki 5.1 naj bo $a = w(A, \overline{T})$, $b = w(B, \overline{T})$, $\alpha = w(A, B \setminus A)$ in $\beta = w(B, A \setminus B)$. Potem je $w(A, \overline{A}) = a + \alpha = \lambda$, $w(B, \overline{B}) = b + \beta = \lambda$ in $w(T, \overline{T}) = a + b = \lambda$.

Poleg tega vemo tudi, da je $w(A \setminus B, B \cup \overline{T}) = a + \beta \geq \lambda$ in $w(B \setminus A, A \cup \overline{T}) = b + \alpha \geq \lambda$. Ta sistem enačb in neenačb ima enolično rešitev: $a = \alpha = b = \beta = \frac{\lambda}{2}$.

□



Slika 5.1: Presek dveh minimalnih prerezov A in B

Definicija 5.18 Paru $\langle S_1, S_2\rangle$ pravimo *prekrižni prerez*, če sta S_1, S_2 dva minimalna prereza in niso prazne niti $S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1$ niti $\bar{S_1} \cap \bar{S_2}$.

Lema 5.19 *Naj bo par $\langle S_1, S_2\rangle$ prekrižni prerez in $A = S_1 \cap S_2$, $B = S_1 \setminus S_2$, $C = S_2 \setminus S_1$ in $D = \overline{S_1} \cap \overline{S_2}$. Potem:*

1. *A, B, C, in D so minimalni prerezi*
2. *$w(A, D) = w(B, C) = 0$*
3. *$w(A, B) = w(B, D) = w(D, C) = w(C, A) = \frac{\lambda}{2}$.*

Dokaz. Ker že vemo, da sta S_1 in S_2 minimalna prereza, lahko zaključimo, da

$$\begin{aligned} w(S_1, \overline{S_1}) &= w(A, C) + w(A, D) + w(B, C) + w(B, D) = \lambda \\ w(S_2, \overline{S_2}) &= w(A, B) + w(A, D) + w(B, C) + w(C, D) = \lambda \end{aligned}$$

in ker ne obstaja prerez s težo manjšo od λ, vemo da

$$\begin{aligned} w(A, \overline{A}) &= w(A, B) + w(A, C) + w(A, D) \geq \lambda \\ w(B, \overline{B}) &= w(A, B) + w(B, C) + w(B, D) \geq \lambda \\ w(C, \overline{C}) &= w(A, C) + w(B, C) + w(C, D) \geq \lambda \\ w(D, \overline{D}) &= w(A, D) + w(B, D) + w(C, D) \geq \lambda. \end{aligned}$$

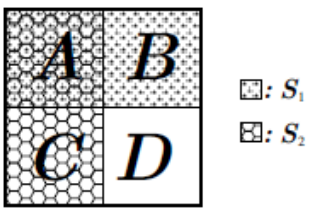
Če množimo prvi dve enačbi z dve in ju seštejemo, dobimo

$$2w(A, B) + 2w(A, C) + 4w(A, D) + 4w(B, C) + 2w(B, D) + 2w(C, D) = 4\lambda$$

in ko seštejemo še obe strani vseh štirih neenačb, imamo

$$2w(A, B) + 2w(A, C) + 2w(A, D) + 2w(B, C) + 2w(B, D) + 2w(C, D) \geq 4\lambda.$$

Sledi, da je $w(A, D) = w(B, C) = 0$. Z drugimi besedami, ne obstajajo diagonalne povezave v sliki 5.2.



Slika 5.2: Prekrižni prerez $\langle S_1, S_2\rangle$ s $S_1 = A \cup B$ in $S_2 = A \cup C$

Za boljšo predstavo: predpostavimo, da je dolžina štirih notranjih črt, ki v sliki 5.2 ločujejo A,B,C in D, sorazmerna z vsoto vseh uteži povezav, ki sekajo te črte. Tako je skupna dolžina l obeh horizontalnih (ali pa obeh vertikalnih črt) sorazmerna s težo λ.

Predpostavimo, da so te štiri črte različnih dolžin oz., da se črti, ki ločujeta množici S_1 od $\overline{S_1}$ (oz. S_2 od $\overline{S_2}$), ne sekata točno v središču kvadrata; potem je celotna dolžina ločitvenih črt množice točk $\Delta = A, B, C$ ali D krajša od l. Torej $w(\delta, \overline{\delta}) \leq \lambda$. To je protislovje.

Kot posledico dobimo, da $w(A, B) = w(B, D) = w(D, C) = w(C, A) = \frac{\lambda}{2}$ in $w(A, \overline{A}) = w(B, \overline{B}) = w(C, \overline{C}) = w(D, \overline{D}) = \lambda$.

□

Prekrižni prerez v $G = (V, E)$ razdeli množico točk V na natančno štiri dele. Sledi bolj splošna definicija, kjer lahko množico točk razdelimo na 3 ali več delov.

Definicija 5.20 *Krožna particija* je particija množice V na $k \geq 3$ disjunktnih množic $V_1, V_2, \ldots, V_k$, tako da:

1. $w(V_i, V_j) = \begin{cases} \frac{\lambda}{2}, & |i - j| = 1 \bmod k \\ 0, & \text{sicer;} \end{cases}$

2. Če je S minimalni prerez, potem:

   (a) S ali $\overline{S}$ je prava podmnožica neke V_i ali

   (b) Krožna particija je dopolnjenje particije, definirane z minimalnim prerezom S. Z drugimi besedami: minimalni prerez je unija nekaterih množic krožne particije.

Če so $V_1, V_2, \ldots, V_k$ disjunktne množice krožne particije, potem je za vse $1 \leq a \leq b \leq k$, $S = (\cup_{i=a}^{b} V_i)$ minimalni prerez. Seveda je tudi komplement S, ki vsebuje V_k, minimalni prerez. Definirajmo te minimalne prereze kot *prereze krožne particije*. Vsak $V_i, 1 \leq i \leq k$ je minimalni prerez (lastnost 1. prejšnje definicije).

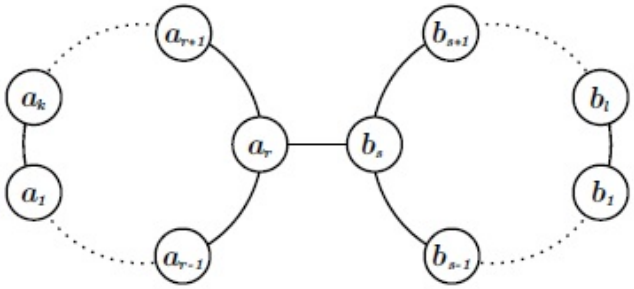
Naj bo minimalni prerez S tak, da ni niti S niti njegov komplement vsebovan v množici krožne particije. Ker je S povezan (Opomba 1), sta S in njegov komplement enaka $\cup_{i=a}^{b} V_i$ za neke $1 \leq a \leq b \leq k$.

Še več: za nobeno množico V_i krožne particije ne obstaja minimalni prerez S, tako da bi bil $\langle V_i, S \rangle$ prekrižni prerez (lastnost 2. zadnje definicije).

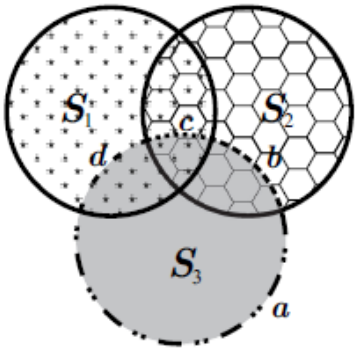
Definicija 5.21 Dve različni krožni particiji $P = \{U_1, \ldots, U_k\}$ in $Q = \{V_1, \ldots, V_k\}$ sta *kompatibilni* če obstajata enolična r in s, $1 \leq r, s \leq k$ taka, da za vse $i \neq r : U_i \subseteq V_s$ in za vse $j \neq s : V_j \subseteq U_r$.

Lema 5.22 *[148] Vse različne krožne particije so paroma kompatibilne.*

Dokaz. Oglejmo si dve krožni particiji P in Q v grafu $G = (V, E)$. Vse množice te particije so minimalni prerezi. Predpostavimo, da je množica $S \in P$ enaka uniji več kot ene in ne vseh množic Q. Natanko dve množici $A, B \in Q$ vsebovani v S sta povezani z najmanj eno povezavo z vozli iz $V \setminus S$. Vzemimo T iz S tako, da zamenjamo $A \subset S$ z elementom Q, ki je povezan z B in ni vsebovan v S. Potem je $\langle S, T \rangle$ prekrižni prerez in imamo protislovje.



Slika 5.3: Graf $G = (\{a_1, \ldots, a_r, b_1, \ldots, b_s\}, E)$ prikazuje dve kompatibilni particiji P, Q definirani kot $P = \{\{a_1\}, \ldots, \{a_{r-1}\}, \{a_r, b_1, \ldots, b_l\}, \{a_{r+1}\}, \ldots, \{a_k\}\}$, $Q = \{\{b_1\}, \ldots, \{b_{s-1}\}, \{b_s, a_1, \ldots, a_k\}, \{b_{s+1}\}, \ldots, \{b_l\}\}$



Slika 5.4: Trije paroma prekrižni prerezi S_1, S_2 in S_3

Torej je vsaka množica P ali njen komplement vsebovana v neki množici Q.

Predpostavimo, da sta dve množici iz P vsebovani v dveh različnih množicah Q. Ker vsak komplement preostalih množic P ne more biti vsebovan v eni množici Q, mora biti vsaka preostala množica P vsebovana v eni podmnožici Q. Zatorej, $P = Q$. To je protislovje.

Predpostavimo zdaj, da so vse množice P vsebovane v množici Y iz Q. Potem je $Y = V$. Ponovno pridemo do protislovja.

Ker je unija dveh komplementnih množic iz P enaka V in Q vsebuje vsaj 3 množice, je lahko samo en komplement vsebovan v eni množici iz Q. Tako obstaja natanko ena množica X iz P, ki ni vsebovana v Y iz Q, ampak $\overline{X} \subset Y$.

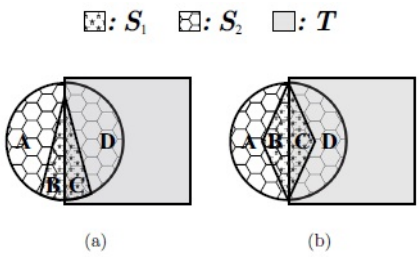
□

Lema 5.23 *Če so S_1, S_2 in S_3 paroma prekrižni prerezi, potem $S_1 \cap S_2 \cap S_3 = \emptyset$.*

Dokaz. Predpostavimo, da lema ne drži. Kot je prikazano v sliki 5.4, naj bodo

$$\begin{aligned} a &= w(S_3 \setminus (S_1 \cup S_2), \overline{(S_1 \cap S_2 \cap S_3)}) \\ b &= w((S_2 \cap S_3) \setminus S_1, S_2 \setminus (S_1 \cup S_3)) \\ c &= w(S_1 \cap S_2 \cap S_3, (S_1 \cap S_2) \setminus S_3) \\ d &= w((S_1 \cap S_3) \setminus S_2, S_1 \setminus (S_2 \cup S_3)). \end{aligned}$$

Po eni strani je $S_1 \cap S_2$ minimalni prerez (po Lemi 5.19(1)), tako da $c \geq \frac{\lambda}{2}$. Zatorej $b = d = 0$ in $(S_1 \cap S_3) \setminus S_2 = (S_2 \cap S_3) \setminus S_1 = \emptyset$.



Slika 5.5: Presek treh minimalnih prerezov

Če uporabimo Lemo 5.19 (2) na S_1 in S_2, potem $S_1 \cap S_2 \cap S_3$ in $S_3 \setminus (S_1 \cup S_2)$ niso povezani, kar je protislovje.

□

Lema 5.24 *Če so S_1, S_2 in T taki minimalni prerezi, da $S_1 \subset S_2$, $T \not\subset S_2$ in je $\langle S_1, T\rangle$ prekrižni prerez, potem so $A = (S_2 \setminus S_1) \setminus T$, $B = S_1 \setminus T$, $C = S_1 \cap T$ in $D = (S_2 \setminus S_1) \cap T$ minimalni prerezi in $w(A,B) = w(B,C) = w(C,D) = \frac{\lambda}{2}$ in $w(A,C) = w(A,D) = w(B,D) = 0$.*

Dokaz. Ker je $\langle S_1, T\rangle$ prekrižni prerez, je zato tudi $\langle S_2, T\rangle$ prekrižni prerez,
$w(A \cup B, C \cup D) = \frac{\lambda}{2}$
$w(B,C) = \frac{\lambda}{2}$,
$w(A,B) + w(B, \overline{S_1 \cup S_2}) = w(B, A \cup \overline{S1 \cup S_2}) = \frac{\lambda}{2}$ in
$w(A, \overline{S_1 \cup S_2}) = w(A \cup B, \overline{S_1 \cup S_2}) = \frac{\lambda}{2}$.
Vse enakosti sledijo iz Leme 5.19(3). Še več: $w(A, T \setminus S_2) = 0, w(D, \overline{S_1 \cup S_2}) = 0$ (po Lemi 5.19 (2)) in B, C sta minimalna prereza. Iz prvih dveh enačb in

$$w(A \cup B, C \cup D) = w(A,C) + w(A,D) + w(B,C) + w(B,D)$$

lahko sklepamo, da $w(A,C) = w(A,D) = w(B,D) = 0$.

Posledica tretje in četrte enačbe je, da $w(A, \overline{S_1 \cup S_2}) = w(A,B)$. Še več: $w(A,B) \geq \frac{\lambda}{2}$ (Lema 5.16) in $w(A, \overline{S_1 \cup S_2}) \leq w(A, \overline{S_1} \cup S_2) = \frac{\lambda}{2}$. Zatorej $w(A, \overline{S_1 \cup S_2}) = w(A,B) = \frac{\lambda}{2}$ in A je minimalni prerez.

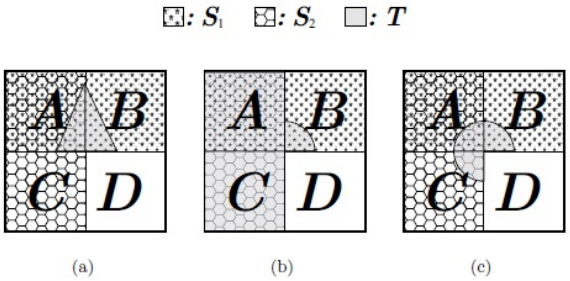
S podobnimi argumenti lahko vidimo, da $w(C,D) = \frac{\lambda}{2}$ in da je D minimalni prerez. Zato lahko spošni primer iz slike 5.5(a) vedno transformiramo v sliko 5.5(b).

□

Na kratko, če imamo dane neke množice $S_1, ...S_k$, naj bo

$$\mathcal{F}^{\alpha_1,\ldots,\alpha_k}_{S_1,\ldots,S_k} = \bigcap_{i=1}^{k} \left\{ \begin{array}{ll} S_i & \text{if } \alpha_i = 1 \\ \overline{S_i} & \text{if } \alpha_i = 0 \end{array} \right\} \text{ in } \mathcal{F}_{\{S_1,\ldots,S_k\}} = \left(\bigcup_{\alpha_1,\ldots,\alpha_k \in \{0,1\}^k} \mathcal{F}^{\alpha_1,\ldots,\alpha_k}_{\{S_1,\ldots,S_k\}} \right) \setminus \{\emptyset\}.$$

Lema 5.25 *Naj bo $\langle S_1, S_2\rangle$ prekrižni prerez in $A \in \mathcal{F}_{S_1,S_2}$. Izberimo tak $B \in \mathcal{F}_{S_1,S_2}$, da $w(A,B) = \frac{\lambda}{2}$. Za vse prekrižne prereze $\langle B, T\rangle$ velja:*
$w(A, B \cap T) = \frac{\lambda}{2}$ *ali* $w(A, B \cap \overline{T}) = \frac{\lambda}{2}$.



Slika 5.6: Minimalni prerez T in prekrižni prerez $\langle S_1, S_2\rangle$

Dokaz. Brez škode za splošnost: $A = S_1 \cap S_2$ (sicer zamenjamo S_1 in $\overline{S_1}$ ali pa S_2 in $\overline{S_2}$), $B = S_1 \setminus S_2$ (sicer zamenjamo S_1 in S_2). Naj bo $S = S_2 \setminus S_1$ in $D = \overline{S_1} \cap \overline{S_2}$. Potem: $w(B, C) = 0$ (Lema 5.19(2)).

Oglejmo si naslednje štiri primere:

Primer $T \subset (A \cup B)$**:** $w(A, B \cap T) = \frac{\lambda}{2}$, glej sliko 5.6(a) (Lema 5.24).
Primer $T \cap D \neq \emptyset$**:** Ker je $\langle S_1, T\rangle$ prekrižni prerez, sledi
$w(A \setminus T, A \cap T) + w(A \setminus T, B \cap T) + w(B \setminus T, A \cap T) + w(B \setminus T, B \cap T)$
$= w((A \setminus T) \cup (B \setminus T), (A \cap T) \cup (B \cap T))$
$= w(S_1 \setminus T, S_1 \cap T) = \frac{\lambda}{2}$.

Skupaj z $w(B \setminus T, B \cap T) \geq \frac{\lambda}{2}$ lahko zaključimo, da

- $w(A \setminus T, A \cap T) = 0$ in zato $A \cap T = \emptyset$ ali $A \setminus T = \emptyset$,
- $w(A \setminus T, B \cap T) = 0$ in
- $w(A \cap T, B \setminus T) = 0$.

Vemo, da $w(A, B) = \frac{\lambda}{2}$. Če $A \cap T = \emptyset$, potem $w(A, B \cap T) = 0$ in $w(A, B \setminus T) = \frac{\lambda}{2}$.

Sicer $A \setminus T = \emptyset$, $w(A, B \setminus T) = 0$ in $w(A, B \cap T) = \frac{\lambda}{2}$.

Primer $T \not\subset (A \cup B)$ **in** $T \cap D = \emptyset$ **in** $(A \cup C) \subset T$**:** glej sliko 5.6(b)

$$w(A, T \cap B) = w(A \cup C, T \cap B) = w((A \cup C) \cap T, T \setminus (A \cup C)) \geq \frac{\lambda}{2},$$

ker je $(A \cup C)$ minimalni prerez (Lema 5.16). Če uporabimo dejstvo, da $w(A, B) = \frac{\lambda}{2}$, dobimo $w(A, T \cap B) = \frac{\lambda}{2}$.

Primer $T \not\subset (A \cup B)$ **in** $T \cap D = \emptyset$ **in** $(A \cup C) \not\subset T$**:** glej sliko 5.6(c)

$$w(A, T \cap B) = w(A \cup C, T \cap B) = w(A \cup C, T \setminus (A \cup C)) = \frac{\lambda}{2},$$

ker je $\langle A \cup C, T\rangle$ prekrižni prerez.

□

Posledica 5.26 *Presek dveh prekrižnih prerezov razdeli točke osnovnega grafa na štiri minimalne prereze. Lema 5.19 (3) nam zagotavlja, da za vsakega od štirih minimalnih prerezov A obstajajo dva ali trije preostali taki minimalni prerezi B,C, da* $w(A,B) = w(A,C) = \frac{\lambda}{2}$. *Čeprav morda lahko množici B in C še naprej razdelimo na manjše dele s prekrižnimi prerezi, obstajata vedno dva disjunktna minimalna prereza* $X \subseteq B$ *in* $Y \subseteq C$ *z* $w(A,X) = w(A,Y) = \frac{\lambda}{2}$.

Dokaz. Predpostavimo, da posledica ne drži. Naj bo $\langle S, X_{1,2}\rangle$ prvi prekrižni prerez, ki razdeli množico $X_{1,2}$ z $w(A,X_{1,2}) = \frac{\lambda}{2}$ na dve disjunktni množici X_1, X_2 z $w(A,X_1)$, $w(A,X_2) \geq 0$. Potem je tudi $\langle S,B\rangle$ ali pa $\langle \overline{S},B\rangle$ prekrižni prerez, ki razdeli B na B_1 in B_2 z $X_1 \subseteq B_1$ in $X_2 \subseteq B_2$. Tako sta $w(A,B_1), w(A,B_2) \geq 0$. To je protislovno z lemo 5.25.

□

Izrek 5.27 *[149, 144] V grafu* $G = (V,E)$ *obstaja za vsako particijo P množice V na* 4 *disjunktne množice glede na prekrižni prerez v G krožna particija v G, ki je izpopolnitev P.*

Dokaz. Glede na dan prekrižni prerez $\langle S_1, S_2\rangle$, izberemo začetno množico $\Lambda := \{S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1, \overline{S_1 \cup S_2}\}$.

Dokler obstaja prekrižni prerez $\langle S,T\rangle$ za kakšen $T \notin \Lambda$ in $S \in \Lambda$, dodaj T v Λ. Ta proces se konča, ker lahko dodamo vsak $T \in \mathcal{P}(V)$ v Λ samo enkrat. Vse množice v Λ so minimalni prerezi. Λ ustreza definiciji 3(b).

Disjunktni minimalni prerezi $\mathcal{F}(\Lambda)$ nam dajo particijo grafa. Vse množice v $\mathcal{F}(\Lambda)$ lahko generiramo s prekrižnimi prerezi minimalnih prerezov v Λ. Zato ima vsaka množica v $\mathcal{F}(\Lambda)$ natanko dva soseda, t.j., za vsako množico $X \in \mathcal{F}(\Lambda)$ obstajata natanko dve taki različni množici $Y, Z \in \mathcal{F}(\Lambda)$, da $w(X,Y) = w(X,Z) = \frac{\lambda}{2}$ (Posledica 5.26). Za vse ostale množice $Z \in \mathcal{F}(\Lambda)$ je $w(X,Z) = 0$. Ker je G povezan graf, lahko vse množice iz $\mathcal{F}(\Lambda)$ uredimo tako, da definicija 5.20(a) drži.

Opazimo, da tudi definicija 5.20(b) še vedno drži, ker razbijanje množic v Λ na manjše množice še vedno dovoljuje rekonstrukcijo množic v Λ.

□

Lema 5.28 *Graf* $G = (V,E)$ *ima* $\mathcal{O}\left(\binom{|V|}{2}\right)$ *minimalnih prerezov in ta meja je natančna. To pomeni, da ima graf lahko* $\Omega\left(\binom{|V|}{2}\right)$ *minimalnih prerezov.*

Dokaz. Zgornja meja je posledica zadnjega izreka. Če imamo dan graf $G = (V,E)$, naslednja rekurzivna funkcija Z opiše število minimalnih prerezov v G:

$$Z(|V|) = \begin{cases} \sum_{i=1}^{k}(Z(|V_i|)) + \binom{k}{2} & \text{obstaja krožna particija } V_1,...,V_k \text{ v G} \\ Z(|S|) + Z(|V-S|) + 1 & \text{ni krožne particije, obstaja minimalni prerez } S \text{ v } G \\ 0 & \text{sicer} \end{cases}$$

Lahko je videti, da ta funkcija doseže maksimum v primeru, ko krožna particija $W_1,...,W_{|V|}$ obstaja. Zato $Z(|V|) = \mathcal{O}\left(\binom{|V|}{2}\right)$.

Spodnjo mejo dosežemo s preprostim ciklom n točk. Imamo $\Omega\left(\binom{n}{2}\right)$ parov povezav. Vsak par povezav definira nova dva minimalna prereza S in $\overline{S}$. Ti dve množici sta ločeni, če preprosto odstranimo par povezav.

□

5.5 Predstavitev minimalnih prerezov s kaktusom

V naslednjih vrsticah je podan opis kaktusa. Najprej si oglejmo graf $G = (V, E)$ brez krožnih particij. Ker ne obstaja prekrižni prerez, so vsi minimalni prerezi G laminarni.

Množica $\mathcal{S}$ se imenuje *laminarna*, če za vsak par množic $S_1, S_2 \in \mathcal{S}$ velja, da sta ali S_1 in S_2 disjunktni ali pa je S_1 vsebovana v S_2 (oz. obratno). Tako ima vsaka množica $T \in \mathcal{S}$, ki je vsebovana v neki $S_1, S_2, \ldots \in \mathcal{S}$, enolično najmanjšo nadmnožico. Vsako laminarno množico $\mathcal{S}$ lahko predstavimo kot drevo. Od tu dalje zaradi boljše preglednosti pravimo, da ima drevo vozle in liste, medtem ko ima graf točke. Vsak vozel prestavlja eno množico v $\mathcal{S}$; listi predstavljajo množice v $\mathcal{S}$, ki ne vsebujejo nobene druge množice iz $\mathcal{S}$. Oče vozla, ki predstavlja množico T, predstavlja najmanjšo nadmnožico T. Ta konstrukcija se konča z množico dreves, ki se ji reče gozd. Dodamo dodaten vozel r v gozd in povežemo vse korene dreves gozda s tem novim vozlom r, ki je sedaj novi koren enega velikega drevesa. Zato vozli enega drevesa predstavljajo vse množice $\mathcal{S}$ in koren drevesa predstavlja celotno osnovno množico, t.j. unijo vseh elementov vseh $S \in \mathcal{S}$. Če ima ta unija n elementov, potem ima tako drevo lahko največ n listov in zato največ $2n - 1$ vozlov.

Ker so vsi minimalni prerezi G laminarni, lahko te predstavimo z drevesom T_G sledeče: vzemimo najmanjšo množico točk vsakega minimalnega prereza. Označimo to množico množic z Λ. Če so množice točk minimalnega prereza enake velikosti, vzamemo eno izmed teh množic. Predstavimo vsako množico iz Λ z enim samim vozlom. Dva vozla, ki pripadata minimalnim prerezom A in B v G, sta povezana s povezavo, če $A \subset B$ in ne obstaja noben drug C, da $A \subset C \subset B$. Koreni drevesa predstavljajo minimalne prereze v Λ, ki niso vsebovani v nobenem drugem minimalnem prerezu v Λ. Spet, povežemo vse korene gozda s točko, ki jo definiramo kot koren drevesa.

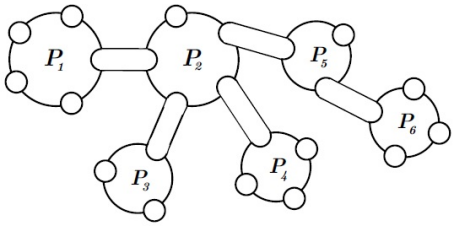
Ker z odstranitvijo ene povezave v drevesu ločimo pod-drevo od ostalega drevesa, definiramo naslednjo preslikavo: vsako točko grafa G preslikamo v vozel drevesa T_G, ki pripada najmanjšemu prerezu, ki vsebuje to točko. Vse točke, ki niso vsebovane v nobenem vozlu T_G, so preslikane v koren drevesa T_G.

Za vsak minimalni prerez S v G, točke iz S potem preslikamo v tako množico vozlov X, da obstaja povezava in če odstranimo to povezavo, ločimo vozle X od preostalega drevesa. Obratno, če odstranimo eno povezavo iz T_G, s tem ločimo vozle drevesa na dva taka dela, da je množica vseh točk, preslikanih v en del, minimalni prerez.

Če G nima krožne particije, je drevo T_G *kaktus* C_G za G. Število vozlov kaktusa je omejeno z $2|V| - 1$.

Oglejmo si graf $G = (V, E)$, ki ima samo eno krožno particijo $V_1, \ldots V_k$. Prereze krožne particije lahko predstavimo s ciklom k vozlov. Za $1 \leq i \leq k$ so točke vsakega dela V_i predstavljene z enim vozlom N_i iz cikla na tak način, da sta dva dela V_i in V_{i+1} predstavljena z dvema sosednjima vozloma.

Zdaj uporabimo dejstvo, da je za vsak minimalni prerez S, ki ni prerez krožne particije, ali S ali $\overline{S}$ prava podmnožica V_i. Zato lahko konstruiramo drevo $T_{(V_t,E)}$ za vse minimalne prereze, ki so podmnožica V_i, ampak tokrat z omejitvijo, da so samo točke V_i preslikane na to drevo. Koren drevesa $T_{(V_t,E)}$ pripada natanko množici V_i. Na ta način lahko spojimo vozel N_i iz cikla in koren $T_{(V_t,E)}$ za vse $1 \leq i \leq k$. Ta cikel, povezan z vsemi drevesi, je kaktus C_G za G. Število vozlov je enako vsoti vseh vozlov v drevesih $T_{(V_t,E)}$ z $1 \leq i \leq k$.



Slika 5.7: Kaktus, ki predstavlja prereze krožne particije šestih krožnih particij.

Zato je število vozlov kaktusa omejeno z $2|V| - 1$ in spet, obstaja $1 - 1$ korespondenca med minimalnimi prerezi v G in ločitvijo C_G na dva dela.

Zdaj si oglejmo graf $G = (V, E)$ s krožnimi particijami $P_1, \ldots, P_z$. Vzamemo vse krožne particije kot množico množic. Konstruiramo kaktus C_G, ki predstavlja prereze krožnih particij G na naslednji način: točke vsake množice $F \in \mathcal{F}_{P_1 \cup \cdots \cup P_z}$ preslikamo v en vozel in dva vozla sta povezana, če za njuni pripadajoči množici F_1 in F_2 velja, da $w(F_1, F_2) \geq 0$. Potem vsaka krožna particija ustvari en cikel v C_G. Ker so vse krožne particije paroma kompatibilne, so cikli povezani s povezavami, ki niso del nobenega cikla. Kaktus C_G je zdaj graf, ki je podoben drevesu (slika 5.7).

Ko predstavimo preostale minimalne prereze, ki niso del krožne particije, dobimo kaktus T_C za G. Kot prej, število vozlov kaktusa je omejeno z $2|V| - 1$.

5.6 Algoritmi na grafih: povezanost na podlagi pretoka

Razlikujemo algoritme, ki preverjajo, če je graf G k-povezan (po točkah ali povezavah) za določen $k \in \mathbb{N}$ in algoritme, ki glede na dan graf izračunajo pripadajočo povezanost grafa $\kappa(G)$ ali povezanost po povezavah grafa $\lambda(G)$. Večina algoritmov, ki izračunavajo povezanost grafa po točkah ali povezavah temelji na izračunu maksimalnega pretoka skozi dano omrežje.

Definicija 5.29 Omrežju pravimo *omrežje z enotno kapaciteto* (ali $0 - 1$ *omrežje*), če imajo vse povezave kapaciteto 1. Omrežje z enotno kapaciteto je *tipa 1*, če ne vsebuje vzporednih povezav. Pravimo, da je *tipa 2*, kadar sta za vsako točko v, ($v \neq s$, $v \neq t$) vhodna stopnja $d^-(v)$ ali izhodna stopnja $d^+(v)$ enaki 1.

Lema 5.30 *1. Za omrežja z enotno kapaciteto je časovna zahtevnost izračuna maksimalnega pretoka z uporabo Dinitzovega algoritma enaka $\mathcal{O}(m^{3/2})$.*

2. Za omrežja z enotno kapaciteto tipa 1 je časovna zahtevnost izračuna maksimalnega pretoka z uporabo Dinitzovega algoritma enaka $\mathcal{O}(mn^{2/3})$.

3. Za omrežja z enotno kapaciteto tipa 2 je časovna zahtevnost izračuna maksimalnega pretoka z uporabo Dinitzovega algoritma enaka $\mathcal{O}(mn^{1/2})$.

Za dokaz leme glej [150, 151, 152].

5.6.1 Algoritmi na grafih: povezanost po točkah

Osnova vseh algoritmov na grafih, ki preverjajo povezanost na podlagi pretoka, je funkcija, ki izračunava lokalno povezanost med dvema različnima točkama s in t. Even je predstavil metodo za izračun $\kappa_G(s,t)$, ki temelji na naslednjem principu: Iz danega grafa $G=(V,E)$, ki ima n točk in m povezav, konstruiramo usmerjen graf $\overline{G}=(\overline{V},\overline{E})$ z $|\overline{V}|=2n$ in $|\overline{E}|=2m+n$ tako, da vsako točko $v\in V$ zamenjamo z dvema točkama $v',v''\in\overline{V}$, povezanima z (notranjo) povezavo $e_v=(v',v'')\in\overline{E}$. Vsako povezavo $e=(u,v)\in E$ zamenjamo z dvema (zunanjima) povezavama $e'=(u'',v'),e''=(v'',u')\in\overline{E}$.

$\kappa(s,t)$ sedaj izračunamo kot maksimalni pretok grafa $\overline{G}$ od izhodišča s'' do konca t' z enotnimi kapacitetami na vseh povezavah. Za vsak par $v',v''\in\overline{V}$, ki predstavlja točko $v\in V$, je notranja povezava (v',v'') edina povezava, ki izvira iz v' in edina, ki vstopa v v'', zato je $\overline{G}$ omrežje tipa 2. Iz Leme 5.30 sledi, da je časovna zahtevnost izračuna maksimalnega pretoka v tovrstnem grafu enaka $\mathcal{O}(m\sqrt{n})$.

Trivialen algoritem za računanje $\kappa(G)$ bi enostavno izračunal minimum lokalnih povezanosti po točkah za vse pare točk. Ker je $\kappa_G(s,t)=n-1$ za vse pare (s,t), ki so direktno povezani, bi tak algoritem $\big(\frac{n(n-1)}{2}-m\big)$-krat izvršil fukncijo, ki preverja povezanost na podlagi pretoka (MaxFlow funkcijo). V nadaljevanju bomo videli, da se da takšno časovno zahtevnost še izboljšati.

Naj bo S minimalna prerezna množica točk, $S\subset V$, ki ločuje "levo" podmnožico točk $L\subset V$ od "desne" $R\subset V$. Potem lahko izračunamo $\kappa(G)$ tako, da eno izmed točk, na primer s, fiksiramo v eno izmed podmnožic L ali R, in izračunamo njeno lokalno povezanost $\kappa_G(s,t)$ z vsemi točkami $t\in V\backslash\{s\}$, ki ležijo v nasprotni podmnožici oziroma na drugi strani točkovnega prereza. Pri tem imamo naslednjo težavo: kako izbrati točko s, da ne bo pripadala vsaki minimalni prerezni množici točk? Ker je $\kappa(G)\le\delta(G)$ (za vse netrivialne grafe G namreč velja: $\kappa(G)\le\lambda(G)\le\delta(G)$), lahko s izbiramo med $\delta(G)+1$ točkami in ena izmed njih ne sme biti del vseh minimalnih prereznih množic točk. Tak algoritem bi imel časovno zahtevnost $\mathcal{O}((\delta+1)nm\sqrt{n})=\mathcal{O}(\delta mn^{3/2})$.

Even in Tarjan sta predlagala Algoritem 5, ki preneha z računanjem lokalnih povezanosti, če velikost minimalnega prereza pade pod število pregledanih točk.

Tak algoritem v zanki, ki teče po spremenljivki i, ne pregleda več kot $\kappa+1$ točk. Vsaka točka ima vsaj $\delta(G)$ sosedov, zato je časovna zahtevnost funkcije maksimalnega pretoka največ $\mathcal{O}((n-\delta-1)(\kappa+1))$. Ker je po izreku $\kappa(G)\le 2m/n$, dobimo minimalno kapaciteto najkasneje po $2m/n+1$ izvršitvah funkcije. Od tod sledi, da je skupna časovna zahtevnost algoritma $\mathcal{O}(m^2\sqrt{n})$.

Esfahanian in Hakimi sta ta algoritem še izboljšala z uporabo naslednje leme [153].

Lema 5.31 *Če točka v pripada vsem minimalnim prereznim množicam točk, potem obstajata za vsak minimalni prerez po točkah S dve točki $l\in L_S$ in $r\in R_S$, ki sta sosednji z v.*

Dokaz. Predpostavimo, da je v vsebovana v vseh minimalnih prereznih množicah točk grafa G. Imejmo particijo množice točk V, ki je inducirana z minimalno prerezno množico točk S s komponentama L ("leva" stran) in pripadajočo "desno" stranjo R. Vsaka stran mora vsebovati vsaj enega od sosedov točke v, sicer točka v ne bi bila potrebna za razbitje grafa na dva dela. Pravzaprav mora vsaka stran, ki ima več kot eno točko, vsebovati dva

Algorithm 5 Even & Tarjan: povezanost po točkah

Vhod: (Neusmerjen) graf $G = (V, E)$
Izhod: $\kappa(G)$

```
κ_min ← n − 1
i ← 1
while i ≤ κ_min do
    for j ← i + 1 TO n do
        if i > κ_min then
            break
        else if {v_i, v_j} ∉ E then
            izračunaj κ_G(v_i, v_j) z uporabo MaxFlow algoritma
            κ_min ← min{κ_min, κ_G(v_i, v_j)}
        end if
    end for
end whilereturn κ_min
```

soseda, sicer bi z zamenjavo točke v z edinim sosedom dobili minimalni prerez brez v, kar pa je v protislovju s predpostavko.

□

Z upoštevanjem zgornjih premislekov dobimo Algoritem 6. Prva zanka $\big(n-\delta-1\big)$-krat izvrši MaxFlow funkcijo, drugo zanko algoritem izvrši $\big(\kappa(2\delta-\kappa-3)/2\big)$-krat. Skupna časovna zahtevnost algoritma je torej $n-\delta-1+\kappa(2\delta-\kappa-3)/2$.

5.6.2 Algoritmi na grafih: povezanost po povezavah

Podobno kot pri računanju povezanosti po točkah je osnova za računanje povezanosti po povezavah algoritem maksimalnega pretoka, ki reši lokalen problem povezanosti po povezavah, torej izračuna $\lambda_G(s,t)$. Vse neusmerjene povezave zamenjamo s pari nevzporednih usmerjenih povezav s kapaciteto 1 in izračunamo maksimalni pretok od izhodišča s do konca t. Dobimo omrežje tipa 1 (ker ne vsebuje vzporednih povezav), zato je po Lemi 5.30 časovna zahtevnost tega izračuna $\mathcal{O}(\min\{m^{3/2}, mn^{2/3}\})$.

Trivialen algoritem za računanje $\lambda(G)$ bi enostavno izračunal minimum lokalnih povezanosti po povezavah za vse pare točk. Tak algoritem bi $\big(n(n-1)/2\big)$-krat izvršil funkcijo MaxFlow. Časovno zahtevnost lahko izboljšamo tako, da obravnavamo le lokalne povezanosti $\lambda_G(s,t)$ za neko fiksno točko s in vse ostale točke t. Ker mora biti ena od točk $t \in V\backslash\{s\}$ ločena od s z minimalnim prerezom po povezavah, je $\lambda(G)$ enaka minimumu vseh teh vrednosti. Zato je število izvršitev funkcije MaxFlow enako $n-1$ in skupna časovna zahtevnost je enaka $\mathcal{O}(nm\ \min\{m^{1/2}, n^{2/3}\})$. Prej omenjeni algoritem deluje tudi v primeru, če celo množico točk zamenjamo s podmnožico, ki vsebuje dve točki, ki sta ločeni z minimalnim prerezom po povezavah. Naslednji algoritmi želijo posledično zmanjšati velikost zgornje množice točk (ki se imenuje λ-*pokritje*). Pri tem izkoriščajo naslednjo lemo.

Lema 5.32 *Naj bo S minimalni prerez po povezavah grafa $G = (V, E)$ in naj bo $L, R \subset V$ particija množice točk, taka, da sta L in R ločeni s S. Če je $\lambda(G) < \delta(G)$, potem vsaka komponenta $G - S$ vsebuje več kot $\delta(G)$ točk, t.j. $|L| > \delta(G)$ in $|R| > \delta(G)$.*

Algorithm 6 Esfahanian & Hakimi: povezanost po točkah

Vhod: (Neusmerjen) graf $G = (V, E)$
Izhod: $\kappa(G)$

```
κ_min ← n − 1
Izberi v ∈ V z najmanjšo stopnjo, d(v) = δ(G)
Označi sosede N(v) z v_1, v_2, ..., v_δ
for all nesosednji w ∈ V\(N(v) ∪ {v}) do
    izračunaj κ_G(v, w) z uporabo MaxFlow algoritma
    κ_min ← min{κ_min, κ_G(v, w)}
end for
i ← 1
while i ≤ κ_min do
    for j ← i + 1  TO  δ − 1 do
        if i ≥ δ − 2  OR  i ≥ κ_min then return κ_min
        else if {v, w} ∉ E then
            izračunaj κ_G(v_i, v_j) z uporabo MaxFlow algoritma
            κ_min ← min{κ_min, κ_G(v_i, v_j)}
        end if
    end for
    i ← i + 1
end whilereturn κ_min
```

Dokaz. Elemente L označimo z $\{l_1, l_2, \ldots, l_k\}$, inducirane povezave pa z $E[L] = E(G[L])$.

$$k\,\delta(G) \leq \sum_{i=1}^{k} d_G(l_i)$$

$$\leq 2 \cdot |E[L]| + |S|$$

$$\leq 2 \cdot \frac{k(k-1)}{2} + |S|$$

$$< k(k-1) + \delta(G)$$

Iz $\delta(G) \cdot (k-1) < k(k-1)$ sledi, da je $|L| = k > 1$ in $|L| = k > \delta(G)$ (prav tako tudi $|R| > \delta(G)$).

□

Posledica 5.33 *Če je $\lambda(G) < \delta(G)$, potem vsaka komponenta $G - S$ vsebuje točko, ki ni incidenčna točka nobene povezave v S.*

Lema 5.34 *Spet predpostavimo, da je $\lambda(G) < \delta(G)$. Če je T vpeto drevo grafa G, potem vsaka komponenta $G - S$ vsebuje vsaj eno točko, ki ni list drevesa T (t.j. točke, vsebovane v T, ki niso listi, tvorijo λ-pokritje).*

Zgornja lema predlaga algoritem, ki najprej izračuna vpeto drevo danega grafa, potem naključno izbere neko notranjo točko drevesa, v, in izračuna lokalno povezanost $\lambda(v, w)$

do vsake druge točke w, ki ni list. Minimum teh vrednosti nam skupaj z $\delta(G)$ da točno povezanost po povezavah $\lambda(G)$.

Algorithm 7 Esfahanian & Hakimi: računanje vpetega drevesa

Vhod: (Neusmerjen) graf $G = (V, E)$
Izhod: Vpeto drevo T z listom in notranjo točko v L in R
Izberi $v \in V$
$T \leftarrow$ vse povezave, ki so incidenčne točki v
while $|E(T)| < n - 1$ **do**
izberi list w v T, tako da je za vse liste r v T: $|N(w) \cap (V - V(T))| \geq |N(r) \cap (V - V(T))|$
$T \leftarrow T \cup G[w \cup \{N(w) \cap (V - V(T))\}]$
end whilereturn T

Esfahanian in Hakimi [153] sta predlagala algoritem za računanje vpetega drevesa T grafa G, tako da obe strani, L in R, nekega minimalnega prereza po povezavah, vsebujeta vsaj en list drevesa T, in zaradi Leme 5.34 tudi vsaj eno notranjo točko. Povezanost po povezavah grafa je potem izračunana s pomočjo Algoritma 8. Ker je P izbrana tako, da je manjša izmed množice listov in množice ne-listov, mora algoritem vsaj $\binom{n/2}{}$-krat izvršiti funkcijo, ki računa lokalno povezanost. Skupna časovna zahtevnost znaša $\mathcal{O}(\lambda mn)$.

Algorithm 8 Esfahanian & Hakimi: povezanost po povezavah

Vhod: (Neusmerjen) graf $G = (V, E)$
Izhod: $\lambda(G)$
Konstruiraj vpeto drevo T z uporabo Algoritma 3
P naj označuje manjšo izmed obeh množic, listov ali notranjih vozlišč T
Izberi točko $u \in P$
$c \leftarrow min\{\lambda_G(u, v) : v \in P \backslash \{u\}\}$
$\lambda \leftarrow min(\delta(G), c)$ **return** λ

Definicija 5.35 Naj bo G graf z množico vozlišč $V(G)$ in množico povezav $E(G)$. Množica $S \subseteq V(G)$ je *dominantna množica*, če je za vsako vozlišče $v \in V(G)$ bodisi $v \in S$, bodisi obstaja tako vozlišče $u \in S$, da je $vu \in E(G)$. Povedano drugače, vsako vozlišče je bodisi v dominantni množici, bodisi ima soseda v dominantni množici. Dominantna množica z najmanjšo možno močjo je *najmanjša dominantna množica*.

Časovno zahtevnost Algoritma 4 bi lahko izboljšal Matula [154], ki je upošteval naslednjo lemo.

Lema 5.36 *V primeru, da je $\lambda(G) < \delta(G)$, je vsaka dominantna množica grafa G tudi njegovo λ-pokritje.*

Podobno kot pri vpetem drevesu, lahko povezanost po povezavah izračunamo tako, da izberemo množico D, ki dominira G, naključno izberemo točko $u \in D$ in izračunamo lokalne povezanosti po povezavah med točko u in vsemi ostalimi točkami v D. Minimum

vseh vrednosti skupaj z minimalno stopnjo $\delta(G)$ nam da rezultat. Problem iskanja najmanjše dominantne množice je sicer NP-težak, da pa se pokazati, da je lahko časovna zahtevnost zgornjega algoritma enaka $\mathcal{O}(nm)$, če je dominantna množica izbrana s pomočjo Algoritma 9.

Algorithm 9 Računanje dominantne množice

Vhod: (Neusmerjen) graf $G = (V, E)$
Izhod: Dominantna množica D
  Izberi $v \in V$
  $D \leftarrow \{v\}$
  while $V \backslash (D \cup N(D)) \neq \emptyset$ **do**
    Izberi točko $w \in V \backslash (D \cup N(D))$
    $D \leftarrow D \cup \{w\}$
  end whilereturn D

5.7 Algoritmi, ki ne temeljijo na pretoku

V tem poglavju bomo obravnavali algoritme povezanosti, ki ne temeljijo na pretoku skozi omrežje.

5.7.1 Algoritem: Minimalni prerez (Stoer in Wagner)

Je zelo preprost algoritem v primerjavi s prej omenjenimi. Podoben je Primovem algoritmu za iskanje minimalnega vpetega drevesa in Dijkstri, ki izračuna najkrajšo pot v grafu. Zato je tudi časovna zahtevnost enaka - za vsako fazo algoritma znaša $\mathcal{O}(m + n \log n)$, kar vodi do skupne časovne zahtevnosti algoritma $\mathcal{O}(nm + n^2 \log n)$.

Po tem, ko naključno izbere začetno točko a, algoritem vzdržuje podmnožico točk A, ki na začetku vsebuje le začetno točko in se povečuje, ko v njo na vsakem koraku doda točko $v \notin A$, ki ima največjo vsoto uteži po vseh povezavah do točk v A. Ko so vse točke dodane v A, zadnji dve točki, s in t, združi v eno. Medtem ko se povezave med s in t med krčenjem preprosto izbrišejo, pa vse ostale povezave med s in t in ostalimi točkami zamenja s povezavo, uteženo z vsoto uteži prejšnjih povezav. Prerez, ki ločuje točko, ki je bila zadnja dodana, od preostalega grafa, se imenuje fazni prerez ("cut-of-the-phase").

Lema 5.37 *Fazni prerez je minimalen $s-t$-prerez v modificiranem grafu, kjer sta s in t zadnji točki, ki ju dodamo v A.*

Izrek 5.38 *Fazni prerez, ki ima minimalno težo med vsemi faznimi prerezi, je prerez prvotnega grafa z minimalno kapaciteto.*

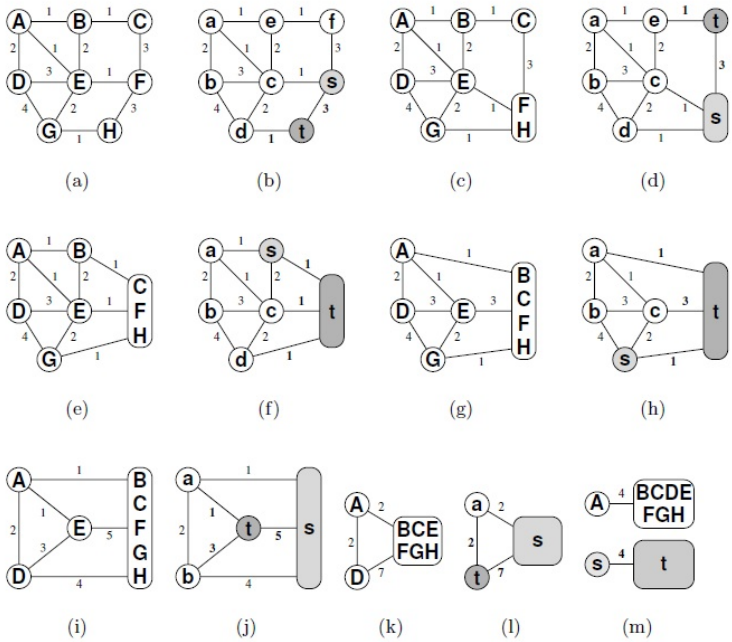
Dokaz. V primeru, ko graf vsebuje samo dve točki, je dokaz trivialen. Zato predpostavimo, da je $|V| > 2$. Ločimo dva primera.

1. Graf ima prerez z minimalno kapaciteto, ki je hkrati minimalni s-t-prerez (kjer sta s in t točki, ki ju zadnji dodamo v fazo). Potem iz Leme 5.37 sledi, da je to prerez z minimalno kapaciteto originalnega grafa.

Algorithm 10 Stoer & Wagner: računanje minimalnega prereza

Vhod: Neusmerjen graf $G = (V, E)$
Izhod: Minimalni prerez C_{min}, ki ustreza $\lambda(G)$

```
Naključno izberi začetno točko a
C_min ← nedefiniran
V' ← V
while |V'| > 1 do
    A ← {a}
    while A ≠ V' do
        A dodaj točko z največjo stopnjo
        Prilagodi kapacitete med A in točkami V'\A
    end while
    C := prerez V', ki ločuje zadnjo dodano točko v A od preostalega grafa
    if C_min = nedefiniran OR w(C) < w(C_min) then
        C_min ← C
    end if
    Združi točki, ki sta bili zadnji dodani v A
end whilereturn C_min
```



Slika 5.8: Primer Stoer/Wagner algoritma. Minimalni prerez $\{ABDEG\}|\{CFH\}$ ima kapaciteto 3 in ga najdemo v točki f).

2. Graf ima minimalni prerez, kjer sta s in t na isti strani prereza. Če torej združimo točki s in t, to ne vpliva na prerez z minimalno kapaciteto.

Z indukcijo po točkah tako dokažemo, da je prerez z minimalno kapaciteto fazni prerez z najmanjšo težo.

□

5.8 2-povezane komponente

Srečamo se z vprašanjem katera vozlišča vedno ostanejo povezana v omrežju v primeru, da odstranimo poljubno vozlišče. Težava je pravzaprav računanje 2-povezanih (ali neločljivih) komponent grafa, ki jim pravimo tudi bloki. Najprej uporabimo metodo pregleda grafa v globino na neusmerjenem povezanem grafu $G = (V, E)$. V metodi obiskane točke zaporedno, kakor jih obiskujemo, oštevilčimo s številkami od 1 do $n = |V|$, to številčenje oziroma vrsto, pa poimenujemo num. Opazimo, da lahko obiščemo dve različni vrsti točk, take ki vodijo do še ne označenih novih točk ter take do katerih povezave vodijo do že označenih točk. Slednjim povezavam, takim ki vodijo od novo obiskanih točk do že obiskanih in označenih točk bomo rekli *nazajšnja povezava*. Za vsako točko v ohranimo najmanjšo oznako neke točke, ki je dosegljiva preko poljubne drevesne povezave in nima več kot ene nazajšnje povezave, na primer najmanjša številka neke točke, ki leži na istem ciklu. Kadarkoli odkrijemo novo točko z metodo pregleda v globino je oznaka low inicializirana z novo številko. Če se vrnemo od potomca do otroka w, na primer preko drevesne povezave (v, w), se vrednost low(v) spremeni na minimum vrednosti otroka (low$[w]$) ter prejšnje oznake (low$[v]$). V primeru, da najdemo nazajšnjo povezavo (v, w), spremenimo vrednost low$[v]$ na minimum njene prejšnje vrednosti ter oznake od w. Za odkrivanje presečnih točk v grafu lahko sedaj uporabimo naslednjo lemo.

Lema 5.39 *Sledimo zgoraj opisani metodi za izračunavanje vrednosti low in num z metodo pregleda v globino po grafu G. Točka v je prerezna točka natanko tedaj, ko velja eden izmed spodnjih pogojev:*

1. *če je v začetek drevesa, najdenega z metodo pregleda v globino in je vsebovan v vsaj dveh povezavah drevesa najdenega z metodo pregleda v globino,*

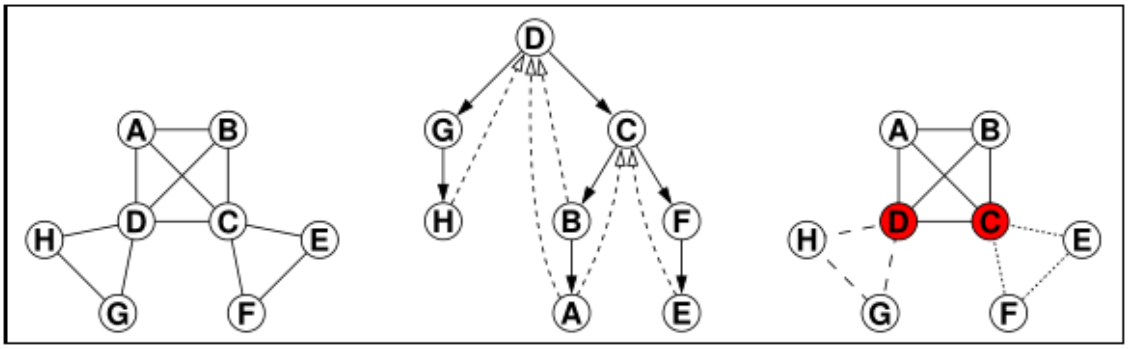
2. *če v ni začetek drevesa, vendar obstaja otrok w točke v tak, da velja* $low[w] \geq num[v]$.

Dokaz. 1. Predpostavimo, da je točka v koren drevesa najdenega z metodo pregleda v globimo.
→ Če imamo iz v več drevesnih povezav, bi z odstranitvijo točke v iz grafa G povezave med njenimi otroki razpadle.
← Če je v prerezna točka potem obstajata točki $x, y \in V$, ki postaneta z odstranitvijo točke v nepovezani. Točka v na vsaki poti povezuje x in y. Točko y lahko pravzaprav odkrijemo še le ko se vrnemo nazaj v točko v. Iz tega sledi, da ima točka v vsaj dva otroka v DFS drevesu.
2. Predpostavimo, da točka v ni koren DFS drevesa.
→ Potem obstaja otrok w od točke v in ni taka točka, da velja $low[w] > num[v]$. To pomeni, da obstaja le ena pot, ki povezuje potomca w z vsemi predniki točke v. Torej je

v prerezna točka.
$\leftarrow$ Če je v prerezna točka potem obstajata točki $x, y \in V$ in za poti med njima velja, da vedno vsebujejo točko v. Če bi imeli vsi otroci točke v posredno povezavo (preko poljubnega drevesa) do nekega prednika točke v bi bil graf povezan. Torej mora obstajati otrok za katerega velja $low[w] \geq num[v]$.

□

Da bi našli 2-povezane komponente npr. množico povezav, dodamo vsako novo povezavo na kup povezav. Kadarkoli pogoj $low[w] \geq num[v]$ velja tudi po vrnitvi rekurzivnega klica za otroka w od v sledi, da povezave na vrhu kupa vključno z povezavo (v,w) tvorijo nov blok in jih zato odstarnimo s kupa povezav.



Izračunanje 2-povezanih komponent v neusmerjenem grafu.
Levo: neusmerjen začetni graf Sredina: DFS drevo s polnimi povezavami in črtastimi povratnimi povezavami Desno: bloki grafa

5.9 Močno povezane komponente

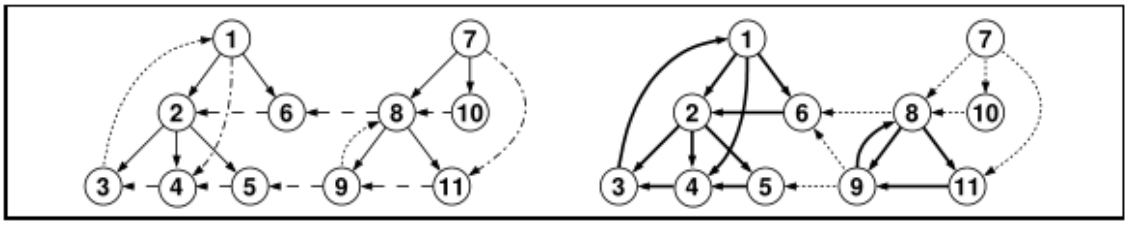
Sedaj premislimo kako bi izračunali močne komponente npr. najmočneje povezane podgrafe v usmerjenih grafih. Podobno je tudi računanje 2-povezanih komponent v neusmerjenih grafih. Tu uporabimo modificirano metodo pregleda v globino, ki označuje točke z zaporednimi števili od 1 do n. V primeru, da se pregled konča ne da bi odkril vse točke v grafu moramo postopek DFS na novo zagnati v eni od še neoznačenih točk. Naš rezultat je v tem primeru gozd F. Povezave $e = (v, w)$, ki jih odkrijemo z metodo pregleda v globino lahko razdelimo v naslednje kategorije:

1. Vse povezave, ki vodijo do neoznačenih točk imenujemo »drevesne povezave« (vsebovane so v drevesih gozda odkritega z metodo pregleda v globino).

2. Povezave, ki vodijo do točke w, ki je bila že označena v prejšnjem koraku metode lahko razdelimo v naslednje razrede:

   - če je $num[w] > num[v]$ rečemo, da je povezava e »naprejšnja povezava« (forward edge),
   - če je w sorodna od v v drevesu najdenem z pregledom v globino (metoda DFS) rečemo povezavi e povratna povezava (backward edge),

- drugače pa povezavo e imenujemo »križna povezava« (cross edge), saj je usmerjena od enega poddrevesa k drugemu.



DFS gozd za računanje močno povezanih komponent v povezanih grafih: drevo, naprejšnje, povratne in križne povezave.

Točki v in w sta v isti močni komponenti natanko tedaj ko obstaja direktna pot od točke v do točke w ter direktna pot od w do v. To označuje enakovrednostno razmerje ter vsebovanost točke v nizu (v primerjavi z 2-povezanostjo komponent, ker je povezava del ene komponente, medtem ko je ena točka lahko del več različnih komponent).

Med DFS (depth-first search) sprehodom smo poiskali korene (roots) močno povezanih komponent (v vsaki komponenti točka, ki ima najmanjšo oznako). Kot v primeru 2-povezanih komponent moramo pogledati za vsakega potomca w točke v ali obstaja tudi povezava v obratni smeri, torej od w do v. Sedaj lahko definiramo $lowlink[v]$, ki naj bo najmanjša oznaka neke točke v isti močni komponenti, ki je lahko dosegljiva po poljubno mnogo povezavah med drevesi in ima največ eno povratna ali križno povezavo.

Lema 5.40 *Točka v je koren močne komponente natanko tedaj, ko veljata oba naslednja pogoja:*

1. *Ne obstaja povratna povezava od v ali potomca v do prednika točke v.*

2. *Ne obstaja nobena križna povezava (v, w) od točke v ali njenih potomcev do točke w, da bi bil koren močne komponente točke w prednik točke v.*

To je ekvivalentno ko je $lowlink[v] = num[v]$.

Dokaz. $\rightarrow$ Predpostavimo, da enakost velja, vendar naj bo u koren od $v$jeve močne komponente in $u \neq v$. Obstajati mora direktna pot od v do u. Prva povezava na poti, ki vodi do točke w in ni potomec v v DFS grevesu je povratna ali križna povezava. To pomeni, da je $lowlink[v] \leq num[w] < num[v]$, saj je skupni prednik točk v in w z največjo oznako tudi v tej močni komponenti.
$\leftarrow$ Če je v koren neke močne komponente iz nekega gozda lahko rečemo, da velja $lowlink[v] = num[v]$. Če sedaj predpostavimo na primer, da je $lowlink[v] < num[v]$, bi to pomenilo, da je v tej močni komponenti tudi neki pravi prednik točke v. To pa bi pomenilo, da v ni koren te močne komponente.

□

5.10 3-povezane komponente

Definicija 5.41 Naj bo $G = (V, E)$ 2-povezan (multi)graf. Točki $a, b \in V$ imenujemo ločevalni par (separation pair) grafa G, če podgraf iz točk $V \setminus \{a, b\}$ ni povezan.

Par (a, b) deli povezave grafa G v ekvivalenčne razrede $E_1, \ldots, E_k$ (ločevalni razredi – separation classes). Dve povezavi sta iz istega razreda natanko tedaj, ko obe ležita na isti poti p, ki ne vsebuje niti a niti b kot ene izmed notranjih točk poti. Če vsebuje točko a ali b je ta točka robna točka poti p. Par (a, b) je ločevalni par če imamo vsaj dva ločena razreda razen v naslednjih posebnih primerih:

- imamo natanko dva ločna razreda in eden izmed njiju je sestavljen natanko iz ene povezave
- imamo tri ločene razrede in vsak izmed njih je sestavljen iz natanko ene povezave.

Graf G je 3-povezan če ne vsebuje nobenega ločevalnega para.

Definicija 5.42 Naj bo (a, b) ločevalni par dvojno povezanega multi grafa G in naj bodo ločevalni razredi $E_{1,\ldots,k}$ razdeljeni v dve skupini $E' = \cup_{i=1}^{l} E_i$ in $E'' = \cup_{i=l+1}^{k} E_i$ ter naj vsebujejo vsaj dve povezavi. Grafa $G' = (V(E' \cup e), E' \cup e)$ in $G'' = (V(E'' \cup e), E'' \cup e)$, ki sta posledica razdelitve grafa na dve skupini povezav $[E', E'']$ in dodatne nove izmišljene povezave $e = (a, b)$, ki jo dodamo vsakemu »novemu« grafu, imenujemo razdeljena grafa (split graphs) in sta ponovno dvojno povezana. Če je operacijo »razbitje« grafov uporabimo rekurzivno na razdeljenih grafih lahko tako sestavimo graf G (ni nujno enolična rešitev).

Vsaka povezava iz E je vsebovana natanko enkrat in vsaka virtualna povezava je vsebovana natanko dvakrat v dveh komponentah razdelitve.

Lema 5.43 *Naj bo $G = (V, E)$ 2-povezan multi graf z $|E| \geq 3$. Potem je celotno število povezav vsebovanih v vseh komponentah razdelitev omejeno z $3|E| - 6$.*

Dokaz. Dokazali bomo z indukcijo po povezavah grafa G: Če je $|E| = 3$, G ni razcepni graf torej lema velja. Sedaj predpostavimo, da lema velja za grafe, ki imajo do $m - 1$ povezav. Če ima graf G m točk lema velja, če ga ni moč razcepiti. Če pa ga lahko razcepimo, graf tako razpade na dva dela, ki imata $k + 1$ in $m - k + 1$ povezav pri čemer je $2 \leq k \leq m - 2$. S to predpostavko je torej število povezav omejeno na $3(k + 1) - 6 + 3(m - k + 1) - 6 = 3m - 6$

□

Lema 5.44 *3-povezane komponente (multi) grafa so enolično določene.*

Sedaj si poglejmo definicijo SPQR dreves. Razdeljen par (split pair) 2-povezanega grafa G je ali par razdelitve (separation pair) ali pa par sosednjih točk. Razdeljena komponenta razdeljenega para $\{u, v\}$ je ali (u, v)-povezava ali pa vsebovan v maksimalnem podgrafu G, kjer $\{u, v\}$ ni razdeljen par. Razdeljen par $\{u, v\}$ iz grafa G imenujemo maksimalni par razdelitve za razdeljen par $\{s, t\}$ grafa G, če za kakšno razdelitev $\{u', v'\}$, če so točke u, v, t in s v isti razdelitveni komponenti.

Definicija 5.45 Naj bo $e = (s, t)$ povezava iz grafa G. SPQR-drevo T grafa G za neko referenčno povezavo je drevo z korenom sestavljeno iz štirih različnih tipov vozlišč (S.P.Q.R.). T je rekurzivno definirano kot:

(Q) Trivialni primer (Simple Case): Če je G iz natanko dveh vzporednih povezav $s - t$, potem je T edino Q-vozlišče s skeletom G.

(P) Vzporedni primer (Parallel Case): Če ima razdeljen par $\{s, t\}$ več kot dve razdeljeni komponenti $G_{1,\ldots,k}$, je koren T P-vozlišče s skeleton iz k vzporednih $s - t$-povezav $e_{1,\ldots,k}$ in $e_1 = e$.

(S) Primer serije (Series Case) Če ima razdeljen par $\{s, t\}$ natanko dve razdeljeni komponenti in je ena e, drugo označimo z G'. Če ima G' prerezne točke $c_{1,\ldots,k}(k \geq 2)$, ki razdelijo particijo G na bloke $G_{1,\ldots,k}$ (urejene od s do t), je koren od T S-vozlišče, čigar skelet je cikel sestavljen iz povezav $e_{0,\ldots,k}$, pri čemer je $e_0 = e$ in $e_i = (c_{i-1}, c_i)$ za $i = 1, \ldots, k$, $c_0 = s$ in $c_k = t$.

(R) Tog primer (Rigid Case)V vseh ostalih primerih naj bo $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$ maksimalni razdeljeni pari G za $\{s, t\}$. Nadalje naj bo G_i za $i = 1, \ldots, k$ označuje unijo vseh razdeljenih komponent $\{s_i, t_i\}$ razen za tistega, ki vsebuje e. Koren T je R-vozlišče, kjer je skelet sestavljen iz G z zamenjavo vsakega podgrafa G_i z povezavo $e_i = (s_i, t_i)$.

Za ne trivialne primere so otroci $\mu_{1,\ldots,k}$ vozlišča koreni SPQR dreves $G_i \cup e_i$ za e_i. Točke, ki so povezane z vsako povezavo e_i so poli vozlišča μ_i. Virtualna povezava vozlišča μ_i je povezava e_i skeleta vozlišča. SPQR drevo T je zaključeno, ko dodamo Q-vozlišče kot starša vozlišča in tako postane nov koren.

Vsaka povezava iz G se povezuje z nekim Q-vozliščem drevesa T in vsaka povezava e_i iz skeleta vozlišča se povezuje z njenim otrokom μ_i. Za koren T lahko vzamemo neko Q-vozlišče in tako dobimo SPQR drevo za pripadajočo povezavo.

Izrek 5.46 *Naj bo G 2-povezan multi graf z SPQR drevesom T.*

1. *skeletni grafi drevesa T so 3-povezane komponente grafa G. P-vozlišča ustrezajo vezem, S-vozlišča poligonom in R-vozlišča 3-povezanim enostavnim grafom*

2. *med dvema vozliščema $\mu, \nu \in T$ obstaja povezava natanko tedaj, ko si pripadajoči 3-povezani komponenti delita skupno virtualno povezavo*

3. *velikost T, skupaj z vsemi skeletnimi grafi je linearna v z velikostjo G.*

Definicija 5.47 Palma P je usmerjen multi graf, ki je sestavljen iz take množice drevesnih lokov $v \to w$ in take množice listov $v \hookrightarrow w$, da drevesni loki tvorijo vpeto drevo P (tako da koren nima vhodnih povezav, vse ostale točke pa imajo natanko enega starša)in če je $v \hookrightarrow w$ je list, potem obstaja usmerjena pot od w do v.

Pa recimo sedaj, da je P palma za pripadajoč enostaven 2-povezan graf $G' = (V, E')$ (z oznakami na točkah $1, \ldots, |V|$). Računanje razdelitvenih parov temelji na naslednjih spremenljivkah:

$$\text{lowpt1}(v) = \min(\{v\} \cup (\{w | v \to^* \hookrightarrow w\})$$

$$\text{lowpt2}(v) = \min(\{v\} \cup (\{w | v \to^* \hookrightarrow w\}\{\text{lowpt1}(v)\})$$

To sta točki z najmanjšo oznako, ki sta dosegljivi iz v s prečkanjem poljubnega števila (vključno z nič) drevesnih lokov in enega lista palme P (ali pa je to v, v primeru, da taka točka ne obstaja).

Poglavje 6

Gručavost

Uroš Kovač, Katarina Zadražnik, Maja Alif

Vaš krog prijateljev, roj čebel, filmski žanri, koncerni ... Ogromno je pojavov v naravi in vsaj toliko v človeškem organiziranju, ko so člani velike skupine razdeljeni v manjše gruče. Pripadnost ni naključna, gruče se formirajo na podlagi neke podobnosti, pa naj gre za sorodstvene vezi, enake potrebe, cilje, način delovanja ... Zaradi tako široke pojavnosti ne preseneča dejstvo, da se je preučevanja tega fenomena lotilo mnogo različnih vej znanosti. Prvi so se s problemom gručavosti resneje začeli ukvarjati v računalništvu pri rudarjenju s podatki (*data mining*). To je bila spodbuda še večim disciplinam, da so začele razvijati teorijo s svojega stališča. Delo vseh pa izhaja iz abstraktizirane splošne opredelitve termina, da je *gručavost pojav, ko se množico osebkov na podlagi nekih skupnih vezi naravno razdeli v več skupinic - gruč.*

V seminarski nalogi je pojem podrobneje predstavljen, a zato se je bilo nujno omejiti na en pristop. Izhodiščno idejo smo vzeli enako kot mnogi: opazovati, kako močno so povezani osebki znotraj posameznih gruč v primerjavi s povezavami med gručami. To je preprosta in zelo temeljna ideja. Vendar pa se naš pristop v nečem bistveno razlikuje od klasične teorije. Ta jemlje objekte kot vložene v metrični prostor, razdalja je sorazmerna s podobnostjo med njimi. Tu bo drugače, saj so vhodni podatki omrežja, ki v splošnem niso polna, kar pomeni, da ni povezave med vsakim parom. Privzeti način se zdi ustreznejši tudi zaradi opažanja, da se pri standardni analizi najpogosteje pojavljajo redka omrežja.

Predpostavka o razmerju gostot znotraj in med gručami je priljubljena predvsem zaradi podobnosti s človeškim dojemanjem pojava. Knjige delimo glede na vsebino na znanstvene, leposlovne, priročnike ... Nato pa drobimo še naprej, znanstvena dela denimo po različnih vejah znanosti itd. Podobnost oziroma povezava med elementi ene skupine je velika, medtem ko je povezava med elementoma različnih skupin (gruč) tipično šibka.

Za gručavost, osnovano na predpostavki razmerja gostot ali bolj sofisticiranih reformulacijah le-te, so idealna struktura nepovezane klike. A zgodi se, da je hoteni rezultat popolnoma drugačen. S tem se ukvarjata teorija vlog in modelov blokov (*blockmodels*).

V začetnih dveh razdelkih bomo poleg definicij osnovnih pojmov vpeljali nekaj možnih načinov merjenja kakovosti gručavosti. V tretjem razdelku nato nadaljujemo s pregledom osnovnih metod za reševanje problema gručavosti. Spoznali bomo požrešno in premično

metodo ter procesa združevanja in delitve. Preostanek razdelka pa bomo namenili še ostalim pomembnim optimizacijskim pristopom h grupiranju. V četrtem razdelku se bomo posvetili tako standardnim kot tudi nestandardnim vhodom pri procesih združevanja in delitve. Vsebinski del naloge zaokrožujeta razdelka o alternativnih pristopih k teoriji gručavosti ter poskusih njene aksiomatizacije.

6.1 Osnovni pojmi

Naj bo $G(V, E)$ usmerjen graf, kjer je V množica vozlišč in E množica povezav grafa G.

Definicija 6.1 Gručavost $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ je particija množice vozlišč V na neprazne podmnožice C_i, ki jim pravimo **gruče**. Izraz k gručavost pomeni gručavost s k gručami.

Z $E(C_i, C_j)$ označimo množico tistih povezav, ki imajo začetno vozlišče v gruči C_i in končno vozlišče v gruči C_j. Za $E(C_i, C_i)$ uvedemo okrajšavo $E(C_i)$. Nadalje definiramo množico $E(\mathcal{C}) := \cup_{i=1}^{k} E(C_i)$, tj. množico povezav z začetkom in koncem znotraj iste gruče. Njen komplement $\overline{E(\mathcal{C})} := E \setminus E(\mathcal{C})$ je torej množica tistih povezav, ki imajo začetek v eni, konec pa v drugi gruči. Z $m(\mathcal{C})$ označimo moč množice $E(\mathcal{C})$ in z $\overline{m}(\mathcal{C})$ moč množice $\overline{E(\mathcal{C})}$.

Opomba 6.2 Gručo C_i lahko razumemo kar kot inducirani podgraf grafa G z vozlišči gruče C_i, tj. $G[C_i] = (C_i, E(C_i))$.

Trivialni gručavosti dobimo v dveh primerih:

- $k = 1$: grafa G sploh ne razbijemo (1-gručavost);
- $k = |V|$: razbitje grafa G na enojčke (k-gručavost).

2-gručavost imenujemo tudi prerez grafa G oziroma razbitje množice vozlišč na dve disjunktni podmnožici. Množico vseh možnih gručavosti označimo z $\mathcal{A}(G)$.

Definicija 6.3 Naj bosta $\mathcal{C}_1 = \{C_1, \ldots, C_k\}$ in $\mathcal{C}_2 = \{C'_1, \ldots, C'_\ell\}$ gručavosti nad istim grafom. Relacija **vsebovanosti** $\leq$ je podana na naslednji način:

$$\mathcal{C}_1 \leq \mathcal{C}_2 : \Longleftrightarrow \ \forall j \in \{1, \ldots, k\} \,.\, \exists j \in \{1, \ldots, \ell\} \,.\, \mathcal{C}_i \subseteq \mathcal{C}_j.$$

Enostavno se preveri, da je relacija vsebovanosti delno urejena relacija, tj. zadošča pogojem refleksivnosti, antisimetričnosti in tranzitivnosti. Od tod sledi naslednja trditev.

Trditev 6.4 *Množica $\mathcal{A}(G)$ je delno urejena množica z relacijo vsebovanosti.*

Verigi gručavosti, tj. podmnožici $\mathcal{A}(G)$, v kateri sta poljubna dva elementa primerljiva, pravimo **hierarhija**. Hierarhija je **totalna**, če sta v njej vsebovani obe trivialni gručavosti. Za hierarhijo, ki vsebuje natanko eno k-gručavost za vsak $k \in \{1, \ldots, n\}$, pravimo, da je **popolna**.

Prerezna funkcija $S : \mathcal{P}(V) \to \mathcal{P}(V)$ je definirana na naslednji način:

$$\forall V' \subseteq V \,.\, S(V') \subseteq V'.$$

Za dano podmnožico $V' \subset V(G)$ nam prerezna funkcija S da njen **prerez**: $(S(V'), V' \setminus S(V'))$. Če je $|V'| > 1$ in jo prerezna funkcija S slika v pravo neprazno podmnožico V', tj. $S(V') \notin \{V', \{\}\}$, dobimo netrivialen prerez. Prereznim funkcijam, ki zadoščajo temu dodatnemu pogoju, pravimo, da so prave prerezne funkcije. Te funkcije so pomembne pri nekaterih tehnikah gručavosti, ki temeljijo na rekurzivnih prerezih in jih bomo predstavili v drugem poglavju.

Za model grafa bomo vzeli enostaven usmerjen graf, brez zank in z utežmi na povezavah.

6.2 Merjenje kakovosti gručavosti

Tehnike pri gručavosti iščejo take gruče, ki so goste, tj. grafi znotraj posameznih gruč so blizu polnim grafom, med posameznimi gručami pa je le nekaj povezav. V nadaljevanju bomo zato poskušali formalno opisati, katere gručavosti so slabe in katere dobre. V ta namen bomo predstavili različne **strukturne indekse**, s katerimi na matematični način izmerimo kakovost gručavosti.

Naj bo $G(V, E, \omega)$ enostaven, usmerjen in utežen graf, kjer preslikava $\omega : E \to \mathbb{R}_0^+$ vsaki povezavi v grafu G priredi njeno utež. Naj bo gručavost $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$. Utež na povezavi uv nam pove, koliko sta si vozlišči u in v podobni, tj. večja utež na povezavi pomeni, da ti dve vozlišči bolj spadata v isto gručo. Dodatno bomo razlikovali med vozliščema, med katerima ni povezave, in vozliščema, ki imata na povezavi med njima utež enako 0. Definirajmo še vsoto uteži podmnožice povezav $E' \subseteq E(G)$:

$$\omega(E') := \sum_{e \in E'} \omega(e).$$

6.2.1 Splošna oblika indeksov

Preden preidemo h konkretnim primerom strukturnih indeksov, bomo predstavili njihovo splošno obliko.

Strukturni indeksi so sestavljeni iz dveh neodvisnih funkcij $f, g : \mathcal{A}(G) \to \mathbb{R}_0^+$, kjer f meri gostoto znotraj posameznih gruč, g pa razpršenost med gručami. V strukturnem indeksu gručavosti $\mathcal{C}$ vključimo ti dve funkciji na naslednji način:

$$\text{indeks}(\mathcal{C}) := \frac{f(\mathcal{C}) + g(\mathcal{C})}{\max\{f(\mathcal{C}') + g(\mathcal{C}') \mid \mathcal{C}' \subseteq \mathcal{A}(G)\}}. \tag{6.1}$$

Indeks je dobro definirana funkcija, saj predpostavimo, da obstaja gručavost $\mathcal{C}'$, za katero velja: $f(\mathcal{C}') + g(\mathcal{C}') \neq 0$. Za nekatere gručavosti seveda velja, da je lahko natanko ena izmed f ali g konstantno enaka 0. V primeru, ko je $g \equiv 0$, pravimo, da ta meri le gostoto v gručah. Če pa je $f \equiv 0$, merimo le razpršenost med posameznimi gručami.

Indeksi služijo dvema namenoma:

- ocenijo particijo glede na že znane modele gručavosti;
- primerjajo gručavosti po kakovosti.

Kakovost gručavosti je povezana s številom nekaterih znanih struktur grafa, npr. s številom povezav, trikotnikov in klik znotraj posameznih gruč. Če je znotraj posameznih gruč

veliko število takih struktur, potem je gručavost gosta. Na drugi strani pa odsotnost teh struktur med različnimi gručami pomeni precejšnjo razpršenost gručavosti. Kakovost dane gručavosti lahko torej obravnavamo s količinskega vidika. V vsaki gruči je število povezav, trikotnikov in klik omejeno z velikostjo gruče. V idealnemu primeru so zgornje meje za število posameznih struktur dosežene. Indeksi niso odvisni od prvotnega grafa, temveč od trenutne gručavosti, zato jih lahko uporabljamo za primerjavo kakovosti gručavosti različnih grafov.

Zgled 6.5 Vzemimo dva grafa G in G' s trenutno gručavostjo:

- graf G z 10 vozlišči in 15 povezavami, ki ima 12 povezav znotraj gruč in 3 povezave med gručami;
- graf G' z 18 vozlišči ter 30 povezavami, ki ima 27 povezav znotraj gruč in 3 povezave med gručami.

V obeh primerih je torej število povezav med gručami enako 3. Izračun razmerja med številom povezav znotraj gruč in številom vseh povezav pa pove naslednje: $12/15 = 4/5 < 27/30 = 9/10$, kar pomeni, da je druga gručavost na grafu G' boljša.

Če pa za računanje indeksa uporabljamo katere izmed ostalih karakteristik grafa, npr. število povezav, največja utež na povezavi, itd., lahko kakovosti gručavosti primerjamo le, kadar je njihov prvotni graf enak. Zato se takim indeksom, ki uporabljajo te karakteristike, rajši izogibamo. Želimo namreč, da je naš indeks neodvisen od začetnega grafa.

6.2.2 Pokritost

Pokritost $\gamma(\mathcal{C})$ je funkcija, ki meri skupno težo povezav znotraj gruč glede na težo vseh povezav. Velja $f(\mathcal{C}) = \omega(E(\mathcal{C}))$ in $g \equiv 0$. Največja vrednost je dosežena pri $\mathcal{C} = V(G)$. Pokritost potem definiramo kot

$$\gamma(\mathcal{C}) := \frac{\omega(E(\mathcal{C}))}{\omega(E)} = \frac{\sum_{e \in E(\mathcal{C})} \omega(e)}{\sum_{e \in E} \omega(e)}.$$

Največja vrednost 1 je dosežena pri trivialni gručavosti, ko množice vozlišč sploh ne razbijemo, vendar se bomo, kot smo že rekli, trivialnim gručavostim raje izognili.

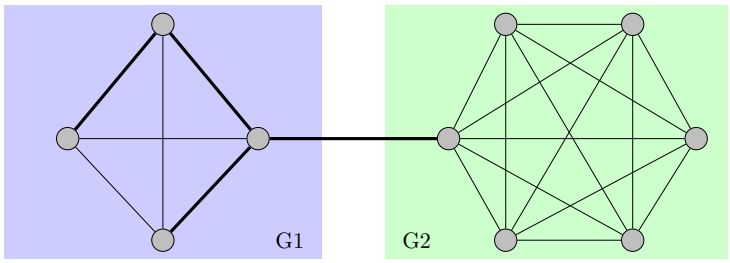
Opomba 6.6 Pokritost nam pove, koliko je skupno število povezav znotraj vseh gruč. Lahko se torej zgodi, da je v eni gruči kljub visokemu indeksu število povezav majhno. In obratno: čeprav je indeks majhen, lahko imamo gručo, ki je zelo gosta. Prikaz tega je na sliki 6.1.

Trditev 6.7 *Gručavost $\mathcal{C}$ ima pokritost $\gamma = 1$ natanko tedaj, ko je množica povezav med gručami prazna ali pa imajo vse povezave v tej množici uteži enake* 0.

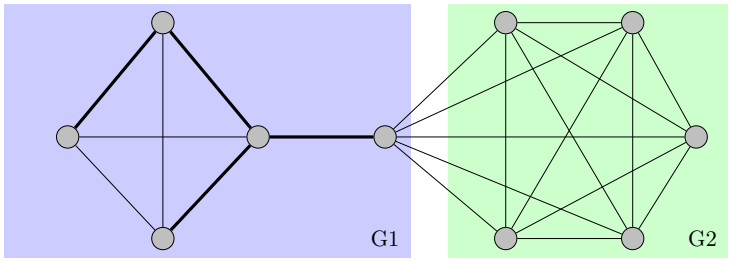
Dokaz. Ker sta $E(\mathcal{C})$ in $\overline{E(\mathcal{C})}$ disjunktni množici povezav, velja enakost

$$\omega(E) = \omega(E(\mathcal{C})) + \omega(\overline{E(\mathcal{C})}).$$

Po definiciji za γ sklepamo, da je $\gamma = 1$ natanko tedaj, ko je $\omega(\overline{E(\mathcal{C})}) = 0$. To pa drži natanko tedaj, ko je množica povezav med gručami prazna oziroma imajo vse povezave iz te množice uteži enake 0.



(a) Intuitivna gručavost.

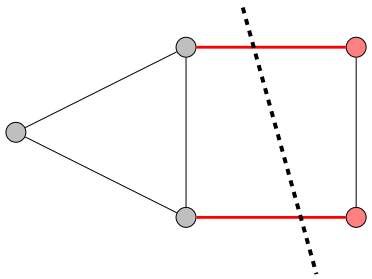


(b) Gručavost, ki ima največjo pokritost.

Slika 6.1: Odebeljene povezave imajo utež enako 100. Ostale povezave imajo utež 1. V tem primeru ima intuitivna gručavost pokritost $\gamma = 159/209 \approx 0.76$. Optimalna vrednost, ki jo doseže pokritost, pa je enaka $413/418 \approx 0.99$.

□

Definicija 6.8 V teoriji grafov je **najmanjši prerez grafa** tisti prerez, pri katerem imamo najmanjše število prereznih povezav, če je graf neuteženi. V primeru uteženega grafa pa je to prerez, pri katerem je vsota uteži na prereznih povezavah najmanjša. Na sliki 6.2 je primer prereza.



Slika 6.2: Najmanjši prerez neuteženega grafa na sliki ima dve prerezni povezavi, ni pa enoličen.

Trditev 6.9 *Naj bo $G(V, E)$ povezan graf, v katerem ima vsak možni prerez pozitivno vsoto uteži na prereznih povezavah. Netrivialne gručavosti z optimalno pokritostjo so tiste z dvema gručama, ki ju inducira najmanjši prerez.*

Dokaz. Ni težko videti, da lahko vsako k-gručavost $\mathcal{C}$ ($k > 1$), preoblikujemo v gručavost $\mathcal{C}'$ z manj kot k gručami in večjo pokritostjo: $\gamma(\mathcal{C}) \leq \gamma(\mathcal{C}')$. To dosežemo namreč tako,

da združimo katerikoli dve gruči v eno gručo. Imenovalec v formuli za pokritost ostane enak, števec pa se poveča, saj lahko dobimo nove povezave v notranjosti združene gruče, tj. tiste, ki so prej povezovale naši dve gruči, ohranile pa so se seveda tudi vse prejšne povezave znotraj naših dveh gruč. Torej ima netrivialna gručavost z optimalno pokritostjo dve gruči in je zato prerez. Če želimo doseči najboljšo pokritost, moramo minimizirati vsoto uteži na povezavah med gručami, tj. točno to, kar naredi najmanjši prerez.

□

Naštete lastnosti nam povedo, da pokritost ni nujno najboljše merilo za kakovost gručavosti. Zato je redko uporabljena kot edino merilo kakovosti gručavosti. Najmanjših prerezov ni vedno lahko najti, hkrati pa optimalna pokritost vzame za merilo kakovosti le število povezav.

V nadaljevanju bomo zato predstavili nove indekse za merjenje kakovosti gručavosti.

6.2.3 Prevodnost

Pokritost je za merilo vzela le vsoto uteži na povezavah med gručami. Za merilo pa lahko vzamemo npr. tudi strukturno lastnost povezanost. Gruče naj bi bile dobro povezane, tj. potrebno je odstraniti veliko povezav, da gruča postane nepovezana. Med dvema različnima gručama pa naj bi bilo čim manj povezav, v idealnem primeru nobene.

Prerezi so uporabni pri merjenju kakovosti povezanosti. Vendar pa ima najmanjši prerez določene pomankljivosti, zato vpeljemo alternativno merilo, povezano s prerezi, ki ga imenujemo **prevodnost**. Ta primerja vsoto vseh uteži na prereznih povezavah z vsoto uteži povezav obeh podgrafov, na katera razpade graf s prerezom.

Definicija 6.10 Naj bo $\mathcal{C}' = (C'_1, C'_2)$ prerez ($C'_2 = V \backslash C'_1$). Potem sta **teža prevodnosti** $a(C'_i)$ in **prevodnost** $\varphi(\mathcal{C}')$ definirani na sledeči način:

$$a(C'_i) := \sum_{(u,v)\in E(C'_i,V)} \omega(u,v)$$

$$\varphi(\mathcal{C}') := \begin{cases} 1, & \text{če } C'_1 \in \{\emptyset, V\}; \\ 0, & \text{če } C'_1 \notin \{\emptyset, V\}, \omega(\overline{E(\mathcal{C}')}) = 0; \\ \frac{\omega(\overline{E(\mathcal{C}')})}{\min(a(C'_1), a(C'_2))}, & \text{sicer.} \end{cases}$$

Prevodnost grafa G pa je definirana kot:

$$\varphi(G) = \min_{\mathcal{C}' \text{ prerez}} \varphi(\mathcal{C}').$$

Opomba 6.11 Drugi primer v definiciji prevodnosti obravnavamo, da se izognemo morebitnemu deljenju z 0.

Lema 6.12 *Naj bo $G = (V, E, \omega)$ neusmerjen graf s pozitivnimi utežmi na povezavah. Tedaj ima G največjo prevodnost, tj. $\varphi(G) = 1$, natanko tedaj, ko je povezan in ima največ 3 vozlišča ali pa je zvezda.*

Dokaz. Najprej si poglejmo dve opazki:

1. Vsi nepovezani grafi imajo prevodnost enako 0, saj zanje obstaja netrivialen prerez, ki ima na prereznih povezavah (teh ni) vsoto uteži 0. Torej ustrezajo drugi točki pri definiciji prevodnosti in res drži $\varphi = 0$.

2. Za netrivialen prerez $\mathcal{C}' = (C_1', V \setminus C_1')$ neusmerjenega grafa lahko težo prevodnosti $a(C_1')$ zapišemo drugače:

$$a(C_1') = \sum_{e \in E(C_1, V)} \omega(e) = \omega(E(C_1') + \omega(\overline{E(\mathcal{C}')}.$$

Tedaj se tretja točka pri definiciji prevodnosti prepiše v

$$\frac{\omega(\overline{E(C')}}{\min\left(a(C_1'), a(C_2')\right)} = \frac{\omega(\overline{E(C')}}{\omega(\overline{E(C')}) + \min\left(\omega(E(C_1')), \omega(E(V \setminus C_1'))\right)}.$$

'$\Longleftarrow$': Če ima graf G le eno vozlišče, je $C_1' \in \{\emptyset, V\}$; velja $\varphi(G) = 1$.
Če ima graf dve ali tri vozlišča oziroma je zvezda, potem vsak netrivialen prerez $\mathcal{C}' = (C_1', V \setminus C_1')$ izolira podmnožico vozlišč, kjer ni nobene povezave med njimi. Pri grafu z največ tremi vozlišči je to manjša od množic C_1' in $V \setminus C_1'$, pri zvezdi pa tista, ki ne vsebuje centralnega vozlišča. Za tako nepovezano množico C_1' velja $\omega(E(C_1')) = 0$ in potem iz druge opazke sledi $\varphi(\mathcal{C}') = 1$. Ker to velja za vsak netrivialen prerez, je $\varphi(G) = 1$.

'$\Longrightarrow$': Če ima graf prevodnost 1, potem je povezan (opazka 1) in za vsak netrivialen prerez $\mathcal{C}'$ ima vsaj ena izmed množic povezav $E(C_1')$ in $E(V \setminus C_1')$ vsoto uteži na povezavah enako 0. Ker so uteži na povezavah pozitivne, mora biti vsaj ena izmed teh dveh množic prazna.

Grafi na največ treh vozliščih očitno izpolnjujejo ta pogoj, zato predpostavimo, da ima graf G vsaj 4 vozlišča. Graf G mora tedaj imeti diameter največ 2. V nasprotnem primeru bi namreč obstajala pot dolžine tri med štirimi različnimi vozlišči $v_1, \ldots, v_4$, kjer $e_i := v_i v_{i+1} \in E$ za $1 \leq i \leq 3$. Od tod bi sledilo, da netrivialen prerez $\mathcal{C}' = (\{v_1, v_2\}, V \setminus \{v_1, v_2\})$ nima prevodnosti 1, kar po opazki 2 sledi iz naslednjih dveh dejstev:

- velja neenakost $\omega(\overline{E(\mathcal{C}')} \geq \omega(e_2) > 0$, ki pove, da gledamo tretji pogoj v definiciji prevodnosti;
- $\min\left(\omega(E(C_1')), \omega(E(V \setminus C_1'))\right) \neq 0$, saj $e_1 \in \{v_1, v_2\}$ in $e_3 \in V \setminus \{v_1, v_2\}$.

Graf G zaradi istega argumenta ne more imeti cikla dolžine štiri. Velja še več: graf G ne more imeti cikla dolžine tri. Recimo, da graf G ima cikel dolžne tri z vozlišči v_1, v_2, v_3. Ker ima graf vsaj štiri vozlišča, obstaja vozlišče v_4, ki ni v ciklu, je pa sosednje vozlišče enega izmed vozlišč v_1, v_2 ali v_3 (graf je povezan), BŠS je to lahko vozlišče v_1. Tedaj imamo netrivialen prerez $(\{v_1, v_4\}, V \setminus \{v_1, v_4\})$, ki nima prevodnosti 1. Graf G torej ne vsebuje ciklov in je zato drevo. Edina drevesa z vsaj štirimi vozlišči in diametrom največ 2 pa so zvezde.

□

Izračun prevodnosti je $\mathcal{NP}$-težek problem [114]. Na srečo obstajajo aproksimacijski algoritmi zahtevnosti $\mathcal{O}(\log n)$ (bralec si o njih lahko prebere v [133]) in $\mathcal{O}(\sqrt{\log n})$ (bralec si lahko o njih prebere v [113]), za nekatere posebne razrede grafov pa celo algoritmi s konstantno zahtevnostjo. Za neuteženе polne grafe je prevodnost uporabno merilo za merjenje njene kakovosti. Naslednja trditev nam pove njeno točno vrednost. Njena vrednost se sicer razlikuje glede na parnost števila vozlišč, vendar se v obeh primerih z rastjo števila vozlišč asimptotično bliža vrednosti $\frac{1}{2}$.

Trditev 6.13 *Za naravno število n velja sledeča enakost:*

$$\varphi(K_n) = \begin{cases} \frac{1}{2} \cdot \frac{n}{n-1} & , \text{če } n \text{ sod;} \\ \frac{1}{2} + \frac{1}{n-1} & , \text{če } n \text{ lih.} \end{cases}$$

Dokaz. Tretja točka v definiciji prevodnosti nas za poln poln graf pripelje do

$$\varphi(K_n) = \min_{\substack{C \subset V \\ 1 \leq |C| = k < n}} \frac{k(n-k)}{\min(k(k-1), (n-k)(n-k-1))}.$$

Enakost je simetrična, zato je dovolj, če k teče od 1 do $\lceil \frac{n}{2} \rceil$. Opazimo tudi, da ulomek monotono pada, ko k narašča. Ločimo dva primera, v obeh vzamemo $k_2 > k_1$.

- $\frac{k(n-k)}{k(k-1)} = \frac{n-k}{k-1}$ pada, ko k raste, saj iz neenakosti $\frac{n-k_1}{k_1-1} < \frac{n-k_2}{k_2-1}$ po malo računanja pridemo do neenakosti $n > 1$, kar očitno drži za polne grafe na več kot eni točki.
- $\frac{k(n-k)}{(n-k)(n-k-1)} = \frac{k}{n-k-1}$ prav tako pada, ko k raste, saj lahko podobno kot prej neenakost $\frac{k_1}{n-k_1-1} < \frac{k_2}{n-k_2-1}$ preoblikujemo do neenakosti $n > 1$.

Minimum je torej dosežen pri $k = \lfloor \frac{n}{2} \rfloor$. Ločimo primera za sode in lihe n in pridemo do končnega rezultata.

□

S pomočjo prevodnosti lahko izpeljemo dva nova indeksa:

- prevodnost znotraj gruč (meri gostoto v gručah);
- prevodnost med gručami (meri povezanost različnih gruč).

Prevodnost znotraj gruče α je definirana kot minimalna prevodnost v podgrafu, ki ga inducira dana gruča, ta podgraf označimo z $G[C_i]$:

$$f(\mathcal{C}) = \min_{1 \leq i \leq k} \varphi(G[C_i]) \text{ in } g \equiv 0.$$

Prevodnost (pod)grafa je majhna, če ga lahko naravno prerežemo, drugače je velika. Tako mora biti v gručevju z majhno prevodnostjo znotraj gruč, vsaj ena gruča, za katero obstaja naravni prerez.

Prevodnost med gručami δ upošteva prereze, ki jih inducirajo gruče:

$$f \equiv 0 \text{ in } g = \begin{cases} 1 & , \text{ če } \mathcal{C} = \{V\}; \\ 1 - \max_{1 \leq i \leq k} \varphi((C_i, V \setminus C_i)) & , \text{ sicer.} \end{cases}$$

Gručavost z majhno prevodnostjo med gručami vsebuje vsaj eno gručo, ki ima relativno veliko povezav zunaj, tj. gručavost je preveč fina (razdeljena na preveč gruč).

Formuli za ta dva indeksa dobimo z upoštevanjem maksimuma $f + g$, ki je za oba indeksa očitno enak 1:

$$\alpha(\mathcal{C}) := \min_{1 \leq i \leq k} \varphi(G[C_i])$$

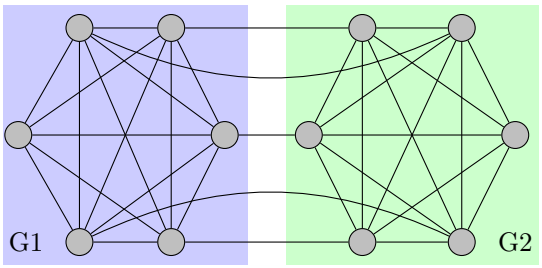
$$\delta(\mathcal{C}) := \begin{cases} 1 & , \text{ če } \mathcal{C} = \{V\}; \\ 1 - \max_{1 \leq i \leq k} \varphi((C_i, V \setminus C_i)) & , \text{ sicer.} \end{cases}$$

Naslednja trditev karakterizira najboljše gručavosti za indeksa.

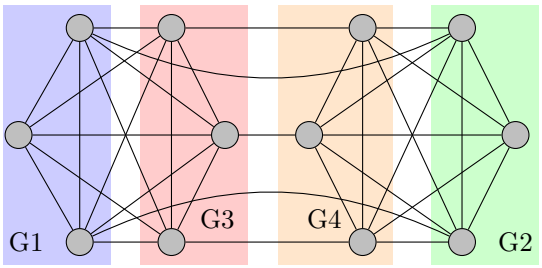
Trditev 6.14 *Karakterizacija najboljših gručavosti za indeksa:*

- *Prevodnost znotraj gruč imajo enako 1 natanko tiste gručavosti, ki so sestavljene iz gruč, ki inducirajo povezane grafe, ki so zvezde, ali pa imajo največ tri vozlišča.*
- *Prevodnost med gručami imajo enako 1 natanko tiste gručavosti, ki imajo na povezavah med gručami uteži enake* 0*, vključno s trivialno gručavostjo* $\mathcal{C} = \{V\}$.

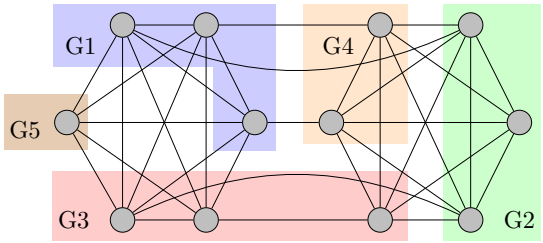
Zgornja trditev sledi iz definicije prevodnosti 6.10 in leme 6.12. Oba indeksa imata določene slabosti. Na spodnjih slikah sta prikazana dva primera, ko se izkažejo slabosti teh dveh indeksov.



(a) Intuitivna gručavost.



(b) Netrivialna gručavost z najboljšo prevodnostjo znotraj gruč.



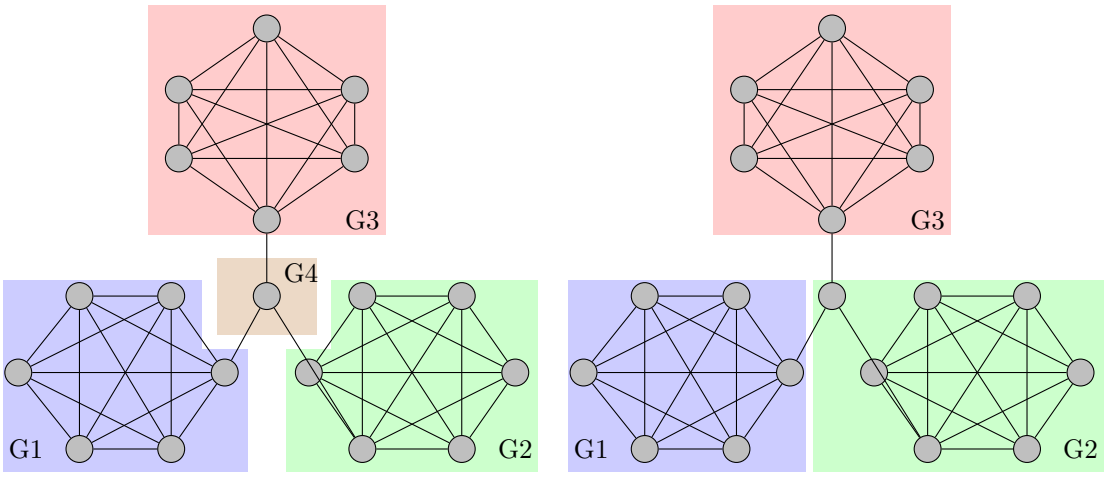
(c) Netrivialna gručavost z najboljšo prevodnostjo znotraj gruč.

Slika 6.3: Intuitivna gručavost ima prevodnost znotraj gruč $\alpha = \frac{3}{4}$. Podgrafa, inducirana z gručami, imata najmanjšo prevodnost pri prerezu, kjer prva množica vsebuje tri vozlišča, ki so sosednja z vsemi ostalimi, druga množica pa preostale tri.

6.2.4 Zmogljivost

Zmogljivost je indeks, ki šteje določene pare vozlišč. Za dano gručavost definiramo t.i. pravilne pare vozlišč. **Pravilni par vozlišč** tvorita bodisi dve različni vozlišči, ki sta krajišči ene povezave znotraj ene gruče, bodisi vozlišči, ki sta v različnih gručah in nista povezani. Funkcija gostote f v indeksu zmogljivosti nam pove število povezav znotraj gruč. Funkcija razpršenosti g pa nam pove število parov vozlišč, ki sta v različnih gručah in nista povezani. Če zapišemo f in g s formulami:

$$f(\mathcal{C}) := \sum_{i=1}^{k} |E(C_i)|$$
$$g(\mathcal{C}) := |\{\{u, v\} \mid uv \notin E, u \in C_i, v \in C_j, i \neq j\}|$$



(a) Intuitivna gručavost.

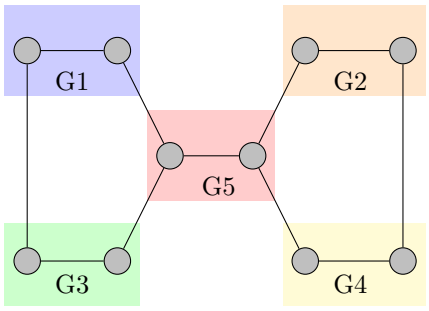
(b) Netrivialna gručavost z najboljšo prevodnostjo med gručami.

Slika 6.4: Podobni gručavosti imata zelo različni prevodnosti med gručami. Intuitivna gručavost ima $\delta = 0$ (dosežena je pri prerezu, ki ima za eno množico vozlišč samo gručo $G4$), druga pa $\delta = \frac{8}{9}$ (dosežena je pri prerezu, ki ima za eno množico vozlišč gručo $G2$).

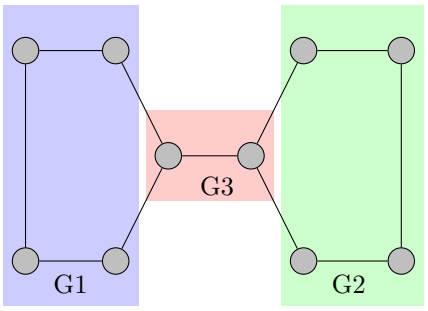
Zgornja meja za $f + g$ je gotovo $\frac{n(n-1)}{2}$, saj je toliko različnih parov vozlišč. Določitev pravega maksimuma pa je $\mathcal{NP}$-zahteven problem. V splošni formuli za indekse (6.1) zato za maksimum vzamemo kar zgornjo mejo $\frac{n(n-1)}{2}$. Funkcijo g lahko izračunamo tako, da od vseh možnih parov vozlišč odštejemo vse možne pare vozlišč v notranjosti gruč in število povezav med različnimi gručami ($\overline{m}(\mathcal{C})$). Formulo za zmogljivost potem poenostavimo:

$$\begin{aligned} perf(\mathcal{C}) &= \frac{m(\mathcal{C}) + \left(\frac{n(n-1)}{2} - \sum_{i=1}^{k} \frac{|C_i||C_i-1|}{2} - \overline{m(\mathcal{C})}\right)}{\frac{n(n-1)}{2}} \\ &= \frac{n(n-1) - 2m + 4m(\mathcal{C}) - \sum_{i=1}^{k} |C_i||C_i-1|}{n(n-1)} \\ &= 1 - \frac{2m(1 - \frac{2m(\mathcal{C})}{m}) + \sum_{i=1}^{k} |C_i||C_i-1|}{n(n-1)}, \end{aligned}$$

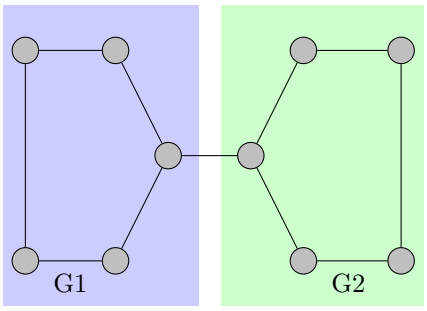
kjer smo upoštevali enakost $m = m(\mathcal{C}) + \overline{m(\mathcal{C})}$. Podobno kot drugi indeksi ima tudi zmogljivost svoje prednosti in slabosti. Njena največja pomanjkljivost pride na dan pri obravnavi zelo razpršenih grafov, ker le-ti nimajo podgrafov poljubnih velikosti in gostot. V takih primerih je razlika med številom možnih povezav pri dani strukturi in številom vseh možnih povezav ne glede na strukturo ($\frac{n(n-1)}{2}$) zelo velika in zmogljivost ne oceni dobro kakovosti različnih gručavosti. Za primer lahko vzamemo ravninske grafe. Ti ne morejo vsebovati polnih grafov s pet ali več vozlišči. Največje število povezav, da je graf ravninski, je linearno v številu vozlišč, medtem ko je za splošne grafe največje število povezav kvadratično. Ugotovimo še, da imajo gručavosti z dobro zmogljivostjo ponavadi veliko manjših gruč. Indeksi nam torej služijo kot merilo kakovosti za gručavosti. Predstavljajo formalen način, kako prepoznati naravna razbitja. Obstajajo še mnogi drugi različni indeksi, nekatere izmed njih so najprej uporabljali v metričnih in vektorskih prostorih, opremljenih z normo, nato pa so jih uvedli v teorijo grafov.



(a) Gručavost z naboljšo zmogljivostjo.



(b) Intuitvna gručavost.



(c) Intuitvna gručavost.

Slika 6.5: Prikaz gručavosti z najboljšo zmogljivostjo. Za primerjavo sta poleg dve intuitivni gručavosti z slabšo zmogljivostjo.

Skupna lastnost indeksov je, da je iskanje gručavosti z največjo vrednostjo indeksa $\mathcal{NP}$-zahteven problem.

Bibliografske opombe

Nadaljnje informacije v zvezi z indeksi lahko najdemo v [125] in [127]. V [117] so predstavljeni primerjalni testi, ki temeljijo na indeksih. Na področju strojnega učenja in nevronskih mrež je potrebno še veliko raziskovalnega dela. Posebaj se je potrebno osredotočiti na indekse in pa pridobivanje strukturnih lastnosti za kakovostno mero particij. Vstopna točka do teh vprašanj se bralcu ponuja v [121] in [133].

6.3 Metode za reševanje problema gručavosti

Predstavljena teorija nam bo zdaj pomagala poiskati algoritme, s katerimi lahko izračunamo optimalno gručavost za dani graf.

Osnovne tehnike, ki jih uporabljamo pri reševanju problema gručavosti, lahko razdelimo v tri razrede:

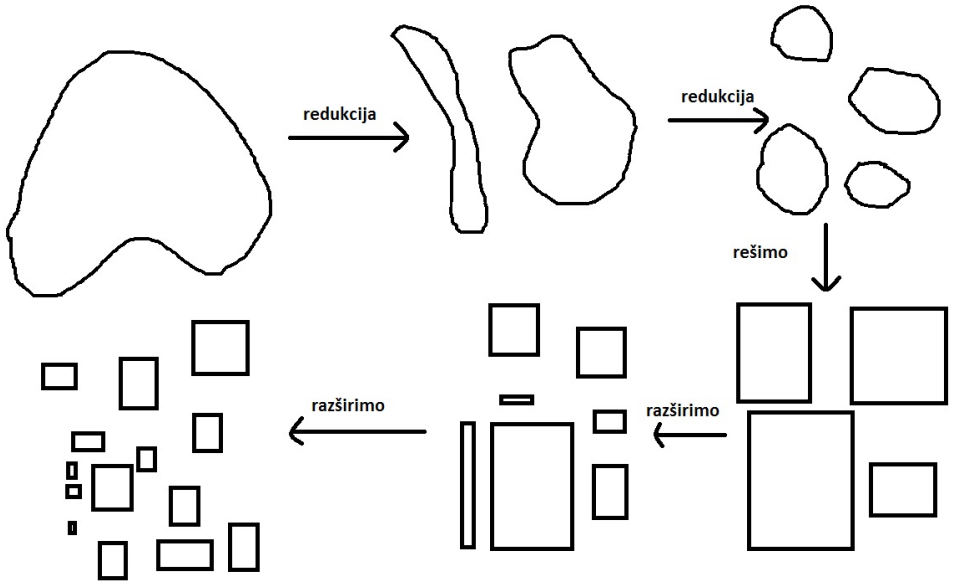
- požrešne metode;
- premične metode;
- splošni optimizacijski oziroma aproksimacijski pristopi, ki dajo približno rešitev.

Skupna ideja zgornjim razredom osnovnih tehnik je ta: *Problem delimo toliko časa, da dobimo obliko, v kateri problem znamo rešiti, nato pa ga razširimo nazaj do prvotnega problema.*

Ta ideja nas spominja na algoritem deli in vladaj, kjer dani problem delimo na vse manjše podprobleme, dokler le-teh ne znamo rešiti, nato pa rekurzivno nazaj sestavimo celotno rešitev. Tehnika, ki jo bomo uporabili mi, je veliko splošnejša, saj se lahko pri delitvi problema vhod za podprobleme popolnoma spremeni. To pomeni, da vhodi za podprobleme niso nujno 'podvhodi' za vhod začetnega problema.
Korak delitve problema običajno sestoji iz dveh delov:

- Prepoznati je potrebno določene podstrukture. To so lahko npr. mostovi, ki ločijo gruče, ali določajo neki del skupine. Taka opredelitev je hipoteza, možni podproblemi podpirajo njeno pravilnost.
- Preoblikujemo trenutni graf G. Običajne transformacije, ki jih izvedemo, so:
  - dodajanje povezav;
  - odstranjevanje povezav;
  - zamenjava korena v podgrafu, tj. predstavitev podgrafa z novim korenom.

Na sliki 6.6 je predstavljena ideja redukcije problema in združevanja podproblemov nazaj v skupno rešitev. Oblike gruč nam dajo določene informacije o trenutni gručavosti. Ovalne



Slika 6.6: Primer reševanja problema z redukcijo in sestavljanjem problema nazaj v celoto.

oblike gruč ponazarjajo, da o gručavosti, ki jo imamo, ne vemo veliko. Po drugi strani pa oglate oblike gruč pomenijo, da o naši gručavosti veliko vemo.

Poglejmo si primer na sliki 6.6. V prvi vrsti smo najprej izvedli redukcijo problema, ga iz prve v drugo vrstico rešili, in nato v drugi vrstici nazaj sestavili optimalno rešitev za vhodni graf G. Levo zgoraj imamo najprej gručavost, ki nam ne da nobene informacije. Med postopkom redukcije postaja graf G vedno bolj nepovezan. To smo ponazorili z

disjunktnimi ovalnimi oblikami. Problem, ki ga dobimo, ko zaključimo s postopkom reduciranja, znamo preprosto rešiti. Glede na dani vhod pretvorimo problem v obliko, ki nam da informacije o gručavosti. Z razširitvijo pridemo nazaj do prvotnega problema.

Glavna razlika med tem postopkom in postopkom deli in vladaj je, da se lahko velikost grafa med postopkom redukcije poveča.

6.3.1 Požrešna metoda

Koncept, ki ga uporablja požrešna metoda, je naslednji: *Začnemo s trivialno dopustno rešitvijo. Na vsakem koraku popravimo rešitev tako, da dosežemo trenutno najbolj optimalno stanje. S tem zmanjšujemo ceno problemu. Če rešitve ne moremo več izboljšati, končamo.*

Shematsko lahko to predstavimo z algoritmom 11. V njem s $c(L)$ označimo ceno trenutnega L. Množica $N_g(L)$ pa je množica vseh možnih rešitev, ki jih lahko dosežemo z izboljšavo začetne rešitve L.

Algorithm 11 Shema požrešne metode.

Naj bo L_0 dopustna rešitev.
$i \leftarrow 0$
while $\{L \mid L \in N_g(L_i), c(L) < c(L_i)\} \neq \emptyset$ **do**
  $L_{i+1} \leftarrow \text{argmin}_{L \in N_g(L_i)} c(L)$
  $i \leftarrow i + 1$
end while
return L_i

Preko hierarhij lahko požrešno metodo uporabimo za opis postopka grupiranja. Požrešna metoda, ki uporablja operaciji *združi* in *razdeli*, nam tako naravno definira hierarhijo. Če se omejimo le na eno izmed operacij, lahko gručavosti med sabo primerjamo, kar pa nam da hierarhijo. Te koncepte bomo v nadaljevanju še razložili, najprej pa bomo spoznali nekaj osnovnih dejstev o hierarhijah.

Hierarhije nam dajo določeno svobodo pri grupiranju: število gruč, ki nastopa v njih, namreč ni fiksno. Po drugi strani pa nam ta svoboda, ki jo dopuščamo, poveča prostorsko zahtevnost algoritmov, kljub temu da lahko hierarhijo predstavimo s posebnim drevesom, ti. **dendrogramom**, s pomočjo katerega lahko prikažemo operacije združevanja.

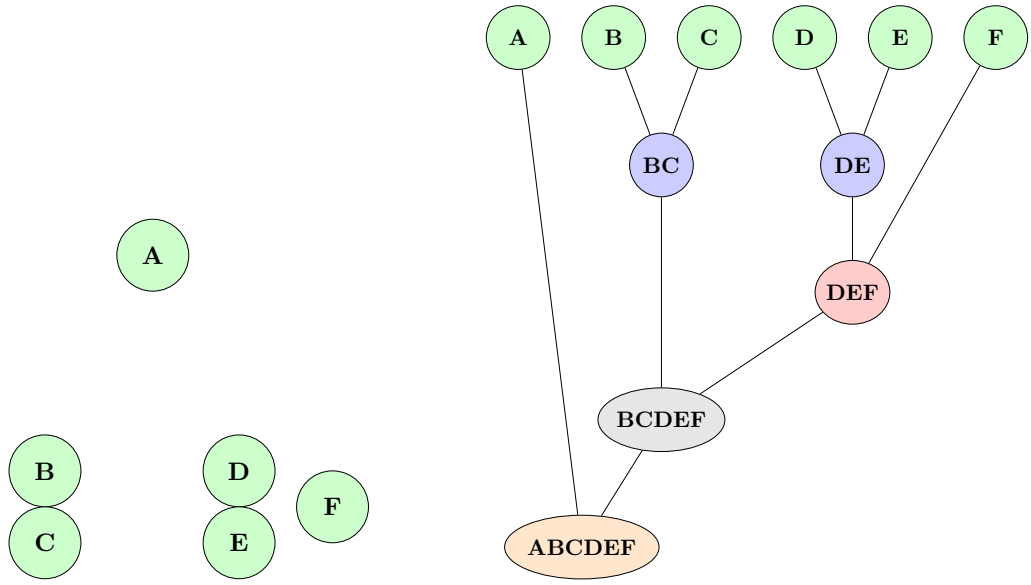
Dendrogram je drevesni diagram, s pomočjo katerega lahko ilustriramo ureditev gruč, ki jih dobimo pri hierarhiji. Pogosta uporaba dendrogramov je npr. v matematični biologiji za ilustracijo gručavosti genov in vzorcev.

V primeru, ko želimo z dendrogramom predstaviti gručavost, prostor opremimo z evklidsko metriko. Na sliki 6.7 levo so vhodni podatki. Na isti sliki desno pa je prikaz dendrograma, ki ga lahko dobimo iz teh začetnih podatkov.

Z algoritmi dendrograme gradimo eksplicitno, zato je prostorska zahtevnost, ki jo dobimo, vsaj kvadratična. V nekaterih posebnih primerih pa lahko s pomočjo določenih podatkovnih struktur to oceno prostorske zahtevnosti tudi še izboljšamo [122].

6.3.2 Proces združevanja

Proces združevanja deluje tako, da iterativno, na vsakem koraku združimo dve gruči, dokler ne dosežemo gručavosti, ki vsebuje eno samo gručo.



(a) Vhodni podatki, ki čakajo na obdelavo.

(b) Dendrogram.

Slika 6.7: Na zgornji ravni v dendrogramu imamo začetne podatke. Preostala vozlišča predstavljajo gruče, katerim pripadajo začetni podatki. S puščicami ponazorimo razdalje med gručami in začetnimi podatki.

Formalna predstavitev procesa združevanja: Naj bodo dani:

- graf $G = (V, E, \omega)$ in začetna gručavost $\mathcal{C}_1$;
- globalna funkcija cene $c_{\text{globalna}} : \mathcal{A}(G) \to \mathbb{R}_0^+$
  ali
- lokalna funkcija cene $c_{\text{lokalna}} : \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$.

Dve gruči v trenutni gručavosti $\mathcal{C}_i := \{C_1, \ldots, C_k\}$, kadar je to možno, izberemo in združimo na naslednji način:

- **Globalna verzija**: Naj bo P množica vseh možnih rezultatov, ki jih dobimo, če združimo dve gruči iz trenutne gručavosti $\mathcal{C}_i$, tj.

  $$P := \{\{C_1, \ldots, C_k\} \backslash \{C_\mu, C_\nu\} \cup \{C_\mu \cup C_\nu\} \mid \mu \neq \nu\}.$$

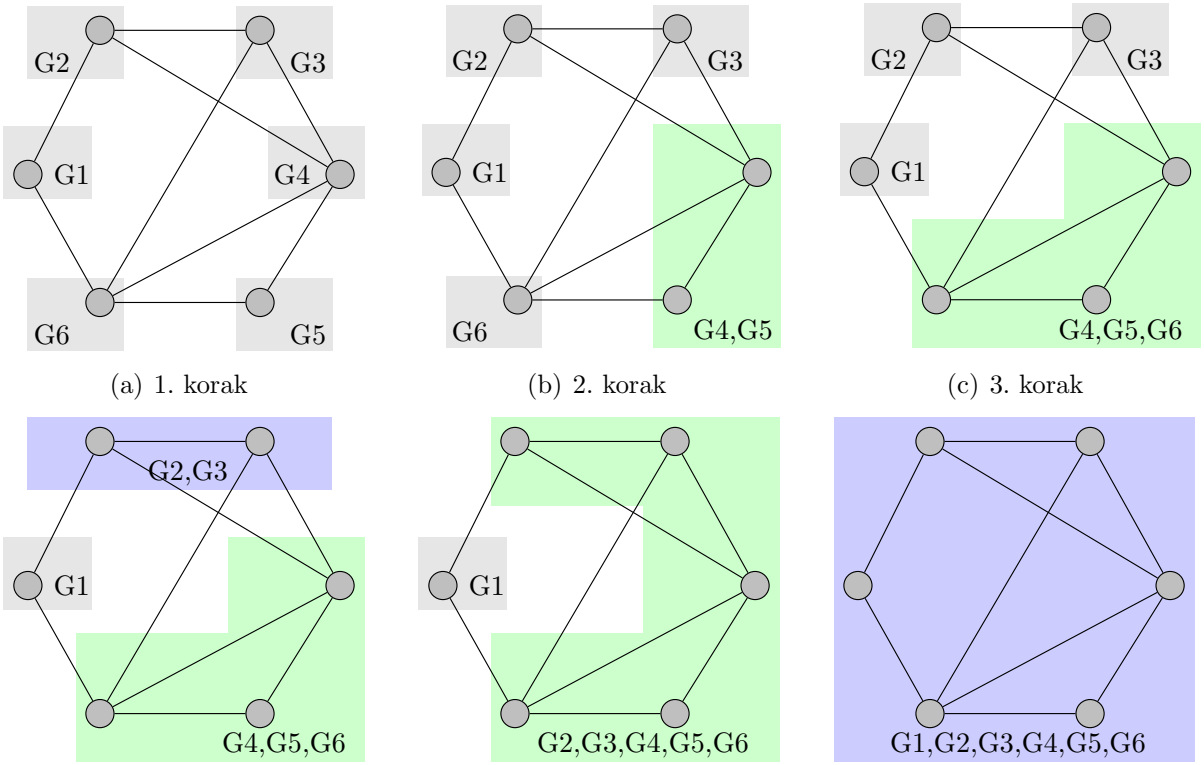
  Nova gručavost $\mathcal{C}_{i+1}$ je določena z:

  $$\mathcal{C}_{i+1} := \text{argmin}_{\mathcal{C} \in P} c_{\text{globalna}}(\mathcal{C}).$$

- **Lokalna verzija**: Naj bosta μ in ν taka različna indeksa, da velja: c_{lokalna} ima en globalni minimum (C_μ, C_ν). Potem je nova gručavost $\mathcal{C}_{i+1}$ definirana z združitvijo C_ν in C_μ, tj.

  $$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \backslash \{C_\mu, C_\mu\} \cup \{C_\mu \cup C_\nu\}.$$

Glavna ideja je izbiranje trenutno najcenejše gručavosti. Ceno operacije združitve dveh gruč lahko določimo na dva načina. Lokalna verzija izračuna ceno le na podlagi gruč, ki ju združujemo. Nasprotni vidik temu pa je globalna verzija, kjer upoštevamo celotno gručavost. Na sliki 6.8 je primer za proces združevanja.



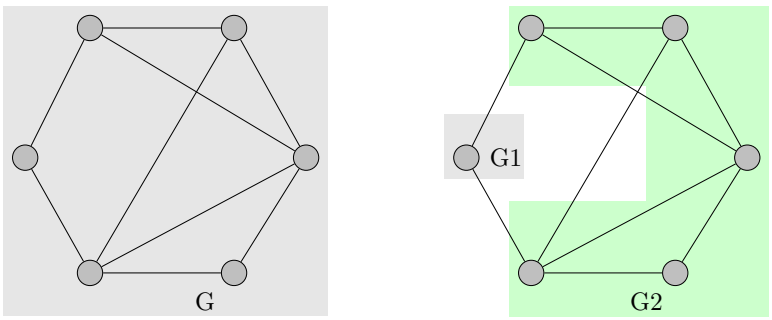
(a) 1. korak (b) 2. korak (c) 3. korak

(d) 4. korak (e) 5. korak (f) 6. korak

Slika 6.8: Proces združevanja.

6.3.3 Proces delitve

Proces delitve je obratni proces združevanju. Pri procesu delitve eno gručo C_i v trenutni gručavosti $\mathcal{C} := \{C_1, \ldots, C_k\}$ razdelimo na dva dela. Primer je na sliki 6.9.



(a) 1. korak (b) 2. korak

Slika 6.9: Prikaz delitve ene gruče v dve novi gruči.

Proces se ustavi, ko se ne da nobene gruče več razdeliti.

Formalna predstavitev procesa delitve: Naj bodo dani:

- graf $G = (V, E, \omega)$ in začetno gručavost $\mathcal{C}_1$;
- globalna ali semi-globalna ali lokalna ali semi-lokalna funkcija cene.

Definirajmo funkcije cene:

- *globalna:* funkcija cene $c_{\text{globalna}} : \mathcal{A}(G) \to \mathbb{R}_0^+$;
- *semi-globalna:* funkcija cene $c_{\text{globalna}} : \mathcal{A}(G) \to \mathbb{R}_0^+$ in pripadajoča funkcija prereza $c_{\text{lokalna}} : \mathcal{P}(V) \to \mathcal{P}(V)$;
- *lokalna:* funkcija cene $c_{\text{lokalna}} : \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$. ;
- *semi-lokalna:* funkcija cene $c_{\text{lokalna}} : \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$ in pripadajoča funkcija prereza $c_{\text{lokalna}} : \mathcal{P}(V) \to \mathcal{P}(V)$.

Gručo v trenutni gručavosti $\mathcal{C}_i := \{C_1, \ldots, C_k\}$, kadar je to možno, izberemo in razdelimo na naslednji način:

- **Globalna verzija:** Naj bo P množica vseh gručavosti, ki jih dobimo iz gručavosti $\mathcal{C}_i$, če razdelimo eno gručo C_i v dve novi gruči, glede na c_{globalna}; tj.

  $$P := \left\{ \{C_1, \ldots, C_k\} \backslash \{C_\mu\} \cup \{C'_\mu, C_\mu \backslash C'_\mu\} \mid \emptyset \neq C'_\mu \not\subseteq C_\mu \right\}.$$

  Novo gručavost $\mathcal{C}_{i+1}$ definiramo kot:

  $$\mathcal{C}_{i+1} := \text{argmin}_{\mathcal{C} \in P} c_{\text{globalna}}(\mathcal{C}).$$

- **Semi-globalna verzija:** Naj bo P množica vseh možnih gručavosti, ki jih dobimo iz gručavosti $\mathcal{C}_i$, glede na S_{lokalna}; tj.

  $$P := \left\{ \{C_1, \ldots, C_k\} \backslash \{C_\mu\} \cup \{S_{lokalna}(C_\mu), C_\mu \backslash S_{\text{lokalna}}(C_\mu)\} \right\}.$$

  Nova gručavost $\mathcal{C}_{i+1}$ je določena kot:

  $$\mathcal{C}_{i+1} := \text{argmin}_{\mathcal{C} \in P} c_{\text{globalna}}(\mathcal{C}).$$

- **Lokalna verzija:** Naj bo μ tak indeks in C_ν taka prava podmožica gruče C_μ, da ima cena funkcije c_{lokalna} globalni minimum v paru $(C_\nu, C_\mu \backslash C_\nu)$. Potem je nova gručavost $\mathcal{C}_{i+1}$ definirana z razdelitvijo gruče C_μ glede na S_{lokalna}, tj.

  $$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \backslash \{C_\mu\} \cup \{C_\mu, C_\mu \backslash C_\nu\}.$$

- **Semi-lokalna verzija:** Naj bo μ indeks, pri katerem ima c_{lokalna} globalni minimum v paru $(S_{\text{lokalna}}(C_\mu), C_\mu \backslash S_{\text{lokalna}}(C_\mu))$. Potem je nova gručavost C_{i+1} definirana z razdelitvijo gruče C_μ glede na S_{lokalna}, tj.

  $$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \backslash \{C_\mu\} \cup \{S_{\text{lokalna}}(C_\mu), C_\mu \backslash S_{\text{lokalna}}(C_\mu)\}.$$

Glavna ideja tukaj je ta, da dobimo najcenejšo gručavost. V nasprotju s procesom združevanja pa imamo pri procesu delitve veliko bolj proste roke. Gručo lahko namreč razdelimo na različne načine. Izračun cene za vse možne prereze in ureditev gruč glede na njihovo ceno se izkaže za časovno zahtevno opravilo in ponavadi zraven zahteva dodatne parametre o funkcijah cene, ki določajo cene prerezov.

Zato uvedemo dodatno funkcijo prereza S_{lokalna}. Semi-globalna in semi-lokalna verzija uporabljata iste principe kot globalna in lokalna verzija, le da je njihova množica potencialnih gruč, ki jih lahko razdelimo, zelo zmanjšana. S pomočjo te dodatne funkcije prereza pridobimo učinkovito izračunljivost, poleg tega pa za razliko od prej ne potrebujemo dodatnih informacij o funkcijah cene. Izbira funkcije prereza je zelo pomemebna, saj bistveno vpliva na kakovost gručavosti, ki jo dobimo.

Oba opisana procesa sta požrešne narave. Na vsakem koraku smo namreč izbrali najcenejšo varianto. Enostavno lahko zgradimo totalno ali kompletno hierarhijo. Totalno hierarhijo lahko dosežemo tako, da dodajamo trivialne gručavosti k hierarhiji, ki smo jo dobili kot rezultat. Pri tem se lastnost hierarhije ohranja, saj so trivialne gručavosti primerljive z ostalimi. Kompletne hierarhije so tiste hierarhije, za katere velja: za vsak $k \in \{1, \ldots, n\}$ je gručavost s k gručami vključena v hierarhijo. Oba procesa nas pripeljeta do kompletne hierarhije, inicializirane s trivialnimi gručavostmi, tj. enojčki pri procesu združevanja in 1-gručavost pri procesu delitve.

Zaradi njune enostavne strukture, predvsem lokalne verzije, sta oba procesa pogosto uporabljena in sta temeljna pri grupiranju.

Splošno lokalno verzijo lahko zelo učinkovito implementiramo. Pri procesu združevanja imamo shranjeno matriko, ki vsebuje cene združevanje za vse možne pare gruč. Ko je na vrsti funkcija za posodobitev podatkov, moramo poračunati samo nove cene vseh gruč z gručo, ki smo jo dobili pri združitvi dveh gruč. Pri procesu delitve je potrebno shraniti edino podatke o prerezih za vsako gručo. Ko je na vrsti funkcija za posodobitev podatkov, moramo na novo izračunati le ceni prerezov za gruči, ki smo ju dobili pri prerezu.

6.3.4 Premična metoda

V nasprotju z globalnimi akcijami prejšnjih požrešnih metod premične metode delujejo bolj lokalno. Izberemo začetno gručavost in jo iterativno spreminjamo, dokler ne najdemo lokalnega optimuma. Ponavadi imamo za to na voljo tri operacije:

- vozlišče premaknemo iz ene gruče v drugo že obstoječo gručo;
- vozlišče premaknemo iz ene gruče v novo gručo, ki jo ustvarimo;
- dve vozlišči zamenjata gruči.

Vselej dovolimo tudi bolj kompleksne operacije, npr. odstranitev obstoječe gruče in reorganizacijo njenih vozlišč v druge že obstoječe gruče. Te kompleksne operacije dovoljujemo npr. zaradi pospešitve postopka, v izogib nekaterim umetnim situacijam ... Predstavljamo si lahko, da so te kompleksne operacije sestavljene iz zaporedja enostavnih operacij.

Ideja premične metode je podobna ideji požrešne metode. V algoritmu 12 je prikazan postopek algoritma za premično metodo. V njem z $N_s(L)$ označimo množico vseh gručavosti, ki jih lahko dobimo, če na gručavosti L izvedemo modifikacije.

Novo gručavost, ki jo moramo izbrati na prvem koraku v zanki algoritma, lahko izberemo naključno oziroma uporabimo potencialno funkcijo, ki nam izbere gručavost. Spodnja tehnična definicija premičnega procesa kot kriterij za selekcijo uporablja potencialno funkcijo.

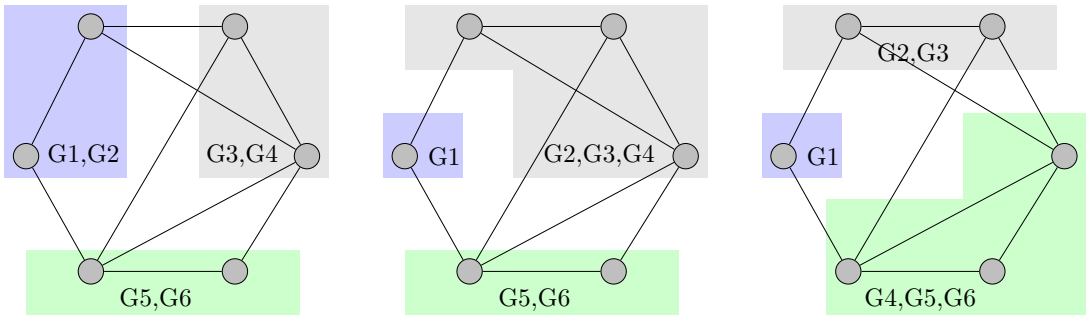
Definicija 6.15 Naj bodo dani graf $G = (V, E, \omega)$, začetna gručavost $\mathcal{C}_1$ in potencialna funkcija $\phi : \mathcal{A}(G) \times \mathcal{A}(G) \to \mathbb{R}$. **Premični proces** je izvedba neke operacije na trenutni gručavosti $\mathcal{C}_i$, ki nam da novo gručavost $\mathcal{C}_{i+1}$, tako da velja $\phi(\mathcal{C}_i, \mathcal{C}_{i+1}) > 0$.

Algorithm 12 Shema premične metode.

Naj bo L_0 začetna rešitev.
$i \leftarrow 0$
while $\{L \mid L \in N_s(L_i), c(L) < c(L_i)\} \neq \emptyset$ **do**
  Izberemo $L_{i+1} \in N_s(L_i)$.
  $i \leftarrow i + 1$
end while
return L_i

Imamo dve vrsti potencialnih funkcij: tipske in kompresijske. **Tipske potencialne funkcije** (*type-based potentials*) so funkcije, ki temeljijo na tipu operacij, ki jih izvajamo. Ker je operacija, ki ustvari novo gručo, v katero premaknemo neko vozlišče iz neke že obstoječe gruče, draga, je ponavadi končno število gruč podobno začetnemu. **Kompresijski premik** (*compressed shifts*) je združitev serije operacij v eno meta-operacijo, pri čemer vrednotimo le izhod te skupne operacije. Tako se izognemo vrednotenju vsake posamezne operacije.

Premične metode ponavadi ne uporabljamo samostojno. Zgodi se, da dobimo zaporedje operacij, pri katerem sta začetna in končna gručavost enaki. Takim zaporedjem rečemo *zanke*. Tudi meje za časovno zahtevnost algoritma je težje določiti kot pri požrešnih metodah. Jih pa zato velikokrat uporabljamo kot dodatni korak pri ostalih metodah, da z njihovo pomočjo rešitev lokalno izboljšamo. Na sliki 6.10 je primer za premično metodo.



(a) 1. korak (b) 2. korak (c) 3. korak

Slika 6.10: Primer za premično metodo.

6.3.5 Splošni optimizacijski pristopi h grupiranju

Pristopi, ki jih bomo opisali, temeljijo na ideji, da lahko gručavost formuliramo kot rezultat nekega splošnega optimizacijskega procesa. Vhodni podatki so lahko generirani na točno določen način z implicitno strukturo gručavosti. Optimizacijski problem je najti gručavost, ki je čim bolj podobno dejanski rešitvi. Tehnik s katerimi lahko rešimo dani optimizacijski problem, je ogromno; mi si bomo pogledali naslednje tri:

- parametrično ocenjevanje;
- evolucijski pristop;

- iskalni pristop (*search-based approaches*).

Parametrično ocenjevanje

Princip temelji na predpostavki, da so vhodni podatki ustvarjeni z naključnim vzorčnim procesom. Postopek potem poskuša oceniti parametre tega procesa vzorčenja. Ti parametri bodo uporabljeni za rekonstrukcijo skritega grafa. Prvotno so bile te metode razvite za iskanje gručavosti za podatke, vložene v metrične prostore. Obstajajo gručavosti, ki bi jih lahko predstavili z unijo porazdelitve - distribucije. Cilj je oceniti število porazdelitev in njihovih parametrov (matematično upanje, standardni odklon ...).

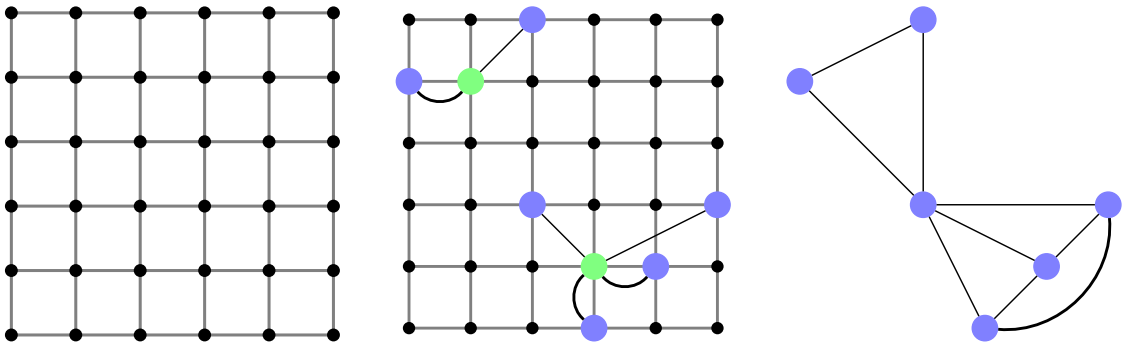
Prvotno so te metode uporabljali za iskanje gručavosti na podatkih, ki so bili predstavljeni v metričnih prostorih. Več o tem lahko bralec prebere v poglavju 3.6 v [125].

Maksimizacija upanja (EM) je najbolj pogosto uporabljena metoda. V splošnem jo apliciramo le na podatke, vložene v metrične prostore. Čeprav grafi pogosto niso taki, lahko vedno v grafe "vključimo" implicitno topologijo. Poglejmo si primer.

Zgled 6.16 Dani naj bodo naslednji podatki:

- končni prostor z metriko;
- množica točk, ki so centri gruč;
- porazdelitve za vse centre gruč.

Potem lahko n vozlišč predstavimo z izbiro centrov gruč in prostorom okrog njih, ki odgovarja tako topologiji kot tudi njeni porazdelitvi. Dve vozlišči sta med seboj povezani, če je njuna razdalja manjša od danega parametra. Na sliki 6.11 je primer tega postopka.



(a) Začetna topologija.

(b) Dva centra gruč (zelena barva) in šest vozlišč (modra barva).

(c) Končni graf.

Slika 6.11: Primer generiranja grafa in njegove gručavosti z uporabo porazdelitev.

V povezavi s parametričnim ocenjevanjem in gručavostjo je leta 2007 izšel zanimiv članek z naslovom "Software Project Efoort Estimation Based on Multiple Parametric Models Generated Through Data Clustering" (bralec ga najde v [119]).

Evolucijski pristopi

Evolucijski pristopi, kot so npr. genetski algoritmi (GA), evolucijske strategije (ES) in evolucijsko programiranje (EP), iterativno spreminjajo končno populacijo z nekimi operacijami. Križanje in mutacija sta najbolj pogosti operaciji, ki ju izvajamo na populaciji. Križanje ustvari nov osebek s kombiniranjem dveh že obstoječih osebkov. Mutacija je operacija, ki spremeni enega izmed že obstoječih osebkov. Vsakemu osebku pripišemo funkcijo uspešnosti. Po nekaj operacijah je ustvarjena nova populacija, ki temelji na obstoječi populaciji in ima osebke izbrane glede na funkcijo uspešnosti. Pri tem moramo zagotoviti, da je spremenjena populacija tudi dopustna rešitev. To ponavadi zagotovimo z ustreznim modelom. Pri problemu gručavosti model običajno uporablja particije ali ekvivalenčne relacije.

V zadnjih letih, ko število uporabnikov svetovnega spleta in raznih družabnih socialnih omrežij strmo narašča, se vedno več raziskuje tudi na področju evolucijskih pristopov v povezavi z gručavostjo. V [123] in [130] najdemo nekaj več o povezavi med gručavostjo in evolucijskimi pristopi.

Iskalni pristop

Iskalni pristop uporablja dano topologijo nekega izbranega prostora in predstavlja naključni sprehod iz izbranega kandidata. Podobno kot pri evolucijskem pristopu je tudi tu sosedstvo izbranega kandidata definirano kot rezultat preprostih operacij, kot so npr. mutacije. Sosedstvo gručavosti je ponavadi množica gručavosti, ki jih dobimo denimo s premiki vozlišč, združevanjem ali delitvijo gruč. Izbira sosedstev temelji na nekaterih funkcijah cene, običajno optimizacijske funkcije, ki jo izračunamo v nekem kandidatu. Iskanje se običajno ustavi bodisi po doseženem številu vnaprej predpisanih iteracij bodisi ko najdemo lokalni optimum, lahko pa tudi kombinaciji obojega.

V povezavi z iskalnimi in evolucijskimi pristopi je bil leta 2003 izdan članek za naslovom "Evolutionary Clustering and Analysis of Bibliographic Networks" (bralec ga najde v [115]).

6.4 Algoritmi za grupiranje

Pogledali si bomo vhodne podatke za proces združevanja in proces delitve.

6.4.1 Vhodi za proces združevanja

Različni vhodi za delovno okolje procesov združevanja so bili prvotno zasnovani za utežene grafe, kjer je bila utež povezave uv kar razdalja povezave. Razdalje so dualna verzija podobnosti v grafih. Zato se pogosto zgodi, da namesto podobnosti uporabimo funkcijo razdalje; to pomeni, da iščemo prostorsko goste gruče, ki so med sabo dobro ločene. Problem nastane, če je funkcija razdalje znana le delno, saj tedaj ne moremo izpeljati podobnosti dveh objektov s pomočjo njunih razdalj do ostalih objektov. Po drugi strani pa lahko razdalje preprosto kombiniramo in na ta način lahko dobimo dolžino poti.

Standardne lokalne funkcije cene definiramo kot:

$$c_{\text{lokalna}}(C_i, C_j) := \bigodot\{d(u,v) \mid u \in C_i, v \in C_j\}, \tag{6.2}$$

kjer je $d(u,v)$ dolžina najkrajše poti med u in v in $\odot$ je neka funkcija, definirana na množici, npr. povprečni, minimalni ali maksimalni element množice. Glede na funkcijo $\odot$ imamo naslednje tri verzije: proces povprečnega združevanja (*Average Linkage*), proces samega združevanja (*Single Linkage*) in proces popolnega združevanja (*Complete Linkage*).

Opazimo, da je lahko funkcija cene asimetrična in ima lahko za vrednost tudi neskončno. Opazimo še, da se ni potrebno ukvarjati z dolžino najkrajših poti, saj bo vsak par vozlišč $(u,v) \in C_i \times C_j$ povezan z eno povezavo. V bistvu bo v idealnem primeru množica $E(C_i, C_j)$ prazna.

Kadar pa namesto z razdaljami delamo s podobnostmi, je potrebno na pravi način definirati dolžino poti. Najbolj enostavno pa je, da dolžino poti zanemarimo in definiramo funkcijo cene kot:

$$c_{\text{lokalna}}(C_i, C_j) := \bigodot\{M - \omega(e) \mid e \in E(C_i, C_j)\}, \tag{6.3}$$

kjer je M maksimalna utež na povezavah v grafu. Alternativno lahko definiramo podobnost poti $P : v_1, \ldots, v_\ell$ z:

$$\omega(P) := \left(\sum_{i=1}^{\ell-1} \frac{1}{\omega(v_i, v_{i+1})} \right)^{-1}. \tag{6.4}$$

Čeprav je ta definicija kompatibilna s trikotniško neenakostjo, pa propade pri nekaterih drugih lastnostih. Še ena definicija, ki je pogosto uporabljena v kontekstu s porazdelitvami, je

$$\omega(P) := \prod_{i=1}^{\ell-1} \omega(v_i, v_{i+1}). \tag{6.5}$$

Če je $\omega(v_i, v_{i+1})$ verjetnost, da je povezava (v_i, v_{i+1}) vključena in so verjetnosti neodvisne med seboj, potem je $\omega(P)$ verjetnost, da celotna pot P obstaja. Dokaz naslednje leme lahko bralec najde v [125].

Lema 6.17 *Dendrogram za proces samega združevanja definiramo s pomočjo minimalnega vpetega drevesa.*

Delovno okolje za proces združevanja pogosto gledamo v kontekstu redkih omrežij in v omrežjih z majhnim pričakovanim številom povezav med gručami. To podkrepimo z opazko, da veliko verzij procesov združevanja stremi h gradnji verige gručavosti.

6.4.2 Vhodi za proces delitve

Čeprav je cela vrsta prerezov, ki jih imamo, je ideja ti. redkih prerezov (*sparse cuts*), ki ločijo različne gručavosti med sabo, med najbolj uporabnimi. Med temi so:

- standardni prerez (*Standard cuts*):

$$S(V) := \min_{\emptyset \neq V' \subset V} \omega(E(V', V \backslash V'));$$

- prerez glede na razmerje (*Ratio cuts*):

$$S_{\text{razmerje}}(V) := \min_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \backslash V'))}{|V'| \cdot (|V| - |V'|)};$$

- uravnovešeni prerez (*Balanced cuts*):

$$S_{\text{balanced}}(V) := \min_{\emptyset \neq V' \subset V} \frac{\min(\omega(E(V', V \backslash V'))}{|V'| \cdot (|V| - |V'|))};$$

- prerez glede na prevodnost(*Conductance Cuts*):

$$S_{\text{prevodnost}}(V) := \min_{\emptyset \neq V' \subset V} \delta(V')$$

- bisektor (*Bisectors*):

$$S_{\text{2-sektor}}(V) := \lim_{\substack{V' \subset V \\ \lfloor |V|/2 \rfloor \leq |V'| \leq \lceil |V|/2 \rceil}} \omega(E(V', V \backslash V')).$$

Prerezi glede na razmerje, uravnovešeni prerezi in 2-sektorji so običajno uporabljeni, kadar je enotnost velikosti gruč pomembna. Večina izmed teh mer je $\mathcal{NP}$-zahteven problem, zato namesto njih uporabljamo aproksimativne ali hevristične algoritme. Opazimo, da imajo uravnovešeni prerezi in prerezi glede na prevodnost isto osnovno idejo, na kateri temeljijo: razvrščanje velikosti/teže prerezov glede na velikosti/tež manjših induciranih prerezov.

Proces delitve ponavadi pride do izraza, če delamo z gostimi omrežji oziroma omrežji, kjer je pričakovano število povezav med posameznimi gručami izredno veliko. Taka omrežja so npr. omrežja za modeliranje genskih zapisov (več o tem lahko bralec najde v [124]). Proces delitve namreč teži k temu, da ustvari majhne in zelo goste gruče.

6.4.3 Nestandardni vhodi za procesa delitve in združevanja

Obstaja vrsta algoritmov, ki uporablja podobne operacije kot procesa delitve in združevanja, vendar ne sodi v okvir teh dveh procesov. Zato take algoritme obravnavamo posebej. Poglejmo si nekaj primerov teh algoritmov.

Identifikacija mostov

Identifikacija mostov je podobna procesu delitve, kjer prereze zamenjamo s povezavami ali podmnožicami vozlišč, ki nam pomagajo odkriti individualne gruče. Skoraj vse tehnike za identifikacijo mostov temeljijo na strukturnih indeksih ali nekih značilnostih, ki jih izpeljemo iz najkrajših poti ali izračuna pretoka. V [129] pa je predstavljena tudi uporaba centralnih indeksov pri identifikaciji.

Večstopenjski pristopi

Večstopenjski pristopi so posplošitev procesa združevanja, kjer grupe vozlišč postopoma združujemo v eno samo vozlišče, dokler ne dobimo problema v obliki, ki ga znamo rešiti. Na koncu pa moramo seveda rešitev prevesti na prvoten graf. Skozi proces se seveda gruče,

ki jih dobimo na nekem koraku, ne ohranijo. Za razliko od prej lahko tukaj "raztrgamo" gručo na več delov. Ponavadi se taki pristopi uporabljajo, ko želimo najti k gruč podobnih velikosti, med katerimi je zelo malo povezav. V tem primeru se ta pristop lahko uspešno kombinira s premičnimi metodami.

Bibliografske opombe

V [125] in [127] lahko bralec najde širše zastavljeno uvodno spoznavanje z različnimi metodami grupiranja. Obravnavane metode povečini temeljijo na ideji rudarjenja s podatki, poleg teh pa je zajetih še dosti ostalih pristopov. Eden izmed pristopov je npr. delitev na gruče, ki so podobne velike. Pretežno se uporablja v metodah, ki delujejo po principu deli in vladaj. Veliko problemov pa se rešuje tudi s pomočjo funkcij prereza in premičnih metod.

Izdelava čipov (VLSI [134]) je pomembno raziskovalno področje, ki je razvilo vrsto takšnih algoritmov. V [112] je predstavljena raziskava, ki zajema splošne metode in njihov pomen v VLSI.

V splošnem se probleme iskanja dobrih particij in aproksimacijskih redkih prerezov ter računanje pretokov rešuje s pomočjo (celoštevilskega) linearnega programiranja. Uvodno znanje o tem si lahko bralec pridobi v [128] in [132].

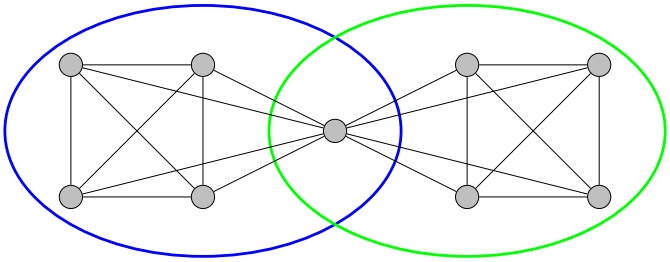
Zanimiva področja v povezavi z gručavostjo so še: umetna inteligenca, nevronske mreže, računalniška obdelava slik ([120]), genetika ([124]) in postavitev objektov ([128], [132]).

6.5 Gručavost - alternativni pristopi

V prejšnjih poglavjih smo obravnavali nekatere vidike teorije, ki nam že dajo prvi občutek za uporabnost te teorije in za temo kot celoto. A ostaja še veliko aspektov, s katerih lahko pristopimo k teoriji. V tem zadnjem delu si bomo ogledali dva - razširitve gručavosti in aksiomatiko.

6.5.1 Prehodna gručavost

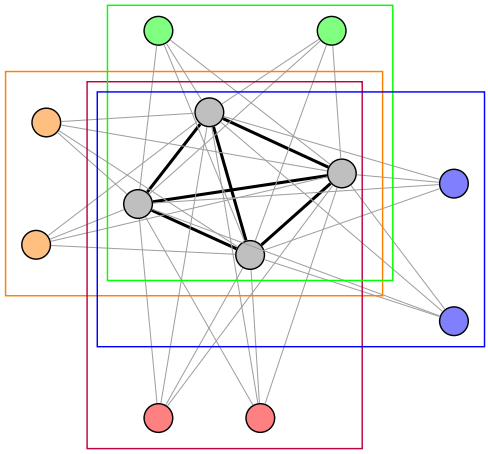
Pri prehodni gručavosti (*Fuzzy clustering*) drugače gledamo na prekrivanje gruč. Osnovna ideja je, da prerezno vozlišče pripada hkrati obema gručama, ki ju povezuje.



Slika 6.12: Primer prehodne gručavosti. Sredinsko vozlišče pripada hkrati levi in desni gruči.

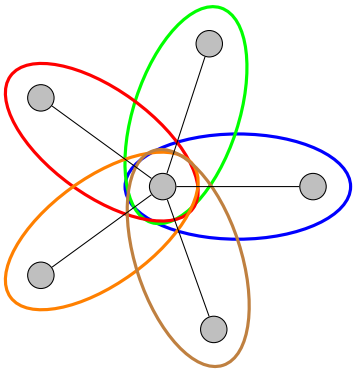
Ta pristop se redko uporablja, saj ga je včasih težko interpretirati. Težave se pojavijo, če vozlišče ali manjša množica vozlišč pripada mnogim (prehodnim) gručam. Če denimo

omejimo število gruč z neko konstanto k, potem je problem, ko ima več kot k klik velikosti vsaj k veliko skupnih vozlišč. Na sliki 6.13 imajo štiri klike velikosti 6 skupen K_4, vsaka je prehodna gruča.



Slika 6.13: Štiri klike velikosti 6 s skupnim K_4. Vsaka maksimalna klika je prehodna gruča.

V primeru, da ima vozlišče stopnjo primerljivo s k, lahko redke gruče razpadejo. Na sliki 10.14 je primer zvezde s 5 listi, ki jo lahko razdelimo na 5 prehodnih gruč, kjer vsaka vsebuje središče in en list.



Slika 6.14: Pri tej zvezdi je pet prehodnih gruč, vsako sestavlja skupno središče in en list.

6.5.2 Gručavost s predstavnikom

Vsaka gruča ima predstavnika, ki je običajno neke vrste centralni element. Prednost tega pristopa se pokaže še posebej takrat, ko je graf vložen v metrični ali vektorski prostor, saj je lahko tedaj predstavnik gruče element prostora in ne nujno vhodnih podatkov. Uporablja se za pohitritve in aproksimativno računanje. Denimo, da potrebujemo razdalje med vsemi vozlišči v dveh gručah. Tedaj je za približen rezultat dovolj izračunati le razdalje med predstavniki gruč.

6.5.3 Gnezdena gručavost

Definicija 6.18 **Gnezdena gručavost** je gnezdeno zaporedje podmnožic vozlišč, tj. preslikava $\eta : \mathbb{N} \to \mathcal{P}(V)$ tako da:

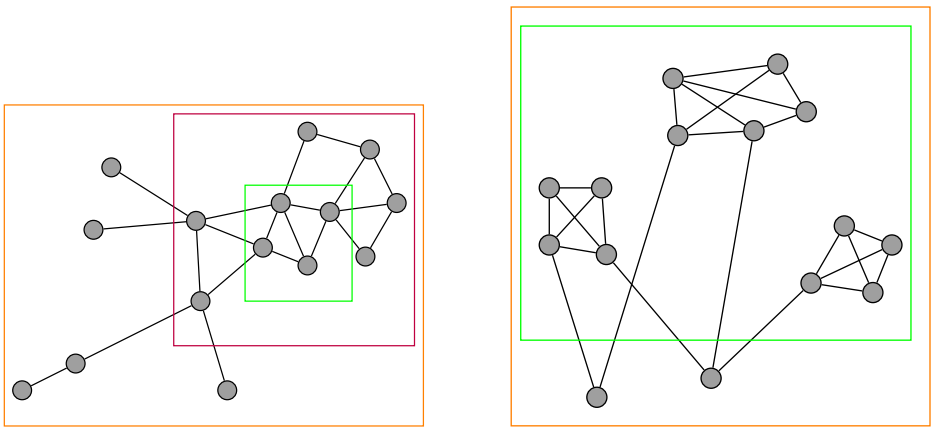
1. množice so gnezdene, tj.
$$\forall i \in \mathbb{N} \,.\, \eta(i+1) \subseteq \eta(i);$$

2. zaporedje je končno, tj.
$$\exists k \in \mathbb{N} \,.\, \forall \ell > k \,.\, \eta(\ell) = \emptyset$$

Najmanjši možni k imenujemo **velikost** zaporedja in $\eta(k)$ **vrhnji element**. Vrhnji element je nekako lokalno "najgostejša" množica. Gostota podmnožic $\eta(i)$ je naraščajoča funkcija argumenta i, to pomeni, da lahko i gledamo kot stopnjo gostote.

Razlikujemo dva ekstremna tipa gnezdene gručavosti. Prvi je **hierarhija**, pri kateri $\eta(i)$ inducira povezan graf. Imamo nek najgostejši vrhnji element, okrog njega pa kot pri čebuli vedno redkejše sloje. Drugi tip so **vrhovi**, ki jih ravno nasprotno definira vsebnost vsaj ene podmnožice $\eta(i)$, ki definira nepovezan graf. To si lahko predstavljamo kot vrelo vodo, kjer je na površju več mehurčkov, ločuje pa jih hladnejša voda.

Če je graf izrazito gručast, je večja verjetnost, da se bodo pojavili vrhovi. V nasprotnem primeru bomo imeli hierarhijo. Posebni primeri gruč, k-jedra, so povezane komponente, ki ostanejo po odstranitvi vseh vozlišč s stopnjo manjšo od k. Več o jedrih si lahko bralec prebere v [116], [118] in [131].



(a) Hierarhija. (b) Vrhovi.

Slika 6.15: Ekstremna primera gnezdene gručavosti.

6.6 Aksiomatika

Z aksiomi povzemamo splošne značilnosti in bistvena načela teorije. Omogočajo nam, da nato dokazujemo lastnosti le za ta sistem aksiomov in s tem pokažemo veljavnost za vse primere, ki jim zadoščajo. V nadaljevanju je predstavljen Kleinbergov predlog aksiomatike [126].

Naj bo $K_n = (V, E)$ poln graf na n vozliščih in $\omega : E \to \mathbb{R}_0^+$ funkcija uteži na povezavah. Množico vseh možnih funkcij uteži označimo z D. Imejmo še preslikavo $f : D \to \mathcal{A}(K_n)$.

Ekvivalenčno relacijo $\sim_{f(\omega)}$ na vozliščih grafa K_n, kjer je ω iz D, definiramo na naslednji način:

$u \sim_{f(\omega)} v \iff u$ in v sta sosednji vozlišči znotraj iste gruče v sliki oz. gručavosti $f(\omega)$.

Definicija 6.19 Funkcija grupiranja f je preslikava $f : D \to \mathcal{A}(K_n)$, ki zadošča naslednjim aksiomom:
Invariantnost na merilo: $\forall \alpha \in \mathbb{R}^+, \omega \in D \,.\, f(\omega) = f(\alpha \cdot \omega)$, kjer je $(\alpha \cdot \omega)(u, v) := \alpha \cdot (\omega(u, v))$;
Bogatost (surjektivnost): za vsako gručavost C v $A(K_n)$ obstaja ω v D tako da je $f(\omega) = C$;
Konsistentnost: za vse $\omega, \omega' \in D$ velja

$$\left(\forall u, v \in V \,.\, \omega'(u, v) \begin{cases} \leq \omega(u, v) & u \sim_{f(\omega)} v \\ \geq \omega(u, v) & \text{sicer} \end{cases}\right) \to f(\omega) = f(\omega').$$

Intuicija za to definicijo je naslednja. *Invariantnost na merilo* zagotavlja, da gručavost ni odvisna od fiksno določenih vrednosti, temveč od razmerij. Če vse razdalje homogeno povečamo, ostane gručavost enaka. *Bogatost* je neke vrste surjektivnost - zagotavlja, da ima vsaka gručavost vsaj eno obteženje za prasliko. Vsaka gručavost je konstruktibilna z dodelitvijo ustreznih uteži na povezave. *Konsistentnost* razlaga odnos med različnimi gručavostmi. Denimo, da je ω fiksiran. Tedaj je $f(\omega)$ gručavost. Ohraniti želimo $f(\omega)$ in gruče, spremeniti pa uteži na povezavah. Modifikacijo na ω lahko naredimo le tako, da znotraj posamezne gruče razdalje oz. uteži kvečjemu zmanjšamo (gruča postane bolj kompaktna), med njimi pa povečamo (gruče so bolj ločene).

Izrek 6.20 ([126]) *Za vse $n \geq 2$ ne obstaja funkcija grupiranja.*

Izrek je negativni rezultat za funkcije grupiranja in posledično algoritme. Dani aksiomi so zelo restriktivni, obstaja pa veliko funkcij, ki jim skoraj zadoščajo.

Lema 6.21 ([126]) *Obstaja veliko funkcij, ki zadoščajo samo dvema od treh aksiomov iz definicije.*

Izkazalo se je, da sprostitev pogojev invariantnosti na merilo in konsistentnosti vodi do funkcij grupiranja. Druga pot je raziskovanje "pomanjkanja informacij"(v našem primeru povezav). Običajno grafi niso polni in jih tudi ne moremo preprosto dopolniti do takih. Standardna tehnika je taka, da grafu dodamo manjkajoče povezave, a jih utežimo z ekstremno velikimi vrednostmi. Vendar moramo imeti tu vse vrednosti končne, invariantnost na merilo in konsistentnost pa omogočata manipulacije z vrednostmi, zato ni nujno, da bodo na začetku ekstremne vrednosti to vlogo tudi ohranile.

Da lahko upoštevamo to pomanjkanje informacij, definiramo funkcijo grupiranja kot preslikavo iz množice uteženih razmerij v množico particij neke množice. Z $\Omega(X)$ označimo množico vseh uteženih (binarnih) relacij nad X. Za vsako relacijo $\omega \in \Omega$ je domena dana z $E(\omega)$.

Definicija 6.22 Naj bo dana množica elementov X. **Funkcija grupiranja grafa** f je preslikava $f : \Omega(X) \to \mathcal{A}(X)$, ki zadošča naslednjim aksiomom.

Invariantnost na merilo: $\forall \alpha \in \mathbb{R}^+, \omega \in \Omega(X) \,.\, f(\omega) = f(\alpha \cdot \omega)$, kjer je $(\alpha \cdot \omega)(u,v) := \alpha \cdot (\omega(u,v))$;
Bogatost (surjektivnost): za vsako gručavost C v $A(K_n)$ obstaja ω v $\Omega(X)$ tako da je $f(\omega) = C$;
Konsistentnost: za vse $\omega, \omega' \in \Omega(X)$, pri čemer je $E(\omega) \subseteq E(\omega')$, velja

$$\left(\forall u,v \in V \,.\, \omega'(u,v) \begin{cases} \leq \omega(u,v) & u \sim_{f(\omega)} v \\ \geq \omega(u,v) & \text{sicer} \end{cases}\right) \Rightarrow f(\omega) = f(\omega'),$$

kjer je $\omega(u,v) = \infty$ za $(u,v) \in E(\omega') \backslash E(\omega)$.

Definiciji sta ekvivalentni. Pogoj za konsistentnost je bolj zapleten, ker lahko imajo relacije iz Ω različne domene. Množica D vseh tehtanj (polnega grafa) je seveda podmnožica Ω. Ime je dobila funkcija po tem, da lahko na X in $\omega \in \Omega$ gledamo kot na graf: $G_\omega = (X, E(\omega), \omega)$. Torej funkcija f slika množico (uteženih enostavnih) grafov z n elementi v množico particij n elementov.

Lema 6.23 *Funkcija f_{comp}, ki slika $\omega \in \Omega$ v gručavost, kjer so gruče povezane komponente G_ω, je funkcija grupiranja grafa.*

Dokaz. Za preverjanje aksiomov je dovolj pogledati, če ima f_{comp} pravi domeno in kodomeno. Prvemu in tretjemu pogoju je zadoščeno, saj nima $f_{comp}(\omega)$ za noben $\omega \in \Omega$ povezav znotraj gruč. Povezave sicer lahko dodajamo preko aksioma o konsistentnosti, vendar jih ne moremo dati znotraj gruč, saj se mora ohranjati gručavost $f_{comp}(\omega)$. Tudi drugemu pogoju je zadoščeno. Za vsako grupiranje $\mathcal{C}$ namreč obstaja vpeti gozd F z istimi povezanimi komponentami kot v gručah. Njegove relacije na povezavah E_F inducirajo uteženo relacijo ω preko $\omega(u,v) = 1$, če je $(u,v) \in E_F$. Tedaj bo to relacijo ω v $\mathcal{C}$ preslikala $f_{comp}(\omega)$.

□

Zadnji definiciji in lema nakazujeta, da je lahko pomanjkanje bolj zgovorno od popolnih informacij. Drugače kot v definiciji funkcije grupiranja grafa lahko pogoj za konsistentnost prilagodimo še na alternativne načine: omejimo se na gručo ali povezanost med gručami, uvedemo homogeno skaliranje ali pa vnaprej določene prelomne/stične točke za kontrolirane razcepe/združitve. Te ideje so obetavne, a večinoma še neraziskane.

6.7 Zaključek

Naravna dekompozicija je glavno vodilo iskanja gruč v grafu. Matematični model tega je particija vozlišč, med katerimi nato opazujemo gostoto povezav znotraj posamezne gruče v primerjavi z gostoto v vmesnem prostoru.

Vendar pa sta definicija gručavosti kot naravne dekompozicije kot tudi zgornji model zgolj intuitivna, brez pravih matematičnih parametrov. Če želimo doseči večjo formalnost, moramo vpeljati indekse, ki bodo merili to razmerje gostot. Dodatna pozitivna posledica je, da nam je omogočena primerjava gručavosti različnih grafov. Velika uporabnost tega se pokaže predvsem pri delu z grafi z določeno stopnjo negotovosti. Žal pa je večina problemov, povezanih z indeksi, $\mathcal{NP}$-polnih.

Po uvodnem prvem poglavju so bile v začetnem delu drugega poglavja najprej predstavljene osnovne metode grupiranja. Te so preproste, denimo iterativno spajanje ali delitve

gruč, koncepti premičnih metod in osnovne tehnike optimizacije. Predstavljene tehnike so spremenljive in lahko vsebujejo še veliko število problemsko specifičnih zahtev. Drugi del je govoril o parametrih, ki nastopajo v teh pristopih. Seminarsko nalogo zaključuje poglavje z nekaj razširitvami teorije, ki so prej predstavljenim precej podobne. Gre za to, da so dani podatki, kako deliti elemente, razširjeni z dodatnimi informacijami ali pa zamenjani z drugimi načini dekompozicij. Nazadnje je predstavljen še predlog aksiomatskega sistema za karakterizacijo metod grupiranja. Metode v drugem poglavju so se razlikovale po različnih idejah, na katerih so temeljile, aksiomi pa skušajo zaobjeti vse.

Teorija gručavosti na grafih nedvomno ponuja veliko uporabno vrednost. Nekatere izpeljane tehnike že zelo dobro razumemo tako z vsakdanjega kot tudi matematičnega stališča. Formalizacija 'naravne dekompozicije' in stroga znanstvena podlaga sta nujna za kvalitetno grupiranje. Žal temeljitega in homogenega matematičnega sistema še ni. Tu se ponuja odprt izziv, katerega rezultat mnogi težko pričakujejo. Uporabna vrednost teorije je že danes velika, v prihodnje pa se bo zagotovo pojavljalo še več problemov, na katere bo znala odgovoriti.

Poglavje 7

Dodelitve vlog

Andreja Zorko, Lučka Lenič, Lea Letnar, Klavdija Jagar

7.1 Uvod

Za razumevanje velikih in kompleksnih sistemov, ki so zgrajeni iz več posameznih delov, je ključno razvrščanje. Na primer, v študiji prehranjevalnih mrež je celo za preproste ekosisteme nemogoče razumeti razmerja med posameznimi organizmi. Sistem pa lahko do neke mere vseeno razumemo z razvrščanjem posameznikov in opisom razmerij med razredi. Cilj razvrščanja v omrežjih je opisati regularne vzorce interakcij in izpostaviti njihovo bistveno strukturo, ki ostane stabilna po dolgem časovnem obdobju.

V tem seminarju bomo formalizirali razvrščanje vozlišč v grafu. Rekli bomo, da vozlišča iz istega razreda zavzemajo isto *pozicijo* oziroma igrajo isto *vlogo* v omrežju. To idejo pozicije oziroma vloge v omrežju sta s posebnim tipom particije vozlišč prva predstavila Lorrain in White [155]. Trdila sta, da vozlišča igrajo isto vlogo, če imajo identične soseščine. Poznejša dela Sailerja [156], Whitea in Reitza [157] so to zgodnjo definicijo posplošila tako, da je bila bolj primerna za modeliranje socialnih vlog. Vse te definicije imajo skupno to, da morajo vozlišča, za katera trdimo, da igrajo isto vlogo, imeti nekaj skupnega z ostalimi vozlišči. To pomeni, da lahko splošno definicijo za to poglavje podamo z

- danim grafom $G = (V, E)$,
- iskanjem particije vozlišč V, ki je *združljiva* z E.

Generični del tukaj je izraz 'združljiv z E'. V tem seminarju bomo predstavili definicije za take združljivostne zahteve in lastnosti razredov, dobljenih iz particij vozlišč. Obravnavali bomo notacijo, nato pa raziskali različne tipe dodelitev vlog in delno urejenost na tej množici, predstavili algoritem za izračun določenih elementov, obravnavili kompleksnost nekaterih odločitvenih problemov ter prilagodili nekatere definicije za grafe, ki bodo običajno usmerjeni, z možnimi zankami, razen če bo posebej navedeno. Za vsak tip particij vozlišč si bomo pogledali še uporabnost dodeljevenja vlog v empiričnih omrežjih. Najbolj se bomo posvetili regularnim ekvivalencam, saj so primerne za raziskavo tipov

dodelitev vlog. Rezultate regularnih ekvivalenc se da pogosto prevesti na druge tipe ekvivalenc, zato bomo, kadar bo primerno, to tudi izpostavili.

7.1.1 Oznake in definicije

V nadaljevanju bomo pogosto preklapljali med particijami vozlišč, ekvivalenčnimi relacijami na množici vozlišč in dodelitvami vlog, saj je od konteksta odvisno, kateri pogled je bolj intuitiven kot ostali. Drugače pa so to samo tri različne formulacije istega osnovnega koncepta.

Naj bo V množica. Ekvivalenčna relacija $\sim$ je binarna relacija na V, ki je *refleksivna*, *simetrična* in *tranzitivna*. Če je $v \in V$, potem je $[v] = \{u; u \sim v\}$ njegov *ekvivalenčni razred*. *Particija* $\mathcal{P} = \{C_1, \ldots, C_k\}$ množice V je množica nepraznih, disjunktnih podmnožic $C_i \subseteq V$, imenovanih *razredi* ali *bloki*, tako da je $V = \bigcup_{i=1}^{k} C_i$. To pomeni, da je vsako vozlišče $v \in V$ v natanko enem razredu. Če je $\mathcal{P} = \{V\}$, pravimo, da je particija *polna*. Množica ekvivalenčnih razredov ekvivalenčne relacije $\sim$ na V je particija vozlišč V. Če definiramo, da sta dve vozlišči ekvivalentni natanko tedaj, ko pripadata istemu razredu v $\mathcal{P}$, velja tudi obrat, da particija $\mathcal{P}$ inducira ekvivalenčno relacijo. Ti dve preslikavi sta si inverzni.

Definicija 7.1 *Dodelitev vlog* za V je surjektivna preslikava $r : V \longrightarrow W$ na neko množico vlog W.

Brez škode za splošnost lahko zahtevamo surjektivnost, saj lahko preslikavo vedno zožimo na njeno sliko. Dodelitve vlog lahko smatramo tudi kot barvanje vozlišč, kjer ne zahtevamo, da morata imeti sosednji vozlišči različni barvi. Dodelitev vlog definira particijo množice V, če za razrede vzamemo praslike $r^{-1}(w) = \{v \in V; r(v) = w, w \in W\}$. Obratno, ekvivalenčna relacija s preslikavo $v \mapsto [v]$ na V inducira dodelitev vlog. Ti preslikavi sta si inverzni do izomorfizma množice vlog natančno. To bomo povzeli v naslednji opombi.

Opomba 7.2 Za vsako particijo vozlišč je enolično določena ekvivalenčna relacija in enolično določena dodelitev vlog in isto velja za vse ostale kombinacije.

V preostanku seminarja se bodo definicije za particije vozlišč prevedle na ustrezne ekvivalenčne relacije in dodelitve vlog.

7.1.2 Graf vlog

Slika dodelitve vlog se lahko naravno predstavi s strukturo grafov. Definirali bomo, da sta vlogi sosednji, če igrajo sosednja vozlišča naslednje vloge:

Definicija 7.3 Naj bo $G = (V, E)$ graf in $r : V \longrightarrow W$ dodelitev vlog. *Graf vlog* $R = (W, F)$ je graf z množico vozlišč W (tj. množica vlog) in množico povezav $F \subseteq W \times W$, definirano z

$$F := \{(r(u), r(v)); \exists u, v \in V,\ da\ je\ (u, v) \in E\}.$$

R imenujemo tudi kvocient grafa G nad r.

Graf vlog R modelira vloge in njihove relacije. Gledamo ga lahko tudi kot manjši model originalnega grafa G. Tako lahko dodelitev vlog gledamo kot neke vrste stisk omrežja, pri

čemer se nekaj informacij izgubi. Cilj analize vlog je poiskati take dodelitve vlog, da bo dobljen graf vlog predstavil bistvene strukturne lastnosti omrežja, tj. da se ne bo izgubilo preveč informacij.

Tako imamo dve različni motivaciji za iskanje dobrih dodelitev vlog. Kot prvo želimo vedeti, kateri posamezniki (vozlišča) so si 'podobni'. Kot drugo pa želimo reducirati kompleksnost omrežja: če je omrežje zelo veliko ali nepravilno, ne moremo zajeti njegove strukture na nivoju posameznikov (vozlišč), ampak morda le na nivoju skupnosti (vlog). Upamo torej, da bo graf vlog izpostavil bistveno strukturo omrežja. Medtem ko posamezniki pridejo in gredo ter se nepravilno obnašajo, od vlog pričakujemo, da bodo (vsaj za daljše časovno obdobje) ostale stabilne in prikazale nek reden vzorec medsebojnega vpliva.

7.2 Strukturna ekvivalenca

Kot smo omenili že v uvodu, je cilj analize vlog poiskati smiselne particije vlog, ki so združljive s povezavami grafa. V tem poglavju bomo definirali najbolj preprosto, ampak tudi najbolj strogo zahtevo za združljivost. Lorrain in White [155] sta trdila, da so posamezniki glede na vlogo ekvivalentni, če so v sorodu z istimi posamezniki.

Definicija 7.4 Naj bo $G = (V, E)$ graf in $r : V \to W$ dodelitev vlog. Pravimo, da je r *krepko strukturna*, če imajo ekvivalentna vozlišča iste (izhodne in vhodne) soseščine, tj. če za vse $u, v \in V$ velja

$$r(u) = r(v) \implies N^+(u) = N^+(v) \; in \; N^-(u) = N^-(v).$$

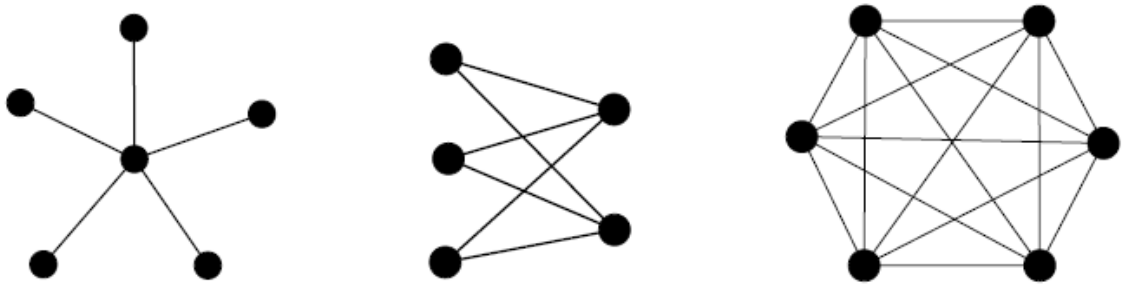
Spomnimo se opombe 7.2: definicije za dodelitve vlog se prevedejo na ustrezne particije in ekvivalenčne relacije.

Opomba 7.5 Po definiciji 7.3 za vsako dodelitev vlog r velja, da če je (u, v) povezava v grafu, je potem $(r(u), r(v))$ povezava v grafu vlog. Če pa je r krepko strukturna, velja tudi obrat (glej [157]). Ta pogoj je celo ekvivalenten temu, da je dodelitev vlog krepko strukturna. Tj. dodelitev vlog r je krepko strukturna natanko tedaj, ko za vse $u, v \in V$ velja, da je $(r(u), r(v))$ povezava v grafu vlog natanko tedaj, ko je (u, v) povezava v grafu.

Predstavili bomo nekaj primerov krepko strukturnih ekvivalenc. Identična preslikava $id : V \to V,\ v \mapsto v$, je krepko strukturna za vsak graf $G = (V, E)$, neodvisno od E. Nekateri nekoliko manj trivialni primeri so prikazani na sliki 7.1. Za zvezdo je dodelitev vlog, ki preslika središčno vozlišče v eno vlogo in vsa ostala vozlišča v drugo, krepko strukturna. Particija polnega dvodelnega grafa je krepko strukturna. Poln graf brez zank razen identitete nima nobene druge krepko strukturne dodelitve vlog, saj je soseščina vsakega vozlišča v edina množica, ki ne vsebuje v.

Zapišimo nekaj osnovih lastnosti. Razred krepko strukturno ekvivalentnih vozlišč v grafu je bodisi neodvisna množica (inducira graf brez povezav) ali pa klika z vsemi zankami. Torej, če sta dve sosednji vozlišči u, v krepko strukturno ekvivalentni, potem sta tako (u, v) kot (v, u) povezavi v grafu in obe vozlišči imata zanko.

Neusmerjena razdalja dveh strukturno ekvivalentnih neizoliranih vozlišč je največ 2. Če sta u in v strukturno ekvivalentni in ima u sosedo w, potem je w tudi soseda v. Tako lahko strukturna ekvivalenca identificira samo tista vozlišča, ki so si blizu.



Slika 7.1: Zvezda (levo), poln dvodelen graf (sredina) in poln graf (desno)

Čeprav v večini nepravilnih grafov ni nobene netrivialne strukrurne ekvivalence, je lahko množica strukturnih ekvivalenc velika. Za polne grafe z zankami je vsaka ekvivalenca strukturna. V poglavju 7.2.2 bomo raziskali delno urejenost na tej množici.

Variacije strukturnih ekvivalenc. Zahtevo, da morajo krepko strukturno ekvivalentna sosednja vozlišča imeti zanke, so nekateri avtorji opustili.

Definicija 7.6 Ekvivalenco $\sim$ na množici vozlišč grafa imenujemo *strukturna ekvivalenca*, če je za vsa vozlišča $u \sim v$ transpozicija vozlišč u in v avtomorfizem grafa.

White in Reitz [157] sta podala nekoliko drugačno definicijo, ki z definicijo 7.6 sovpada na grafih brez zank.

7.2.1 Mreža ekvivalenčnih relacij

Množica ekvivalenčnih relacij na množici V je zelo velika. Tu bomo pokazali, da ta množica naravno dopušča delno urejenost in izkaže se, da je mreža. Ekvivalenčne relacije na množici V so podmnožice $V \times V$, torej se jih lahko delno uredi z inkluzijo ($\sim_1 \leq \sim_2$ natanko tedaj, ko $\sim_1 \subseteq \sim_2$). Pravimo, da je $\sim_1$ *finejša* kot $\sim_2$ in $\sim_2$ je *šibkejša* (bolj groba) kot $\sim_1$. Ta delna urejenost za ekvivalence se prevede na ustrezne particije in dodelitve vlog (glej opombo 7.2).

V delno urejenih množicah dva elementa nista nujno primerljiva. V nekaterih primerih pa lahko zagotovimo vsaj obstoj zgornjih in spodnjih mej.

Definicija 7.7 Naj bo X množica, ki je delno urejena z $\leq$ in naj bo $Y \subseteq X$. $y^* \in X$ je *zgornja meja* (ali *spodnja meja*) za Y natanko tedaj, ko je za vsak $y \in Y, y \leq y^*$ ($y^* \leq y$).

$y^* \in X$ je *supremum* (*infimum*) za Y, če je zgornja meja (spodnja meja) in za vsak $y' \in X$, ki je zgornja meja (spodnja meja) za Y, sledi $y^* \leq y'$ ($y' \leq y^*$). Drugi pogoj nam zagotovi, da sta supremum in infimum (če obstajata) enolična.

Supremum za Y označimo s *sup(Y)*, infimum pa z *inf(Y)*. Namesto *sup($\{x, y\}$)* in *inf($\{x, y\}$)* pišemo tudi *sup(x, y)* in *inf(x, y)*.

Mreža je delno urejena množica L, za katero za vse $a, b \in L$ obstajata *sup(a, b)* in *inf(a, b)*. Običajno *sup(a, b)* imenujemo tudi *spoj* a in b, ki ga označimo z $a \vee b$, *inf(a, b)* pa imenujemo tudi *stik* a in b ter ga označimo z $a \wedge b$.

Če sta $\sim_1$ in $\sim_2$ ekvivalenčni relaciji na V, je infimum $\sim_1$ in $\sim_2$ njun presek (kot množici). Supremum pa je nekoliko bolj kompliciran. Vsebovati mora vse pare vozlišč, ki

so ekvivalentna bodisi v $\sim_1$ bodisi v $\sim_2$, in tudi vozlišča, ki so s takimi pari povezana z verigo: *tranzitivno zaprtje* relacije $R \subseteq V \times V$ definiramo kot relacijo $S \subseteq V \times V$, kjer za vse $u, v \in V$ velja

$$uSv \Leftrightarrow \exists k \in \mathbb{N}, \exists w_1, \ldots, w_k \in V, \text{ tako da } u = w_1, v = w_k \text{ in } \forall i = 1, \ldots, k-1 \text{ je } w_i R w_{i+1}.$$

Tranzitivno zaprtje simetrične relacije je simetrično, tranzitivno zaprtje refleksivne relacije je refleksivno, tranzitivno zaprtje katerekoli relacije je tranzitivo.

Če sta $\sim_1$ in $\sim_2$ ekvivalenčni relaciji na V, sledi da je tranzitivno zaprtje njune unije supremum za $\sim_1$ in $\sim_2$.

To povzamemo v naslednjem izreku.

Izrek 7.8 *Množica ekvivalenčnih relacij je mreža.*

Interpretacija v našem kontekstu je naslednja: za podani ekvivalenčni relaciji, ki identificirata vozlišči z istima vlogama, obstaja enolično določena najmanjša ekvivalenca, ki identificira vozlišči z istima vlogama v katerikoli od originalnih ekvivalenc. Še več, obstaja enolično definirana največja ekvivalenca, ki razlikuje med tistimi igralci, ki igrajo različno vlogo v katerikoli od originalnih ekvivalenc.

7.2.2 Mreža strukturnih ekvivalenc

Če sta $\sim_1$ in $\sim_2$ krepko strukturni ekvivalenci v grafu, potem je lahko preveriti, da sta to tudi njun presek in tranzitivno zaprtje njune unije.

Trditev 7.9 *Množica krepko strukturnih ekvivalenc grafa je podmreža mreže vseh ekvivalenčnih relacij.*

V posebnem, v grafu vedno obstaja maksimalna strukturna ekvivalenca (MSE).

Lastnost 'biti krepko strukturen' se pri rafinaciji ohrani:

Trditev 7.10 *Če je $\sim_1 \leq \sim_2$ in je $\sim_2$ krepko strukturna ekvivalenca, potem je to tudi $\sim_1$.*

Čeprav je zgornjo trditev zelo enostavno dokazati, je zelo uporabna, ker iz nje sledi, da je množica vseh strukturnih ekvivalenc v grafu popolnoma opisana z MSE. V naslednjem poglavju bomo predstavili algoritem za izračun MSE grafa, ki ima linearno časovno zahtevnost.

7.2.3 Izračun strukturnih ekvivalenc

Izračun maksimalne krepko strukturne ekvivalence grafa $G = (V, E)$ je precej preprost. Vsako vozlišče $v \in V$ razdeli V na 4 razrede (kakšen je lahko tudi prazen): vozlišča, ki so v $N^+(v)$, v $N^-(v)$, v obeh ali v nobeni.

Osnovna ideja sledečega algoritma 20 je izračun preseka vseh teh particij, tako da vsako povezavo gledamo največ dvakrat. Ta algoritem je prilagoditev algoritma Paiga in Tarjana [158] (glej razdelek 7.3.3) za izračun regularne notranjosti, za precej enostavnejši problem računanja MSE.

Pravilnost algoritma 20 sledi iz dejstva, da loči natanko pare vozlišč z neidentičnimi soseščinami.

Učinkovita implementacija zahteva strukturo podatkov, ki bo predstavljena v detajlih, saj je dobra vaja za razumevanje veliko bolj kompliciranega algoritma v razdelku 7.3.3.

Algorithm 13 Izračun maksimalne krepko strukturne ekvivalence (MSE) grafa

Vhod: graf $G = (V, E)$
Izhod: MSE grafa G

```
hrani particijo P = {C_1, ..., C_k} množice V, ki je na začetku cela particija P = {V}
// na koncu bo P MSE za G
for v ∈ V do
    for razred C, kateremu pripada vozlišče u ∈ N^+(v), do
        naredi nov razred C' v P
        premakni vsa vozlišča iz N^+(v) ∩ C iz C v C'
        if C postane prazen then
            odstrani C iz P
        end if
    end for
    for razred C, kateremu pripada vozlišče u ∈ N^-(v), do
        naredi nov razred C' za P
        premakni vsa vozlišča iz N^-(v) ∩ C iz C v C'
        if C postane prazen then
            odstrani C iz P
        end if
    end for
end for
```

- Graf $G = (V, E)$ mora dopuščati dostop do (izhodno/vhodno) incidenčnega seznama vozlišč v v času, sorazmernem z velikostjo seznama.

- Pregled vseh elementov seznama mora biti mogoč v linearnem času.

- Povezava mora dopuščati dostop do njenega vira in njenega cilja v konstantnem času.

- Particija mora dovoliti vstavljanje in izbris razredov v konstantnem času.

- Razred mora dovoliti vstavljanje in izbris vozlišč v konstantnem času.

- Vozlišče mora dovoliti dostop do njegovega razreda v konstantnem času.

Zahteve za particije in razrede so izpolnjene, če je particija predstavljena z dvojno povezanim seznamom njenih razredov in razred z dvojno povezanim seznamom njegovih vozlišč.

Zunanja zanka za dano točko v poteka takole:

1. Preglej vse izhodne povezave vozlišča v. Za vsako tako povezavo (v, u) določi razred C vozlišča u in ustvari ustrezen blok C', če še ne obstaja. Prestavi u iz C v C'.

2. Med pregledovanjem ustvari seznam tistih razredov C, ki so ločeni. Po pregledu obdelaj seznam ločenih razredov. Za vsak tak razred C označi C' tako, da ni več povezan s C, in izloči C, če je sedaj prazen.

3. Preglej vse vhodne povezave vozlišča v in naredi isti postopek kot zgoraj.

Zanka za dano vozlišče v potrebuje čas, sorazmeren z njegovo stopnjo, če v ni izolirano, in konstanten čas sicer. Sledi, da je skupna časovna zahtevnost $\mathcal{O}(|V| + |E|)$, kar pa je tudi asimptotska meja za prostorsko zahtevnost.

Zaključek. Strukturna ekvivalenca je teoretično in računsko zelo enostavna. Je veliko prestroga, da bi jo lahko aplicirali na nepravilna omrežja in samo na vozlišča na razdalji največ 2, ki se dajo identificirati s strukturno ekvivalenco. Kljub temu pa je strukturna ekvivalenca izhodiščna točka za mnogo omilitev.

7.3 Regularna ekvivalenca

Regularna ekvivalenca se vrača k ideji Sailerja [156] o *strukturni sorodnosti*. Trdil je, da igralci igrajo isto vlogo, če so povezani z vlogovno ekvivalentnimi igralci - v nasprotju s strukturno ekvivalenco, kjer morajo biti povezani z identičnimi igralci. Regularno ekvivalenco sta prva definirala White in Reitz v [157]. Borgatti in Everett (glej npr. [159]) sta ekvivalentno definicijo podala v smislu barvanj (tukaj imenovanih dodelitve vlog). Barvanje je regularno, če imajo vozlišča, ki so enako obarvana, soseščine enakih barv. Če je $r : V \to W$ dodelitev vlog in $U \subseteq V$, potem je $r(U) = \{r(u); u \in U\}$ njena *množica vlog*.

Definicija 7.11 Dodelitev vlog $r : V \to W$ je *regularna*, če za vse $u, v \in V$ velja

$$r(u) = r(v) \implies r(N^+(u)) = r(N^+(v)) \text{ in } r(N^-(u)) = r(N^-(v)).$$

Enačbi na desni sta enačbi množic. Regularne dodelitve vlog se pogosto smatra kot razred dodelitev vlog. Izraz *regularen* se v literaturi pogosto opušča.

Regularna ekvivalenca in bisimulacija. Marx in Masuch [160] sta izpostavila tesno povezavo med regularno ekivivalenco, bisimulacijo in dinamično logiko. Za uspešno iskanje dobrih algoritmov za regularne ekvivalence je treba pregledati nekaj literature o bisimulaciji.

7.3.1 Osnovne lastnosti

V tem poglavju bomo zapisali nekaj lastnosti regularnih ekvivalenčnih relacij.

Identična preslikava id: $V \to V$, $v \mapsto v$, je regularna za vse grafe. Splošneje, vsaka strukturna dodelitev vlog je regularna.

Naslednja trditev pove, kdaj je particija, ki je inducirana s konstantno dodelitvijo vlog $J : V \to 1$ (polna particija), regularna. *Ponor* je vozlišče z ničelno izhodno stopnjo, *izvor* pa vozlišče z ničelno vhodno stopnjo.

Trditev 7.12 *Polna particija grafa $G = (V, E)$ je regularna natanko tedaj, ko G ne vsebuje niti ponorov niti izvorov ali pa je $E = \emptyset$.*

Dokaz. ($\Rightarrow$): Če je $E = \emptyset$, potem je desna stran v definiciji 7.11 $\emptyset = \emptyset$, torej je vsaka dodelitev vlog regularna. Če G nima niti ponorov niti izvorov, potem je za vsak $v \in V$ $J(N^+(v)) = J(N^-(v)) = \{1\}$ in obe enačbi v definiciji 7.11 sta izpolnjeni za vse $u, v \in V$.

($\Leftarrow$): Denimo, da $E \neq \emptyset$ in naj bo v ponor. Ker $E \neq \emptyset$, obstaja $u \in V$ z neničelno izhodno stopnjo. Potem je

$$J(N^+(v)) = \emptyset \neq \{1\} = J(N^+(u)),$$

toda $J(u) = 1 = J(v)$, torej J ni regularna. Primer, ko G vsebuje izvor, obravnavamo analogno. □

□

Identiteto in polno particijo imenujemo *trivialni* dodelitvi vlog. Naslednja lema je formulirana v [161] za neusmerjene grafe, ima pa tudi posplošitev za krepko povezane (usmerjene) grafe.

Lema 7.13 *Naj bo G krepko povezan graf. Potem za katerokoli netrivialno dodelitev vlog r grafa G in za vsako vozlišče v ne drži niti $\{r(v)\} = r(N^+(v))$ niti $\{r(v)\} = r(N^-(v))$.*

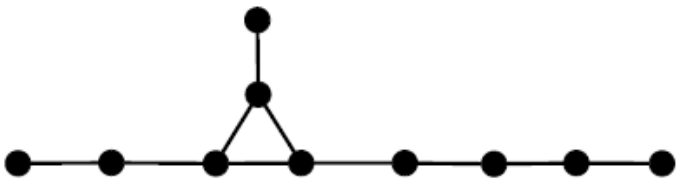
Dokaz. Če za neko vozlišče v velja $\{r(v)\} = r(N^+(v))$, potem bi moralo isto veljati za vsako vozlišče iz $N^+(v)$. Zato bo vsakemu vozlišču v zaporedju izhodnih soseščin dodeljena ista vloga, in ker je G krepko povezan, sledi da je $r(V) = \{r(v)\}$, kar pa je v nasprotju z dejstvom, da je dodelitev vlog netrivialna. Primer $\{r(v)\} = r(N^-(v))$ za neko vozlišče v obravnavamo enako. □

□

Graf z vsaj tremi vozlišči, čigar edine dodelitve vlog so trivialne, imenujemo *vlogovno primitiven*. Obstoj usmerjenih vlogovno primitivnih grafov je trivialen: za vsako usmerjeno pot je edino identična particija regularna. Usmerjeni grafi, ki imajo kot regularne particije natanko identiteto in polno particijo, so naprimer usmerjeni cikli praštevilske dolžine, saj vsaka netrivialna regularna ekvivalenca inducira netrivialni delitelj dolžine cikla.

Obstoj neusmerjenih vlogovno primitivnih grafov pa ni trivialen.

Izrek 7.14 *Graf na sliki 7.2 je vlogovno primitiven.*



Slika 7.2: Neusmerjen vlogovno primitiven graf

Dokaz gre s preverjanjem, da so vse možne dodelitve vlog bodisi neregularne bodisi trivialne, pri čemer lahko uporabimo dejstvo, da edine možne poti v grafu na sliki 7.2 v veliki meri zmanjšujejo možne situacije, ki jih je treba upoštevati. Dokaz bomo izpustili.

Graf, v katerem je vsaka dodelitev vlog regularna, se imenuje *poljubno vlogovno določljiv*. Naslednja lema je formulirana v [161] za neusmerjene povezane grafe.

Lema 7.15 *Krepko povezan graf $G = (V, E)$ je poljubno vlogovno določljiv natanko tedaj, ko je poln, možno z nekaj, toda ne nujno vsemi zankami.*

Dokaz. ($\Leftarrow$): Naj bo $G = (V, E)$ graf, ki zadošča pogojem leme, in naj bo r dodelitev vlog. Pokazati moramo, da za vsa vozlišča $u, v \in V$ velja

$$r(u) = r(v) \implies r(N^+(u)) = r(N^+(v)) \text{ in } r(N^-(u)) = r(N^-(v)).$$

Če je $u = v$, je to trivialno. V nasprotnem primeru sta u in v povezani z neusmerjeno povezavo, tj. množice vlog njunih vhodnih in izhodnih soseščin vsebujejo $r(u)$. Te množice vlog vsebujejo tudi vse ostale vloge, saj sta u in v povezani z vsemi ostalimi vozlišči. Tako množice vlog vhodnih in izhodnih soseščin obeh vozlišč vsebujejo vse vloge, od koder sledi, da sta enaki.

($\Rightarrow$): Naj bo $G = (V, E)$ graf z vozliščema u in v, tako da $u \neq v$ in $(u, v) \notin E$. Eno vlogo dodelimo $V \backslash \{v\}$, drugo pa v. To je netrivialna dodelitev vlog ($n \geq 2$, ker je G povezan) z $r(u) = r(N^+(u))$. Tako po lemi 7.13 ta dodelitev vlog ne more biti regularna.□

□

7.3.2 Struktura mreže in regularna notranjost

Videli smo, da je lahko množica regularnih ekvivalenčnih relacij grafa ogromna. V tem razdelku bomo dokazali, da je to mreža.

Izrek 7.16 *Množica vseh regularnih ekvivalenčnih relacij grafa G tvori mrežo, kjer je supremum zožitev supremuma mreže vseh ekvivalenčnih relacij.*

Pred dokazom pa si oglejmo še naslednjo lemo.

Lema 7.17 *Naj bo $(X, \leq)$ delno urejena množica. Če $\sup H$ obstaja za poljubno podmnožico $H \subseteq X$, potem je $(X, \leq)$ mreža.*

Dokaz. Vse kar moramo pokazati je, da za $x, y \in X$ obstaja $\inf(x, y)$. Naj bo $H := \{z \in X; z \leq x \text{ in } z \leq y\}$. Potem takoj vidimo, da je $\sup H$ infimum $\{x, y\}$. □

□

Sedaj lahko dokažemo še izrek 7.16.

Dokaz. (izreka 7.16) Po lemi 7.17 zadostuje dokazati obstoj supremuma poljubnih podmnožic. Identična particija je najmanjši element v množici regularnih ekvivalenčnih relacij, torej je supremum prazne množice. Zato moramo obravnavati le supremume nepraznih naborov regularnih dodelitev vlog. Ker je množica vseh ekvivalenčnih relacij grafa končna, zadostuje pokazati le obstoj supremuma dveh regularnih ekvivalenčnih relacij.

Naj bosta $\sim_1$ in $\sim_2$ regularni ekvivalenčni relaciji na grafu G. Naj bo tranzitivno zaprtje unije $\sim_1$ in $\sim_2$ definirano z $\equiv$.

Kot smo že prej omenili, $\equiv$ predstavlja supremum $\sim_1$ in $\sim_2$ v mreži vseh ekvivalenčnih relacij, torej je ekvivalenčna relacija in predstavlja supremum $\sim_1$ in $\sim_2$ glede na delno ureditev (ta je enaka v mreži vseh ekvivalenčnih relacij in v mreži regularnih ekvivalenčnih relacij). Potrebno je še pokazati, da je $\equiv$ regularna.

Denimo, da $u \equiv v$ in naj bo $x \in N^+(u)$ za $u, v, x \in V$. Ker $u \equiv v$, obstaja zaporedje $u, w_2, \ldots, w_{k-1}, v \in V$ kjer $u \sim_{j_1} w_2$, $j_1 \in \{1, 2\}$. Ker je $\sim_{j_1}$ regularna in $x \in N^+(u)$, obstaja $x_2 \in V$ tako, da $x_2 \in N^+(w_2)$ in $x_2 \sim_{j_1} x$. Ta postopek iterativno ponavljamo in na koncu dobimo x_k, da je $x_k \in N^+(v)$ in $x \equiv x_k$, kar zadošča pogoju za izhodno soseščino. Analogno obravnavamo primer, ko je $x \in N^-(u)$. □

□

Posledica 7.18 *Če je G graf, potem obstajata maksimalna in minimalna regularna ekvivalenčna relacija za G.*

Dokaz. Maksimalna regularna ekvivalenčna relacija je enostavno supremum vseh regularih ekvivalenčnih relacij, minimalna pa je infimum vseh regularnih ekvivalenčnih relacij. Še lažje: minimalna regularna ekvivalenčna relacija je identična particija, ta je vedno regularna in minimalna. □

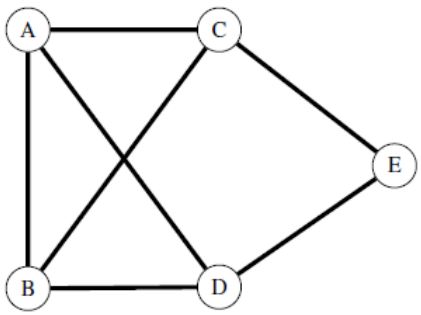
□

Čeprav je supremum mreže regularnih ekvivalenčnih relacij zožitev supremuma mreže vseh ekvivalenčnih relacij, pri infimumu ni tako.

Trditev 7.19 *Mreža regularnih ekvivalenčnih relacij ni podmreža mreže vseh ekvivalenčnih relacij.*

Dokaz. Pokazali bomo, da infimum ni zožitev infimuma mreže vseh ekvivalenčnih relacij (ki je presek). Poglejmo si graf na sliki 7.3 z dvema regularnima particijama $\mathcal{P}_1 := \{\{A, C, E\}, \{B, D\}\}$ in $\mathcal{P}_2 := \{\{A, C\}, \{B, D, E\}\}$. Presek $\mathcal{P}_1$ in $\mathcal{P}_2$ je $\mathcal{P} = \{\{A, C\}, \{B, D\}, \{E\}\}$, ki ni regularen. □

□



Slika 7.3: Infimum ni enak preseku

Iz dejstva, da je supremum mreže regularnih ekvivalenčnih relacij zožitev supremuma mreže vseh ekvivalenčnih relacij, sledi obstoj maksimalne regularne ekvivalenčne relacije, ki leži pod poljubno dano ekvivalenčno relacijo.

Definicija 7.20 Naj bo G graf in $\sim$ ekvivalenčna relacija na množici vozlišč grafa G. Ekvivalenčno relacijo $\sim_1$ imenujemo *regularna notranjost* $\sim$, če zadošča naslednjim pogojem:

1. $\sim_1$ je regularna,
2. $\sim_1 \leq \sim$, in
3. za vse $\sim_2$, ki izpolnjujejo zgornja pogoja, velja $\sim_2 \leq \sim_1$.

Posledica 7.21 *Naj bo G graf in $\sim$ ekvivalenčna relacija na množici vozlišč grafa G. Potem regularna notranjost $\sim$ obstaja.*

Po drugi strani, v splošnem ne obstaja minimalna regularna ekvivalenčna relacija nad dano ekvivalenčno relacijo (temu bi rekli regularno zaprtje *ali* regularna ogrinjača*).*

Dokaz. Za prvi del naj bo $G = (V, E)$ graf in $\sim$ poljubna ekvivalenčna relacija na možici vozlišč. Potem je supremum množice vseh tistih regularnih ekvivalenčnih relacij, ki so finejše kot $\sim$, regularna notranjost $\sim$.

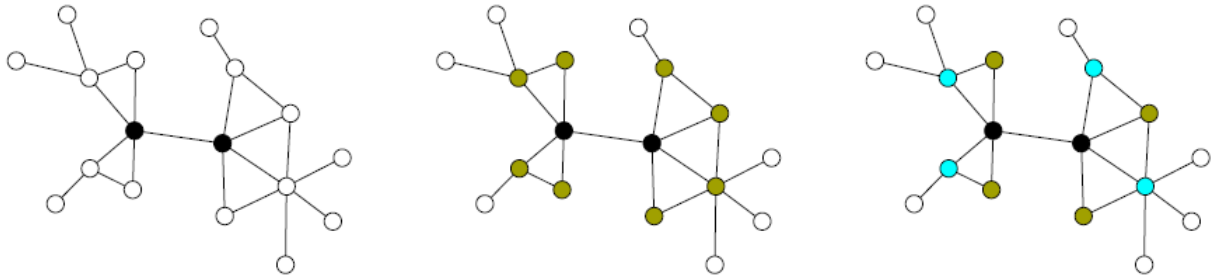
Za drugi del se spomnimo primera iz dokaza trditve 7.19 (na sliki 7.3). Lahko je videti, da sta regularni particiji $\mathcal{P}_1 := \{\{A, C, E\}, \{B, D\}\}$ in $\mathcal{P}_2 := \{\{A, C\}, \{B, D, E\}\}$ nad neregularno particijo $\mathcal{P} = \{\{A, C\}, \{B, D\}, \{E\}\}$ in da sta najmanjši taki. □

□

Infimum (v mreži regularnih ekvivalenčnih relacij) dveh regularnih ekvivalenčnih relacij $\sim_1$ in $\sim_2$ je podan z regularno notranjostjo preseka $\sim_1$ in $\sim_2$.

7.3.3 Izračun regularne notranjosti

Regularna notranjost ekvivalenčne relacije $\sim$ je najbolj groba regularna rafinacija $\sim$. Lahko jo izračunamo na naslednji način: začnemo z $\sim$, potem delamo korake rafinacije tako, da množico trenutno ekvivalentnih vozlišč z neekvivalentnimi soseščinami razbijemo. To ponavljamo, dokler ekvivalentna vozlišča nimajo ekvivalentnih soseščin. Za primer takšnega izračuna glej sliko 7.4. Časovna zahtevnost takšnega izračuna je močno odvisna od tega, kako so ti koraki rafinacije razvrščeni.



Slika 7.4: Izračun regularne notranjosti: začetna particija (levo), prvi korak (na sredini) in drugi oz. zadnji korak (desno)

V tem razdelku bomo predstavili dva algoritma za izračun regularne notranjosti. CATREGE [164] je najbolj znan algoritem v teoriji socialnih omrežij. Njegova časovna zahtevnost je $\mathcal{O}(n^3)$. Tarjan in Paige [158] sta predstavila sofisticiran algoritem za *problem relacijsko najšibkejše particije*, ki je v osnovi ekvivalenten izračunu regularne notranjosti, njegova časovna zahtevnost pa je $\mathcal{O}(m \log n)$.

CATREGE. Algoritem je bil sestavljen za izračun maksimalne regularne ekvivalenčne

relacije grafa oz. v splošnem za izračun regularne notranjosti ekvivalenčne relacije. V grobem algoritem izvaja naslednje korake:

- CATREGE v vsakem koraku rafinacije hrani trenutno particijo $\mathcal{P}$, ki je na začetku enaka polni particiji (alternativno lahko kot vhodni podatek podamo poljubno particijo).

- V vsakem koraku rafinacije za vsak par ekvivalentnih vozlišč (glede na $\mathcal{P}$) preveri, ali so njune soseščine ekvivalentne (glede na $\mathcal{P}$). Če so, potem vozlišči ostaneta ekvivalentni, drugače pa po tem koraku rafinacije nista več ekvivalentni.

- Algoritem se konča, če se ne zgodi nobena sprememba.

Število korakov rafinacije je omejeno z n, ker se v vsakem koraku (razen v zadnjem) število ekvivalenčnih razredov poveča za najmanj ena. Časovna zahtevnost enega koraka rafinacije je $\mathcal{O}(n^2)$.

Problem relacijsko najšibkejše particije (PRNP). Kot vhodna podatka problem dobi (usmerjen) graf $G = (V, E)$ in particijo $\mathcal{P}$ množice vozlišč V.

Za podmnožico $S \subseteq V$ pišemo $E(S) := \{v \in V; \exists u \in S, \text{ tako da } uEv\}$ in $E^{-1}(S) := \{u \in V; \exists v \in S, \text{ tako da } uEv\}$. Naj bosta $B \subseteq V$ in $S \subseteq V$. Pravimo, da je B *stabilna* glede na S, če velja $B \subseteq E^{-1}(S)$ ali $B \cap E^{-1}(S) = \emptyset$. Naj bo $\mathcal{P}$ particija V. $\mathcal{P}$ je *stabilna* glede na S, če so vsi njeni bloki stabilni glede na S. Pravimo, da je $\mathcal{P}$ *stabilna*, če je stabilna glede na vsak svoj blok.

Iskanje najbolj grobe stabilne rafinacije za začetno particijo $\mathcal{P}$ je problem PRNP.

V jeziku dodelitev vlog to pomeni, da za vsaki dve vlogi, recimo r_1 in r_2, velja, da imajo vsa vozlišča, dodeljena r_1, izhodno povezavo z vozliščem, ki je dodeljeno r_2, ali pa nobeno.

Algoritem za ta problem ima časovno zahtevnost $\mathcal{O}(m \log n)$ in prostorsko zahtevnost $\mathcal{O}(m + n)$. Za redke grafe deluje precej bolje kot CATREGE.

Funkcija Razbitje. Algoritem uporablja primitivno rafinacijsko operacijo. Za vsako particijo $\mathcal{Q}$ množice V in podmnožico $S \subseteq V$ naj bo Razbitje$(S, \mathcal{Q})$ rafinacija $\mathcal{Q}$, dobljena tako, da je vsak blok B particije $\mathcal{Q}$, za katerega je $B \cap E^{-1}(S) \neq \emptyset$ in $B \setminus E^{-1}(S) \neq \emptyset$, zamenjan z dvema blokoma $B' := B \cap E^{-1}(S)$ in $B'' := B \setminus E^{-1}(S)$. S imenujemo *osnova razbitja* particije $\mathcal{Q}$, če Razbitje$(S, \mathcal{Q}) \neq \mathcal{Q}$. Opazimo, da je $\mathcal{Q}$ nestabilna glede na S, če in samo če je S osnova razbitja particije $\mathcal{Q}$.

Naj bosta S in Q podmnožici V ter $\mathcal{P}$ in $\mathcal{R}$ particiji V. Opazimo naslednje lastnosti in posledice stabilnosti funkcije Razbitje:

1. Rafinacija *podeduje* stabilnost, tj. če je $\mathcal{R}$ rafinacija $\mathcal{P}$ in $\mathcal{P}$ stabilna glede na S, potem je tudi $\mathcal{R}$ stabilna glede na S.

2. Stabilnost se *podeduje* pri uniji, tj. če je particija stabilna glede na dve množici, potem je stabilna tudi glede na njuno unijo.

3. Funkcija Razbitje je *monotona* v drugem argumentu, tj. če je $\mathcal{P}$ rafinacija $\mathcal{R}$, potem je Razbitje$(S, \mathcal{P})$ rafinacija Razbitje$(S, \mathcal{R})$.

4. Funkcija RAZBITJE je *komutativna* v smislu, da za najbolj grobo rafinacijo particije $\mathcal{P}$, ki je stabilna glede na S in Q, velja

$$\text{RAZBITJE}(S, \text{RAZBITJE}(Q, \mathcal{P})) = \text{RAZBITJE}(Q, \text{RAZBITJE}(S, \mathcal{P})).$$

Osnoven algoritem. Za začetek opišimo čisto osnoven algoritem za problem. Algoritem hrani particijo $\mathcal{Q}$, ki je na začetku enaka $\mathcal{P}$, in jo rafinira, dokler ne dobimo najbolj grobe stabilne rafinacije. Algoritem ponavlja naslednji korak, dokler $\mathcal{Q}$ ni stabilna:
RAFINIRAJ: Poišči množico S, ki je unija nekaj blokov $\mathcal{Q}$ in osnova razbitja particije $\mathcal{Q}$; zamenjaj $\mathcal{Q}$ z RAZBITJE$(S, \mathcal{Q})$.

Nekaj opazk. Ker rafinacija podeduje stabilnost, lahko dano množico S uporabimo za osnovo razbitja le enkrat v algoritmu. Ker unija osnov razbitja podeduje stabilnost, potem unije že uporabljenih osnov razbitja ne moremo uporabiti za osnovo razbitja. Posebej, stabilna particija je stabilna glede na unijo poljubnih podmnožic njenih blokov.

Lema 7.22 *Za algoritem vedno velja, da je poljubna stabilna rafinacija particije $\mathcal{P}$ tudi rafinacija trenutne particije $\mathcal{Q}$.*

Dokaz. Z indukcijo na število korakov rafinacije. Na začetku lema drži po definiciji. Recimo, da drži za vse korake pred korakom, kjer bomo pri rafinaciji particije $\mathcal{Q}$ uporabili osnovo razbitja S. Naj bo $\mathcal{R}$ poljubna stabilna rafinacija $\mathcal{P}$. Ker je S unija blokov $\mathcal{Q}$ in po indukcijski predpostavki $\mathcal{R}$ rafinacija $\mathcal{Q}$, je S unija blokov $\mathcal{R}$. Zato je $\mathcal{R}$ stabilna glede na S. Ker je RAZBITJE monotona, velja da je $\mathcal{R}$=RAZBITJE$(S, \mathcal{R})$ rafinacija RAZBITJE$(S, \mathcal{Q})$. □

□

Izrek 7.23 *Rafinacijski algoritem je pravilen in se konča v največ $n - 1$ korakih. Vrne nam enolično določeno najbolj grobo stabilno rafinacijo.*

Dokaz. Trditev o številu korakov sledi iz dejstva, da je število blokov med 1 in n. Ko ni več mogoče narediti naslednjega koraka rafinacije, je $\mathcal{Q}$ stabilna in po lemi 7.22 je poljubna stabilna rafinacija rafinacija particije $\mathcal{Q}$. Sledi, da je $\mathcal{Q}$ enolično določena najbolj groba rafinacija. □

□

Za rešitev problema ne potrebujemo tako splošnega algoritma: ne potrebujemo osnov razbitja, ki so unije blokov. Zadostuje se omejiti le na bloke $\mathcal{Q}$ kot osnove razbitja. Vendar pa je ideja uporabe unij blokov za osnove razbitja odločilna za razvoj hitrejšega algoritma.

Predhodna obdelava. V učinkoviti implementaciji algoritma je uporabno problem skrčiti na takšnega, kjer je $|E(\{v\})| \geq 1$ za vse $v \in V$ (omejimo se na vozlišča z izhodnimi povezavami). Da pridemo do tega, predhodno obdelamo particijo $\mathcal{P}$, tako da vsak blok B razbijemo na $B' := B \cap E^{-1}(V)$ in $B'' := B \setminus E^{-1}(V)$. Blokov oblike B'' rafinacijski algoritem ne bo nikoli razbil, zato lahko algoritem uporabimo le na particiji $\mathcal{P}'$, ki sestoji iz množice blokov oblike B'. $\mathcal{P}'$ je particija množice $V' := E^{-1}(V)$ in ima moč največ m. Najbolj groba stabilna rafinacija particije $\mathcal{P}'$, skupaj s bloki B'', je najbolj groba stabilna rafinacija $\mathcal{P}$. Časovna zahtevnost predhodne in končne obdelave je $\mathcal{O}(m+n)$, če je množica

praslik $E^{-1}(v)$ za vsak $v \in V$ na voljo. Nadalje predpostavimo, da $|E(\{v\})| \geq 1$ za vse $v \in V$. Sledi, da je $m \geq n$.

Časovna zahtevnost osnovnega algoritma. Rafinacijski algoritem lahko implementiramo tako, da bo imel časovno zahtevnost $\mathcal{O}(mn)$. To storimo tako, da za vsak element $v \in V$ shranimo prasliko $E^{-1}(v)$. Iskanje bloka particije $\mathcal{Q}$, ki je osnova razbitja $\mathcal{Q}$, in izvajanje ustreznih razbitij ima časovno zahtevnost $\mathcal{O}(m)$. Časovna zahtevnost celotnega algoritma je največ $\mathcal{O}(mn)$.

Izboljšan algoritem. Če želimo hitrejši algoritem, potrebujemo dober način za iskanje osnov razbitja. Poleg trenutne particije $\mathcal{Q}$ dodatno hranimo še particijo $\mathcal{X}$. Veljati mora, da je $\mathcal{Q}$ rafinacija $\mathcal{X}$ in da je $\mathcal{Q}$ stabilna glede na vsak blok $\mathcal{X}$ (kasneje bomo temu rekli *relativna regularna ekvivalenčna relacija* glede na $\mathcal{X}$). Na začetku je $\mathcal{Q} = \mathcal{P}$ in $\mathcal{X}$ je polna particija (V je edini blok). Izboljšan algoritem ponavlja naslednji korak, dokler ni $\mathcal{Q} = \mathcal{X}$:
RAFINIRAJ: Poišči blok $S \in \mathcal{X}$, ki ni blok $\mathcal{Q}$. Poišči blok $B \in \mathcal{Q}$, tako da $B \subseteq S$ in $|B| \leq |S|/2$. Zamenjaj S znotraj $\mathcal{X}$ z množicama B in $S \setminus B$. Zamenjaj $\mathcal{Q}$ z RAZBITJE($S \setminus B$, RAZBITJE($B, \mathcal{Q}$)).
Pravilnost izboljšanega algoritma sledi iz pravilnosti originalnega algoritma in iz prej podanih dveh načinov kako lahko particija podeduje stabilnost glede na množico.

Poseben primer: E je funkcija. Preden pričnemo razpravljati o algoritmu v splošnem, obravnavajmo še poseben primer, ko je E funkcija, tj. $|E(\{v\})| = 1$ za vse $v \in V$. V tem primeru predpostavimo, da je $\mathcal{Q}$ particija, ki je stabilna glede na množico S, ki je unija nekaj blokov $\mathcal{Q}$, ter da je $B \subseteq S$ blok $\mathcal{Q}$. Potem je RAZBITJE($B, \mathcal{Q}$) stabilna tudi glede na $S \setminus B$. To drži, saj če je B_1 blok RAZBITJE($B, \mathcal{Q}$), potem iz $B_1 \subseteq E^{-1}(B)$ sledi $B_1 \cap E^{-1}(S \setminus B) = \emptyset$, in iz $B_1 \subseteq E^{-1}(S) \setminus E^{-1}(B)$ sledi $B_1 \subseteq E^{-1}(S \setminus B)$. Sledi da zadostuje v vsakem koraku rafinacije zamenjati $\mathcal{Q}$ z RAZBITJE($B, \mathcal{Q}$), saj je RAZBITJE($B, \mathcal{Q}$)=RAZBITJE($S \setminus B$, RAZBITJE($B, \mathcal{Q}$)). To je ideja Hopcroftovega algoritma »obdelaj manjšo polovico« za funkcijski problem najšibkejše particije. Rafinacijska množica B ima moč enako največ polovici moči stabilne množice S, v kateri je vsebovana.

Nazaj k splošnemu primeru. V bolj splošnem problemu relacijsko najšibkejše particije iz stabilnosti glede na S in B ne sledi stabilnost glede na $S \setminus B$. Torej ne moremo uporabiti Hopcroftovega algoritma. To je resen problem, saj si ne moremo privoščiti (v smislu časovne zahtevnosti), da bi morali pregledati množico $S \setminus B$ preden bi lahko izvedli en korak rafinacije. Vseeno pa lahko še vedno izkoristimo to idejo, da z metodo, ki eksplicitno pregleda samo B, rafiniramo glede na B in $S \setminus B$.

Lema 7.24 *Naj bo particija $\mathcal{Q}$ stabilna glede na množico S, ki je unija nekaj blokov particije $\mathcal{Q}$. Naj bo particija $\mathcal{Q}$ najprej rafinirana glede na blok $B \subseteq S$ in potem glede na $S \setminus B$. Potem veljajo naslednje trditve:*

1. *Rafiniranje $\mathcal{Q}$ glede na B razdeli blok $D \in \mathcal{Q}$ na bloka $D_1 = D \cap E^{-1}(B)$ in $D_2 = D - D_1$ če in samo če velja $D \cap E^{-1}(B) \neq \emptyset$ in $D \setminus E^{-1}(B) \neq \emptyset$.*
2. *Rafiniranje* RAZBITJE($B, \mathcal{Q}$) *glede na $S \setminus B$ razdeli D_1 na blok $D_{11} = D_1 \cap E^{-1}(S \setminus B)$ in blok $D_{12} = D_1 - D_{11}$ če in samo če velja $D_1 \cap E^{-1}(S \setminus B) \neq \emptyset$ in $D_1 \setminus E^{-1}(S \setminus B) \neq \emptyset$.*

3. *Rafiniranje* RAZBITJE$(B, \mathcal{Q})$ *glede na* $S \setminus B$ *ne razbije* D_2.

4. $D_{12} = D_1 \cap (E^{-1}(B) \setminus E^{-1}(S \setminus B))$.

Dokaz. Prva in druga točka sledita iz definicije RAZBITJE.

Točka 3: Iz točke 1 sledi, da če je D razbita, potem $D \cap E^{-1}(B) \neq \emptyset$. Ker je D stabilna glede na S in ker $B \subseteq S$, je $D_2 \subseteq D \subseteq E^{-1}(S)$. Ker je po točki 1 $D_2 \cap E^{-1}(B) = \emptyset$, sledi da je $D_2 \subseteq E^{-1}(S \setminus B)$.

Točka 4 sledi iz dejstva, da je $D_1 \subseteq E^{-1}(B)$ in $D_{12} = D_1 \setminus E^{-1}(S \setminus B)$. □

□

Izvajanje trojnega razbitja D na D_{11}, D_{12} in D_2, kot je opisano v lemi 7.24, je najtežji del algoritma. Točka 4 leme 7.24 je odločilna opazka, ki jo bomo uporabili v naši implementaciji. Spomnimo se, da pregled množice $S \setminus B$ vzame preveč časa, da bi dosegli podano časovno zahtevnost. Potrebovali bomo dodatno podatkovno strukturo, da bomo s pregledom le množice B določili $D_1 \setminus E^{-1}(S \setminus B) = (D \cap E^{-1}(B)) \setminus E^{-1}(S \setminus B)$.

Časovna zahtevnost izboljšanega algoritma. Dan element množice V je v največ $\log_2 n + 1$ različnih blokih B, ki jih uporabimo za rafinacijske množice, saj je vsaka naslednja takšna množica za vsaj pol manjša od prejšnje. Opisali bomo implementacijo algoritma, v kateri ima en korak rafinacije glede na blok B časovo zahtevnost $\mathcal{O}(|B| + \sum_{u \in B} |E^{-1}(\{u\})|)$. Iz tega s seštevanjem po vseh blokih B, ki smo jih uporabili za rafinacijo, in po vseh elementih takih blokov sledi, da bo časovna zahtevnost celotnega algoritma največ $\mathcal{O}(m \log n)$.

Podatkovne strukture. Graf $G = (V, E)$ je predstavljen z množicama V in E. Particiji $\mathcal{Q}$ in $\mathcal{X}$ sta predstavljeni z dvojno povezanim seznamom njunih blokov.

Pravimo, da je blok S particije $\mathcal{X}$ *enostaven*, če vsebuje le en blok particije $\mathcal{Q}$ (enak S vendar shranjen v drugem zapisu) in *sestavljen*, če vsebuje vsaj dva bloka particije $\mathcal{Q}$.

Različni zapisi podatkov so povezani med sabo na naslednje načine. Vsaka povezava uEv kaže na svoj izvor u. Vsako vozlišče v kaže na seznam vhodnih povezav uEv. To dopušča pregled množice $E^{-1}(\{v\})$ v času, sorazmernem z njeno velikostjo. Vsakemu bloku $\mathcal{Q}$ pripada število, ki pove njegovo velikost in vsak blok kaže na dvojno povezan seznam svojih vozlišč (dopušča brisanje v $\mathcal{O}(1)$). Vsako vozlišče kaže na blok particije $\mathcal{Q}$, v katerem je vsebovano. Vsak blok particije $\mathcal{X}$ kaže na dvojno povezan seznam blokov $\mathcal{Q}$, ki jih vsebuje. Vsak blok $\mathcal{Q}$ kaže na blok $\mathcal{X}$, v katerem je vsebovan. Hranimo tudi množico C sestavljenih blokov particije $\mathcal{X}$. Na začetku C vsebuje le blok V, ki je unija blokov particije $\mathcal{P}$. Če $\mathcal{P}$ vsebuje le en blok (po predhodni obdelavi), potem je $\mathcal{P}$ že najbolj groba stabilna rafinacija in končamo algoritem.

Da bo trojno razbitje (glej lemo 7.24) hitro, potrebujemo še en nabor podatkov. Za vsak blok S particije $\mathcal{X}$ in za vsak element $v \in E^{-1}(S)$ hranimo število PREŠTEJ$(v, S) := |S \cap E(\{v\})|$. Vsaka povezava uEv za $v \in S$ vsebuje kazalec na PREŠTEJ(u, S). Na začetku imamo eno preštetje na vozlišče (tj. PREŠTEJ$(v, V) = |E(\{v\})|$) in vsaka povezava uEv kaže na PREŠTEJ(u, V).

Prostorska in začetna časovna zahtevnost sta $\mathcal{O}(m)$.

Rafinacijski algoritem ponavlja korake rafinacije, dokler C ni prazna.

Izvedba enega koraka rafinacije. Za boljšo preglednost en korak rafinacije razdelimo na 7 podkorakov.

1. **(izberi rafinacijski blok).** Odstrani nek blok S iz C. (Blok S je sestavljen blok particije $\mathcal{X}$.) Preglej prva dva bloka na seznamu blokov particije $\mathcal{Q}$, ki so vsebovani v S. Naj bo B tisti, ki je manjši.

2. **(posodobi $\mathcal{X}$).** Odstrani B iz S in ustvari nov (enostaven) blok S' particije $\mathcal{X}$, ki vsebuje B kot edini blok iz $\mathcal{Q}$. Če je S še vedno sestavljen, vstavi S nazaj v C.

3. **(izračunaj $E^{-1}(B)$).** Kopiraj vozlišča iz B v začasno množico B'. (To nam olajša razbitje B glede na samega sebe med rafinacijo.) Izračunaj $E^{-1}(B)$ tako, da pregledaš povezave uEv za $v \in B$ in dodaš vsako vozlišče u iz take povezave v $E^{-1}(B)$, če še ni bilo dodano. Dvojnikom se izognemo tako, da vozlišča označimo, ko naletimo nanje, nato pa jih še povežemo med sabo za kasnejše odstranjevanje oznak. Med istim pregledom izračunaj PREŠTEJ$(u, B) = |\{v \in B; uEv\}|$, shrani to preštetje v novo število in naj u kaže nanj. Ta preštetja bomo rabili v koraku 5.

4. **(rafiniraj $\mathcal{Q}$ glede na B).** Vsak blok D particije $\mathcal{Q}$, ki vsebuje nekaj elementov (vozlišč) iz $E^{-1}(B)$, razbij na $D_1 = D \cap E^{-1}(B)$ in $D_2 = D \setminus D_1$. To stori s pregledom elementov $E^{-1}(B)$. Za obdelavo elementa $u \in E^{-1}(B)$ poišči blok D particije $\mathcal{Q}$, ki ga vsebuje, in ustvari blok D', če še ne obstaja. Premakni u iz D v D'.

   Med pregledom zgradi seznam tistih blokov D, ki so razbiti. Po pregledu obdelaj seznam razbitih blokov. Za vsak blok D, ki ima pripadajoč blok D', označi D' tako, da nič več ne pripada D (da bo pravilno obdelan v naslednji iteraciji koraka 4). Odstrani zapis D, če je D prazen, če pa D ni prazen in če je blok particije $\mathcal{X}$, ki vsebuje D in D', postal sestavljen z razbitjem, dodaj ta blok v C.

5. **(izračunaj $E^{-1}(B) \setminus E^{-1}(S \setminus B)$).** Preglej povezave uEv za $v \in B'$. Za obdelavo povezave uEv določi PREŠTEJ(u, B) (nanj kaže u) in PREŠTEJ(u, S) (nanj kaže uEv). Če PREŠTEJ(u, B) =PREŠTEJ(u, S), dodaj u v $E^{-1}(B) \setminus E^{-1}(S \setminus B)$, če še ni bil dodan.

6. **(rafiniraj $\mathcal{Q}$ glede na $S \setminus B$).** Postopaj tako kot v koraku 4, vendar preglej $E^{-1}(B) \setminus E^{-1}(S \setminus B)$ (izračunana v koraku 5) namesto $E^{-1}(B)$.

7. **(posodobi preštetja).** Preglej povezave uEv za $v \in B'$. Za obdelavo povezave uEv zmanjšamo PREŠTEJ(u, S) (na katerega kaže uEv) za 1. Če to preštetje postane nič, potem izbrišemo zapis PREŠTEJ in uEv naj kaže na PREŠTEJ(u, B) (nanj kaže u). Po pregledu vseh primernih povezav zavrzi B'.

Zavedajmo se, da so v koraku 5 pregledane le povezave, ki se končajo v B'. Korak 5 je pravilen (izračuna $E^{-1}(B) \setminus E^{-1}(S \setminus B)$), saj za vsako vozlišče $u \in E^{-1}(B)$ velja, da je $u \in E^{-1}(B) \setminus E^{-1}(S \setminus B) \Leftrightarrow u \notin E^{-1}(S \setminus B) \Leftrightarrow$ vse povezave, ki se začnejo v u in končajo v S, se končajo v $B \Leftrightarrow$ PREŠTEJ(u, B) =PREŠTEJ(u, S).

Pravilnost implementacije enostavno sledi iz zgornje razprave o trojnem razbitju. Za vsako povezavo, ki se konča v B, porabimo $\mathcal{O}(1)$ časa, in enako za vsako vozlišče iz B, torej ima en korak rafinacije časovno zahtevnost enako $\mathcal{O}(|B| + \sum_{v \in B} |E^{-1}(\{v\})|)$. Časovna zahtevnost celotnega algoritma je največ $\mathcal{O}(m \log n)$. Učinkovitost algoritma se da izboljšati za konstanten faktor s kombiniranjem različnih korakov, ki smo jih zaradi boljšega razumevanja obravnavali posebej.

7.3.4 Problem dodelitve vlog

V tem poglavju bomo proučevali računsko zahtevnost odločitvenega problema, ali dan graf ustreza regularni dodelitvi vlog s prej določenim grafom vlog ali številom ekvivalenčnih razredov. Proučevali bomo le neusmerjene grafe.

Najpopolnejšo karakterizacijo sta podala Fial in Paulusma [163]. Naj bo $k \in \mathbb{N}$ in R neusmerjen graf, ki ima lahko zanke.

Problem 7.25 (k-dodelitev vlog (k-DV)) *Naj bo G graf.*

Vprašanje: *Ali obstaja regularna ekvivalenčna relacija za G z natanko k ekvivalenčnimi razredi?*

Problem 7.26 (R-dodelitev vlog R-DV)) *Naj bo G graf.*

Vprašanje: *Ali obstaja regularna dodelitev vlog $r : V(G) \to V(R)$ z grafom vlog R?*

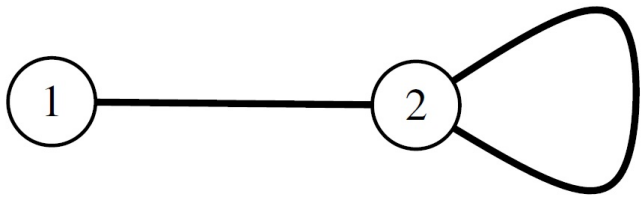
Zahtevamo, da je dodelitev vlog surjektivna preslikava.

Izrek 7.27 *k-DV je polinomsko rešljiva za $k = 1$ in je $\mathcal{NP}$-polna za vse $k \geq 2$.*

Izrek 7.28 *R-DV je polinomsko rešljiva, če je vsaka komponenta grafa R sestavljena iz izoliranih vozlišč, brez ali z zankami, ali iz dveh vozlišč brez zank, sicer je $\mathcal{NP}$-polna.*

Dokažimo izrek v posebnem primeru problema R-dodelitve vlog.

Izrek 7.29 *Naj bo R_0 graf na spodnji sliki. Potem je R_0-DV $\mathcal{NP}$-poln.*



Slika 7.5: Graf vlog R_0

Dokaz. Ker lahko v polinomskem času preverimo, ali je dana funkcija $r : V(G) \to \{1, 2\}$ 2-dodelitev vlog z grafom vlog R_0, je lahko videti, da je R_0-DV v $\mathcal{NP}$.

Pokazali bomo, da je 3SAT problem polinomsko spremenljiv v R_0-DV. Naj bo $U = \{u_1, \ldots, u_n\}$ množica spremenljivk in $C = \{c_1, \ldots, c_m\}$ množica stavkov, vsak je sestavljen iz natanko treh črk. Konstruirali bomo graf $G = (V, E)$ tako, da je G 2-DV z grafom vlog R_0 natanko tedaj, ko lahko C zadostimo.

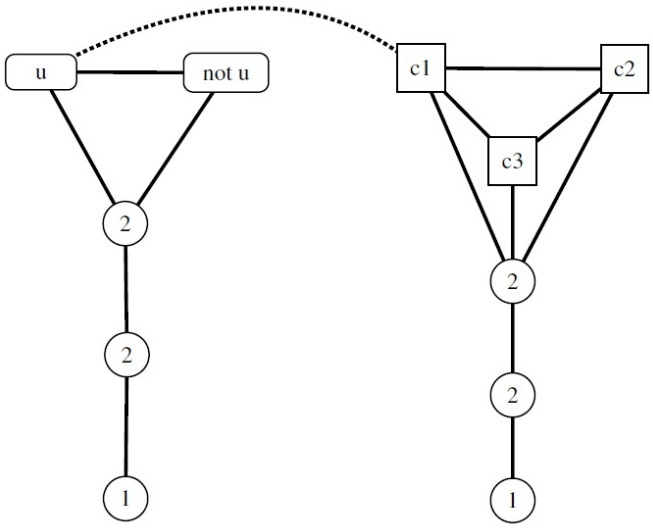
Konstrukcijo bomo naredili iz dveh komponent, tistih, ki predpisujejo resnico, in tistih, ki testirajo zadoščenost.

Za vsako spremenljivko $u_i \in U$ obstaja komponenta, ki predpiše resnico, $T_i = (V_i, E_i)$ z

$$V_i := \{u_i, \bar{u}_i, a_{i1}, a_{i2}, a_{i3}\},$$

$$E_i := \{u_i\bar{u}_i, u_ia_{i3}, \bar{u}_ia_{i3}, a_{i1}a_{i2}, a_{i2}a_{i3}\}.$$

Čeprav pišemo $u_i\bar{u}_i$ za povezavo $\{u_i, \bar{u}_i\}$, je graf neusmerjen. Intuicija za konstrukcijo T_i je naslednja: če graf, ki vsebuje T_i kot podgraf, tako da so a_{ij} sosednje le z vozlišči v



Slika 7.6: Komponente, ki predpisujejo resnico spremenljivke u (levo); komponente, ki testirajo zadoščenost za stavek $\{c_1, c_2, c_3\}$ (desno); vmes komunikacjiska povezava, če je črka c_1 enaka u (črtkano). Vloge vozlišč v navpičnih poteh so enolično določene (kot prikazujejo oznake 1 in 2), če je dodelitev vlog regularna z grafom vlog R_0.

V_i, dopušča regularno dodelitev vlog r z grafom vlog R_0, potem mora biti $r(a_{i1}) = 1$, ker je a_{i1} stopnje 1, in vozlišče, ki mu je dodeljena 2, mora imeti stopnjo ≥ 2. Potem je $r(a_{i2}) = 2$, ker je 1-vozlišče sosednje 2-vozlišču, in $r(a_{i2}) = 2$, ker je 2-vozlišče sosednje 2-vozlišču. Končno je natanko eni od u_i ali $\bar{u}_i$ dodeljena 2, kar pomeni, da je spremenljivka u_i nastavljena na resnico ali na neresnico. Torej komponenta T_i zagotavlja, da spremenljivka dobi resnico ali neresnico.

Za vsak stavek $c_j \in C$, naj bodo c_{j1}, c_{j2} in c_{j3} tri vozlišča, ki se ujemajo s tremi črkami v stavku c_j. Potem obstaja komponenta za testiranje izpolnjenosti $S_j = \left(V'_j, E'_j\right)$ z

$$V'_j := \{c_{j1}, c_{j2}, c_{j3}, b_{j1}, b_{j2}, b_{j3}\},$$

$$E'_j := \{c_{j1}c_{j2}, c_{j1}c_{j3}, c_{j2}c_{j3}, c_{j1}b_{j3}, c_{j2}b_{j3}, c_{j3}b_{j3}, b_{j1}b_{j2}, b_{j2}b_{j3}\}.$$

Intuicija za konstrukcijo T_i je naslednja: če graf, ki vsebuje S_j kot podgraf, tako da so b_{jl} sosednji le z vozlišči v V_j, dopušča regularno dodelitev vlog r z grafom vlog R_0, potem je nujno $r(b_{j1}) = 1$, $r(b_{j2}) = r(b_{j3}) = 2$, kar zagotavlja, da je enemu izmed vozlišč c_{j1}, c_{j2} ali c_{j3} dodeljena 1, kar zagotavlja, da je vsakemu sosednjemu vozlišču tega 1-vozlišča dodeljena 2.

Do sedaj je konstrukcija odvisna le od števila spremenljivk in stavkov. Edini del konstrukcije, odvisen od tega, katera črka se pojavi v katerem stavku, je skupina komunikacijskih povezav. Za vsak stavek $c_j = \{x_{j1}, x_{j2}, x_{j3}\} \in C$ so komunikacijske povezave, ki izhajajo iz S_j, dane z

$$E''_j := \{c_{j1}x_{j1}, c_{j2}x_{j2}, c_{j3}x_{j3}\}.$$

(x_{jl} so spremenljivke v U ali njihove negacije.) Opazimo, da za vsako povezavo c_{jk} obstaja natanko eno vozlišče, ki je sosednje c_{jk} v E''_j. To vozlišče ustreza vozlišču črke c_{jk} v stavku c_j.

Za zaključek konstrukcije primera R_0-DV naj bo $G = (V, E)$, kjer je V unija vseh V_i in V'_j ter E unija vseh E_i, E'_j in E''_j.

Naj bo dana regularna dodelitev vlog za G z grafom vlog R_0. Za vsak $j = 1, \ldots, m$ obstaja vozlišče c_{jk}, tako da je $r(c_{jk}) = 1$, kar pomeni, da je črki, ki ustreza sosednjemu vozlišču, dodeljena 2. Če postavimo to črko na resnično, bomo zadostili stavku c_j.

Torej smo dokazali, da formuli lahko zadostimo, če je G regularno R_0 dodeljiv.

Druga smer: Naj neka dodelitev resničnosti zadošča C. Dodelitev $r : V(G) \to \{1, 2\}$ dobimo na sledeč način. Za vsak $i = 1, \ldots, n$ postavimo $r(u_i) = 2$ in $r(\bar{u_i}) = 1$ natanko takrat, ko je spremenljivka u_i resnična, in postavimo vlogi vozlišč a_{ik} in b_{jk} kot implicira regularnost r. Za vsak $j = 1, \ldots, m$ naj bodo c_{jk}, $k \in \{1, 2, 3\}$, vozlišča, ki ustrezajo črkam v stavku c_j, ki so resnične. Tak k obstaja, ker dodelitev resničnosti zadošča C. Postavimo $r(c_{jk}) := 1$ in $r(c_{jl}) := 2$ za $l \in \{1, 2, 3\}$, $l \neq k$.

Dokaz je zapleten zaradi dejstva, da je več kot ena črka v stavku lahko resnična, toda $r(c_{jk}) = 1$ je dovoljeno samo za en $k \in \{1, 2, 3\}$. Ker je lahko 2-vozlišče sosednje drugemu 2-vozlišču, to ne uniči regularnosti r. □

□

7.3.5 Obstoj k-dodelitev vlog

V prejšnjem poglavju smo videli, da je odločitveni problem, ali graf dopušča regularno ekvivalenčno relacijo z natanko k ekvivalenčnimi razredi, $\mathcal{NP}$-poln za splošen graf. Vseeno je lahko preveriti zadostne, če ne celo potrebne pogoje, ki zagotavljajo obstoj regularne k-dodelitve vlog. Na kratko, pogoj je, da se graf ne razlikuje preveč od regularnega grafa.

Izrek 7.30 *Za vse $k \in \mathbb{N}$ obstaja konstanta $c_k \in \mathbb{R}$, tako da za vse grafe z minimalno stopnjo $\delta = \delta(G)$ in maksimalno stopnjo $\Delta = \Delta(G)$, ki zadoščajo*

$$\delta \geq c_k \log(\Delta),$$

obstaja regularna ekvivalenčna relacija za G z natanko k ekvivalenčnimi razredi.

Da izključimo trivialen protiprimer, predpostavimo, da obravnavamo samo grafe z najmanj k vozlišči. Za dokaz izreka potrebujemo verzijo Lovaszove lokalne leme.

Izrek 7.31 *Naj bodo A_i, $i \in I$, dogodki v diskretnem verjetnostnem prostoru. Če obstaja M, da je za vsak $i \in I$*

$$|\{A_j;\ A_j \text{ odvisen od } A_i\}| \leq M,$$

in če obstaja $p > 0$, tako da je $Pr(A_i) \leq p$ za vse $i \in I$, potem

$$ep(M + 1) \leq 1 \Rightarrow Pr\left(\cap_{i \in I} \bar{A}_i\right) > 0,$$

kjer je e Eulerjevo število $\sum_{i=0}^{\infty} \frac{1}{i!}$.

Dokaz. (Izreka 7.30) Definirajmo $r : V \to \{1, \ldots, k\}$ na naslednji način: za vsak $v \in V$ izberemo $r(v)$ enakomerno naključno iz $\{1, \ldots, k\}$.

Za $v \in V$ naj bo A_v dogodek, da $r(N(v)) \neq \{1, \ldots, k\}$. To je

$$Pr(A_v) \leq k \left(\frac{k-1}{k}\right)^{d(v)} \leq k \left(\frac{k-1}{k}\right)^{\delta(G)}.$$

Ker so $r(w)$ izbrani neodvisno in za fiksirano vrednost i, je verjetnost, da i ni uporabljen za nobeno sosednje vozlišče v, enaka $\left(\frac{k-1}{k}\right)^{d(v)}$ in za i obstaja k izbir.

Opazimo tudi, da sta A_v in A_w neodvisna natanko tedaj, ko $N(v) \cap N(w) \neq \emptyset$. Zato A_v z $M := \Delta(G)^2$ in $p := k\left(\frac{k-1}{k}\right)^{\delta(G)}$ zadošča pogojem Lovaszove lokalne leme. Torej

$$ek\left(\frac{k-1}{k}\right)^{\delta(G)}\left(\Delta(G)^2+1\right) \leq 1 \Rightarrow Pr\left(\cap_{v\in V}\bar{A}_v\right) > 0.$$

Če drži desna stran zgornje enačbe, obstaja najmanj en r, tako da $r(N(v)) = \{1,\ldots,k\}$ za vsak $v \in V$, tj. obstaja najmanj ena regularna k-dodelitev vlog. Leva stran je ekvivalentna

$$\delta(G) \geq \frac{\log\left(ek\left(\Delta(G)^2+1\right)\right)}{\log\frac{k}{k-1}}.$$

Očitno obstaja konstanta c_k, tako da je $c_k \log(\Delta(G))$ večje od desne strani zgornje neenakosti. □

□

Zaključek. Regularna ekvivalenčna relacija je dobro raziskana v računalništvu. Rezultati kažejo, da veliko regularnih ekvivalenčnih relacij obstaja tudi v neregularnih grafih, toda ni znano, kako definirati in/ali izračunati najboljšo. Obstajajo hitri algoritmi za izračun maksimalne regularne ekvivalenčne relacije ali regularne notranjosti prvotne particije. Maksimalna regularna ekvivalenčna relacija bi lahko bila pomembna za usmerjene grafe. Regularna notranjost bi lahko bila dobra dodelitev vlog, če bi imeli idejo za particijo. Določitev števila ekvivalenčnih razredov grafa vlog prinaša v splošnem primeru $\mathcal{NP}$-polne probleme.

7.4 Ostale ekvivalenčne relacije

V tem poglavju na kratko omenimo ostale tipe ekvivalenčnih relacij vlog.

7.4.1 Ekstaktna dodelitev vlog

V tem poglavju bomo definirali razred eksaktne ekvivalenčne relacije, ki je podmnožica regularnih ekvivalenčnih realcij. Prirejene particije poznamo kot ekvitabilne particije v teoriji grafov, prvotno definirane kot delitelji grafov. Medtem ko je za regularno ekvivalenčno relacijo pomembno le, če se vloga pojavi v soseščini, je pri eksaktni ekvivalenčni relaciji pomembno število pojavitev.

Model grafa v tem poglavju je neusmerjen multigraf.

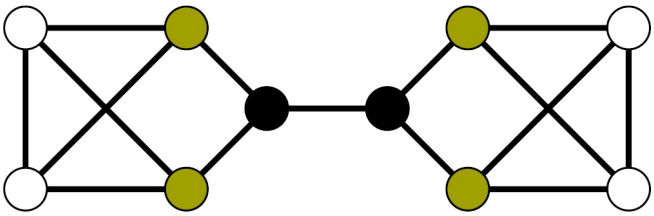
Definicija 7.32 DV je *eksaktna*, če za vse $u, v \in V$ velja

$$r(u) = r(v) \Rightarrow r(N(u)) = r(N(v)),$$

kjer je zadnja enačba enakost multimnožic, tj. vozlišča, ki imajo isto vlogo, morajo imeti enako število pojavitev vsake vloge v svojih soseščinah.

Na spodnji sliki barvanje grafa definira eksaktno dodelitev vlog za spodnji graf.

Medtem ko je ekvivalenčna relacija regularna za multigraf natanko takrat, ko je regularna za vsak induciran enostaven graf, pa je pri eksaktni ekvivalenčni relaciji večkratnost povezave pomembna. Lahko je videti, da če je dodelitev vlog eksaktna, potem je regularna. Obrat ne velja.



Slika 7.7: Eksaktna dodelitev vlog

Definicija 7.33 Particijo $\mathcal{P} = \{C_1, \ldots, C_k\}$ množice vozlišč V neusmerjenega (multi-)grafa $G = (V, E)$ imenujemo *ekvitabilna*, če obstajajo števila b_{ij}, $i, j = 1, \ldots, k$, tako da ima vsako vozlišče v razredu C_i natanko b_{ij} sosedov v razredu C_j. Matrika $B = (b)_{i,j=1,\ldots,k}$ definira (*usmerjen*) multigraf, ki ga imenujemo *kvocient* G po modulu $\mathcal{P}$, označimo pa z $G/\mathcal{P}$.

Particija je ekvitabilna natanko tedaj, ko je prirejena dodelitev vlog eksaktna. Zgornja definicija razširja definicijo kvocienta ali grafa vlog na multigrafe. To je možno le za eksaktne dodelitve vlog.

Tudi če je graf neusmerjen, je kvocient lahko usmerjen. Večkratnost povezave se lahko razlikuje od večkratnosti nasprotno usmerjene povezave. To se zgodi, ko sta dva sosednja ekvivalenčna razreda različne velikosti.

Eksaktna dodelitev vlog je združljiva z algebraičnimi lastnostmi grafa.

Izrek 7.34 *Naj bo G graf in $\mathcal{P}$ ekvitabilna particija. Potem karakteristični polinom kvocienta $G/\mathcal{P}$ deli karakteristični polinom grafa G.*

Izrek implicira, da je spekter kvocienta $G/\mathcal{P}$ podmnožica spektra G.

Množica vseh eksaktnih dodelitev vlog grafa tvori mrežo. Maksimalno eksaktno dodelitev vlog grafa lahko izračunamo s prilagoditvijo algoritma iz poglavja 7.3.3. Veliko problemov v povezavi z eksaktnimi dodelitvami vlog je tudi $\mathcal{NP}$-polnih. Na primer, odločitveni problem, ali graf G dopušča eksaktno dodelitev vlog s kvocientom R, je $\mathcal{NP}$-poln, če sta oba, G in R, vhodna podatka, ali če je R fiksiran. To velja, ker lahko $\mathcal{NP}-$poln odločitveni problem, ali ima 3-regularen graf popolno kodo, formuliramo kot odločitveni problem, ali ima G eksaktno dodelitev vlog s kvocientom

$$R = \begin{bmatrix} 0 & 3 \\ 1 & 2 \end{bmatrix}.$$

Kvocient po ekvitabilni particiji ima veliko več skupnega z originalnim grafom kot kvocient grafa vlog po regularni ekvivalenci. Eksaktna dodelitev vlog zagotavlja tudi, da imajo ekvivalentna vozlišča isto stopnjo, kar ni res pri regularni dodelitvi vlog.

Zaključek. Eksaktna dodelitev vlog, ki jo imenujemo tudi ekvitabilna particija, je dobro raziskana v algebraični teoriji grafov. Medtem ko je nekaj problemov v povezavi z ekvitabilnimi particijami $\mathcal{NP}$-polnih, obstajajo učinkoviti algoritmi za izračun maksimalne ekvitabilne particije grafa ali za izračun najbolj grobe ekvitabilne rafinacije prvotne particije. Te algoritme lahko uporabimo za izračun dodelitve vlog, vendar rezultati v večini primerov vsebujejo preveč razredov in zgrešijo osnovne strukture.

7.4.2 Avtomorfne in orbitne ekvivalenčne relacije

Avtomorfne ekvivalenčne relacije izražajo izmenljivost točk.

Definicija 7.35 Naj bo $G = (V, E)$ graf in $u, v \in V$. u in v imenujemo *avtomorfno ekvivalentni*, če obstaja avtomorfizem ϕ grafa G, tako da velja $\phi(u) = v$.

Avtomorfno ekvivalentnih vozlišč ne moremo razlikovati le v izrazih grafovske strukture. Lahko razpravljamo o tem, da morajo vsaj avtomorfno ekvivalentna vozlišča igrati isto vlogo. Lahko je videti, da so strukturno ekvivalentna vozlišča avtomorfno ekvivalentna.

Particija množice točk, ki ima lastnost, da je vsak par ekvivalentnih vozlišč avtomorfno ekvivalenten, ni nujno regularna ekvivalenca.

Trditev 7.36 *Naj bo $G = (V, E)$ graf z grupo avtomorfizmov $A(G)$ in naj bo H podgrupa $A(G)$. Potem dodelitev vlog glede na orbite H definira eksaktno dodelitev vlog za G. Tako particijo imenujemo* particija orbit.

Dokaz. Naj bo r dodelitev vlog kot v trditvi. Če je $r(u) = r(v)$, potem obstaja $\phi \in H$, tako da je $\phi(u) = v$. Če je $x \in N^+(u)$, potem je $\phi(x) \in N^+(\phi(u)) = N^+(v)$. Dalje: $r(x) = r(\phi(x))$ po definiciji. Sledi da je $r(N^+(u)) \subset r(N^+(v))$ kot multimnožici. Drugo inkluzijo in ujemajočo se trditev za vhodno soseščino pokažemo podobno. □

□

V posebnem, ekvivalenčna relacija orbit je regularna. Na primer, na sliki 3 barvanje definira particijo orbit avtomorfizmov grupe.

Množica ekvivalenčnih relacij orbit tvori podmnožico množice eksaktnih ekvivalenčnih relacij, kar lahko dokažemo s pomočjo kateregakoli regularnega grafa, ki ni vozliščno tranzitiven. Na primer, particija na eno samo množico grafa na sliki 3 je eksaktna, ni pa orbitna particija.

Zgornjo trditev lahko uporabimo pri dokazu, da ima vsak neusmerjen vlogovno primitiven graf trivialno grupo avtomorfizmov. To ni res za usmerjene grafe, kar lahko vidimo pri usmerjenih ciklih praštevilske dolžine.

Izračun orbitne ekvivalenčne relacije je ekvivalenten izračunu grupe avtomorfizmov.

Zaključek. Avtomorfno ekvivalentnih vozlišč ne moremo razlikovati v izrazih grafovske strukture, ampak le z dodatnimi karakteristikami. Nadalje lahko razpravljamo, da le avtomorfno ekvivalentna vozlišča igrajo isto vlogo. Zdi se, da je izračun avtomorfne ekvivalenčne relacije težak, toda v neregularnih omrežjih se ne bodo pojavili pomembni avtomorfizmi.

7.4.3 Popolna ekvivalenčna relacija

Popolna ekvivalenčna relacija je regularna ekvivalenčna relacija z dodatno zahtevo. Izraža idejo, da mora obstajati razlog, da vozlišči nista ekvivalentni.

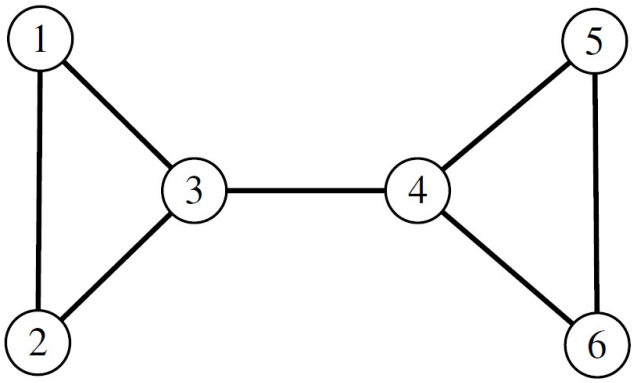
Definicija 7.37 Dodelitev vlog r definira *popolno ekvivalenčno relacijo*, če za vsak $u, v \in V$ velja

$$r(u) = r(v) \Leftrightarrow r\left(N^+(u)\right) = r\left(N^+(v)\right), r\left(N^-(u)\right) = r\left(N^-(v)\right).$$

Regularna ekvivalenčna relacija je popolna natanko tedaj, ko induciran graf vlog nima krepko strukturno ekvivalentnih vozlišč.

Množica popolnih ekvivalenčnih relacij grafa je mreža, ki ni niti podmreža vseh ekvivalenčnih relacij niti podmreža regularnih ekvivalenčnih relacij. Popolna notranjost ekvivalenčne relacije $\sim$ je najbolj groba popolna rafinacija $\sim$. V nasprotju z regularno notranjostjo popolna notranjost v splošnem ne obstaja.

Izrek 7.38 *V splošnem tranzitivno zaprtje unije dveh popolnih ekvivalenčnih relacij ni popolno. V posebnem za nekaj ekvivalenčnih relacij ne obstaja popolna notranjost.*



Slika 7.8: Supremum dveh popolnih ekvivalenčnih relacij ni popoln

Dokaz. Vzemimo graf na sliki 7.8 in dve popolni particiji $\mathcal{P}_1 = \{\{1,5\},\{2,6\}\{3,4\}\}$ in $\mathcal{P}_2 = \{\{1,2\},\{5,6\}\{3\},\{4\}\}$. Tranzitivno zaprtje $\mathcal{P}_1$ in $\mathcal{P}_2$ je $\mathcal{P} = \{\{1,2,5,6\},\{3,4\}\}$, ki ni popolno. Drugi del: $\mathcal{P}_1$ in $\mathcal{P}_2$ sta popolni rafinaciji $\mathcal{P}$ in obe sta maksimalni glede na to lastnost. □

□

Druga izjava ima enostavnejši dokaz: za graf z dvema krepko strukturno ekvivalentnima vozliščema particija na singeltone nima popolne rafinacije.

Nekaj odločitvenih problemov pri popolni ekvivalenčni relaciji je $\mathcal{NP}$-polnih, kar lahko vidimo iz izrekov 7.27 in 7.28, zoženih na grafe vlog brez krepko strukturno ekvivalentnih vozlišč. Čeprav popolna ekvivalenčna relacija izključi nekaj trivialnih regularnih ekvivalenčnih relacij, ni dokaza, zakaj vloge ne bi mogle biti krepko strukturno ekvivalentne.

Zaključek. Popolna ekvivalenčna relacija je regularna ekvivalenčna relacija z dodatno zahtevo, vendar ne daje boljše dodelitve vlog. Nekaj matematičnih lastnosti regularne ekvivalenčne relacije se izgubi in obstajajo primeri, kjer pogoji za popolno ekvivalenčno relacijo izključijo dobre regularne dodelitve vlog.

7.4.4 Relativna regularna ekvivalenčna relacija

Relativna regularna ekvivalenčna relacija izraža idejo, da imajo ekvivalentna vozlišča ekvivalentne soseščine v grobi, vnaprej definirani meri.

Definicija 7.39 Naj bo $G = (V, E)$ graf in $r : V \to W$, $r_0 : V \to W_0$ dodelitvi vlog. r imenujemo *regularno relativna* glede na r_0, če $r \leq r_0$ in za vsaka $u, v \in V$ velja

$$r(u) = r(v) \Rightarrow r_0\left(N^+(u)\right) = r_0\left(N^+(v)\right), r_0\left(N^-(u)\right) = r_0\left(N^-(v)\right).$$

Značilna aplikacija relativne regularne ekvivalenčne relacije je dana z omrežjem simetričnih prijateljskih vezi, ki je prvotno razdeljeno na dve disjunktni kliki A in B. Domnevamo, da ima znotraj vsake klike vsak član najmanj eno vez z ostalimi člani iste klike. Particija na ti dve kliki bo regularna, če ni vezi med klikama ali pa če ima poleg vezi znotraj skupine vsak udeleženec najmanj eno vez s članom iz ostale skupine. Toda predvidimo, da ima nekaj udeležencev prijateljske vezi s člani iz ostale skupine. Particija na A in B ni nič več regularna. Sedaj razdelimo vsako skupino na člane, ki imajo prijateljske vezi s člani iz ostale skupine, in tiste, ki jih nimajo. Particijo razdelimo na A_1, A_2, B_1 in B_2. Ta particija v splošnem ni regularna: mogoče bo nekaj članov, npr. v A_1, ki bodo imeli nekaj vezi samo s člani A_1 ali samo s člani A_2 ali z obojimi; ti nimajo ekvivalentnih soseščin. Toda imajo ekvivalentne soseščine glede na grobo particijo na A in B. Torej je particija na A_1, A_2, B_1 in B_2 relativno regularna glede na particijo na A in B.

Trditev 7.40 *Naj bodo $\sim, \sim_1$ in $\sim_2$ ekvivalenčne relacije na V, tako da $\sim_1 \leq \sim_2$ in $\sim_2$ regularno relativna glede na $\sim$. Potem isto velja tudi za $\sim_1$.*

Trditev implicira, da je množica ekvivalenčnih relacij, ki so relativno regularne glede na fiksirano ekvivalenčno relacijo $\sim$, podmreža vseh ekvivalenčnih relacij in je v celoti opisana z maksimumom množice, označenim z $MRRE(\sim)$. Izračun $MRRE(\sim)$ je mogoč v linearnem času s prilagoditvijo algoritma 7.2.3 za izračun maksimalne strukturne ekvivalenčne relacije: namesto razdelitve ekvivalenčnih razredov z vidika vozlišča uporabimo razdelitev z vidika razredov $\sim$. Razredi $\sim$ so fiksirani in $MRRE(\sim)$ najdemo potem, ko so bili vsi razredi $\sim$ enkrat predelani.

Zaključek. Relativna regularna ekvivalenčna relacija je računsko preprosta, toda potrebuje začetno particijo vozlišč, in ker je združljivostni pogoj le lokalen, ne pričakujemo, da predstavlja globalno omrežno strukturo. Najbolj se uporablja pri večkratnih in sestavljenih relacijah.

7.5 Grafi z več relacijami

Pogosto nas na grafu zanima več relacij, ki so na nek način odvisne med sabo. V tem primeru obravnava vsake relacije posebej ne zadostuje, obravnavati moramo vse relacije hkrati.

Definicija 7.41 *Graf z več relacijami* $\mathcal{G} = (V, \mathcal{E})$ je sestavljen iz končne množice vozlišč V in končne množice relacij $\mathcal{E} = \{E_i \mid i = 1, \ldots, p\}$, kjer je $p \in \mathbb{N}$ in $E_i \subseteq V \times V$ množica povezav.

Graf z več relacijami je torej nabor grafov na isti množici vozlišč. Relaciji sta enaki, če sta njuni množici povezav enaki. Če imamo več enakih relacij, jih lahko nadomestimo z eno samo in dobimo isti graf z več relacijami (saj je $\mathcal{E}$ množica).

Definicija 7.42 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami in $r : V \to W$ dodelitev vlog. *Graf vlog* je graf z več relacijami $\mathcal{R} = (W, \mathcal{F})$, kjer je $\mathcal{F} = \{F_i \mid i = 1, \ldots, p\}$ in $F_i = \{(r(u), r(v)) \mid (u, v) \in E_i\}$.

Lahko se zgodi, da je $F_i = F_j$, čeprav $E_i \neq E_j$. Ker je graf vlog grafa z več relacijami tudi graf z več relacijami, lahko tudi več enakih relacij med vlogami nadomestimo z eno samo.

Iz te definicije vidimo, da so dodelitve vlog pravzaprav preslikave, ki slikajo vozlišča v vozlišča in relacije v relacije. Torej $r : V \to W$ enolično določa preslikavo relacij $r_{rel} : \mathcal{E} \to \mathcal{F}$.

Definicije različnih tipov particij množice vozlišč lahko priredimo za grafe z več relacijami s pomočjo naslednje definicije.

Definicija 7.43 Dodelitev vlog $r : V \to W$ za graf z več relacijami $\mathcal{G} = (V, \mathcal{E})$ je *tipa t, če je za vsak* $E \in \mathcal{E}$ *tipa t za graf* (V, E).

Če za tip t vzamemo regularnost, dobimo naslednjo definicijo.

Definicija 7.44 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami. Dodelitev vlog $r : V \to W$ je *regularna* za $\mathcal{G}$, če je za vsak $E \in \mathcal{E}$ regularna za graf (V, E).

Regularne dodelitve vlog zagotavljajo, da za ekvivalentna vozlišča pri vseh relacijah velja, da so v njihovih soseščinah zastopane iste vloge.

Večina izrekov, ki smo jih spoznali za grafe z eno relacijo, velja tudi za grafe z več relacijami, če ustrezne tipe particij množice vozlišč priredimo s pomočjo definicije 7.43.

Definicija 7.45 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami in $u, v \in V$. *Sveženj (relacij)* od u do v je množica $B_{uv} = \{E \in \mathcal{E} \mid (u, v) \in E\}$.

Definicija 7.46 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami in $\mathcal{B}$ množica vseh nepraznih svežnjev. Za vsak sveženj $B \in \mathcal{B}$ definiramo graf z množico vozlišč V in množico povezav M_B, za katero velja

$$(u, v) \in M_B \;\Leftrightarrow\; B_{uv} = B.$$

Označimo $\mathcal{M} = \{M_B \mid B \in \mathcal{B}\}$. M_B je *mnogoterna relacija*, inducirana z grafom $\mathcal{G}$, $MPX(\mathcal{G}) := (V, \mathcal{M})$ pa *mnogoteren graf* grafa $\mathcal{G}$.

Za poljuben par vozlišč $u, v \in V$ je $(u, v) \in M_B$ za natanko en $B \in \mathcal{B}$ ali pa $(u, v) \notin M_B$ za noben $B \in \mathcal{B}$. Zato si lahko mnogoteren graf grafa $\mathcal{G}$ predstavljamo kot graf z eno relacijo in oznakami na povezavah. Takim grafom pravimo mnogoterni grafi. Torej: mnogoteren graf je graf $\mathcal{G} = (V, \mathcal{M})$, v katerem za poljuben par relacij $M_1, M_2 \in \mathcal{M}$ velja $M_1 \cap M_2 = \emptyset$ ali $M_1 = M_2$. Mnogoteren graf $MPX(\mathcal{G})$ grafa $\mathcal{G}$ je primer mnogoternega grafa.

Sedaj lahko definiramo ekvivalenčno relacijo, ki zagotavlja, da imajo ekvivalentna vozlišča iste svežnje relacij.

Definicija 7.47 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami. Dodelitev vlog $r : V \to W$ je *mnogoterno regularna* za $\mathcal{G}$, če je regularna za $MPX(\mathcal{G})$.

Kot v zgornji definiciji lahko definiramo tudi mnogoterno krepko strukturno dodelitev vlog. Lahko je preveriti, da je krepko strukturna dodelitev vlog na grafu z več relacijami tudi krepko strukturna na njegovem mnogoternem grafu.

Opomba 7.48 Ekvivalentna definicija mnogoternih regularnih dodelitev vlog: Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf, kjer je $\mathcal{E} = \{E_1, \ldots, E_p\}$, in $\mathcal{M} := \left\{\bigcap_{i\in I} E_i \mid I \subseteq \{1, \ldots, p\},\ I \neq \emptyset\right\}$. Regularne dodelitve vlog za graf $(V, \mathcal{M})$ so natanko mnogoterno regularne dodelitve vlog za $\mathcal{G}$.

Regularne dodelitve vlog v splošnem niso mnogoterno regularne. Se pa regularnost ohranja v obratni smeri.

Trditev 7.49 *Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami, $C := MPX(\mathcal{G})$ in $r : V \to W$ dodelitev vlog. Potem velja:*

1. *Če je r regularna za C, potem je regularna za $\mathcal{G}$.*
2. *Če je r krepko strukturna za C, potem je krepko strukturna za $\mathcal{G}$.*

Dokaz. Naj bo $E \in \mathcal{E}$ in $u, v, u' \in V$ takšni, da je $r(u) = r(u')$ in $(u, v) \in E$. Potem je $E \in B_{uv}$. Vpeljimo naslednjo oznako: $N_F^+(a)$ in $N_F^-(a)$ naj označujeta izhodno in vhodno soseščino vozlišča $a \in V$ glede na neko množico povezav F. Ker je $(u, v) \in E$, je $v \in N_E^+(u)$. Naj bo B_{uv} sveženj relacij od u do v in $M := \{(w, z) \mid B_{wz} = B_{uv}\}$ pripadajoča mnogoterna relacija. Ker je $E \in B_{uv}$, za vsak $(w, z) \in M$ velja $E \in B_{wz}$. Torej za vsak $(w, z) \in M$ velja $(w, z) \in E$, kar pomeni, da je $M \subseteq E$.

1. Privzemimo, da je r regularna za C. Potem je regularna za (V, M), torej iz $r(u) = r(u')$ sledi $r(N_M^+(u)) = r(N_M^+(u'))$. Ker je $(u, v) \in E$ in $M \subseteq E$, je $v \in N_M^+(u)$. Zato obstaja takšen $v' \in N_M^+(u')$, da velja $r(v) = r(v')$. Potem je $(u', v') \in M$ in zato $B_{u'v'} = B_{uv}$, torej $E \in B_{u'v'}$. Sledi $(u', v') \in E$, zato je $v' \in N_E^+(u')$. S tem smo dokazali, da iz $r(u) = r(u')$ sledi, da za poljuben $v \in N_E^+(u)$ obstaja $v' \in N_E^+(u')$, da je $r(v) = r(v')$. To pomeni, da iz $r(u) = r(u')$ sledi $r(N_E^+(u)) \subseteq r(N_E^+(u'))$. Z zamenjavo vlog u in u' dobimo še vsebovanost v drugi smeri. Torej iz $r(u) = r(u')$ sledi $r(N_E^+(u)) = r(N_E^+(u'))$. Analogno dokažemo še za vhodne soseščine. Torej je r regularna za graf (V, E). To velja za poljuben $E \in \mathcal{E}$, torej je r regularna za $\mathcal{G}$.

2. Privzemimo, da je r krepko strukturna za C. Potem je r krepko strukturna za (V, M), torej iz $r(u) = r(u')$ sledi $N_M^+(u) = N_M^+(u')$. Ker je $v \in N_M^+(u)$, je $v \in N_M^+(u')$, torej $(u', v) \in M$. Potem je $B_{u'v} = B_{uv}$ in zato $E \in B_{u'v}$. Torej je $(u', v) \in E$, in zato $v \in N_E^+(u')$. S tem smo dokazali, da iz $r(u) = r(u')$ sledi, da za poljuben $v \in N_E^+(u)$ velja $v \in N_E^+(u')$. To pomeni, da iz $r(u) = r(u')$ sledi $N_E^+(u) \subseteq N_E^+(u')$. Z zamenjavo vlog u in u' dobimo še vsebovanost v drugi smeri. Torej iz $r(u) = r(u')$ sledi $N_E^+(u) = N_E^+(u')$. Analogno dokažemo še za vhodne soseščine. Torej je r krepko strukturna za graf (V, E). To velja za poljuben $E \in \mathcal{E}$, torej je po definiciji 7.43 r krepko strukturna za $\mathcal{G}$.

□

□

7.6 Polgrupa grafa

Družabna razmerja lahko tudi neposredno vplivajo drug na drugega. Npr. če sta A in B prijatelja ter A in C sovražnika, to verjetno vpliva na odnos med B in C. Sestavljene relacije (kot je v tem primeru PRIJATELJEV SOVRAŽNIK) želimo formalizirati s kompozicijo relacij.

Definicija 7.50 Naj bosta Q in R binarni operaciji na V.

$$QR := \{(u,v) \mid \exists w \in V,\ da\ je\ (u,w) \in Q\ in\ (w,v) \in R\}$$

je *(Boolov) produkt* Q z R.

Boolovo množenje relacij ustreza Boolovemu množenju pripadajočih matrik sosednosti. Za matriki A in B z elementi iz množice $\{0,1\}$ je Boolov produkt AB definiran z

$$(AB)_{ij} = \bigvee_{k=1}^{n} A_{ik} \wedge B_{kj}.$$

Definicija 7.51 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf (z več relacijami). *Polgrupa, inducirana z* $\mathcal{G}$, je definirana z

$$S(\mathcal{G}) := \{E_1 \dots E_k \mid k \in \mathbb{N},\ E_1, \dots, E_k \in \mathcal{E}\}.$$

Elementa v $S(\mathcal{G})$ sta enaka natanko tedaj, ko vsebujeta isto množico urejenih parov iz $V \times V$. $S(\mathcal{G})$ je res polgrupa, saj je množenje relacij asociativno.

Čeprav dolžina nizov $E_1 \dots E_k \in S(\mathcal{G})$ ni omejena, je $S(\mathcal{G})$ omejena, saj je število njenih elementov navzgor omejeno s številom vseh binarnih relacij na množici V, ki je $2^{|V|^2}$. Ker je število vseh možnih nizov neskončno, $S(\mathcal{G})$ pa končna, obstajajo nizi, ki so si enaki. Te enakosti nam dajo pomembne informacije o omrežju.

Definicija 7.52 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami in $r : V \to W$ dodelitev vlog. Za $Q \in S(\mathcal{G})$ je $r_{rel}(Q)$ relacija na W, definirana z

$$r_{rel}(Q) := \{(r(u), r(v)) \mid (u,v) \in Q\}.$$

$r_{rel}(Q)$ imenujemo *relacija, inducirana* s Q in r.

r torej inducira preslikavo r_{rel} na polgrupi $S(\mathcal{G})$. Če je r regularna dodelitev vlog, potem je $r_{rel}(S(\mathcal{G}))$ polgrupa grafa vlog grafa $\mathcal{G}$ pri relaciji r. V splošnem to ne velja. Dodelitve vlog v splošnem ne ohranjajo kompozicije, torej r_{rel} v splošnem ni homomorfizem polgrup.

Lema 7.53 *Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami, $Q, R \in S(\mathcal{G})$ in $r : V \to W$ dodelitev vlog, ki je regularna glede na Q in R. Potem je $r_{rel}(QR) = r_{rel}(Q) r_{rel}(R)$.*

Dokaz.

- ($\subseteq$): Vzemimo poljubna $w, w' \in W$, za katera velja $(w,w') \in r_{rel}(QR)$. Ker je $r_{rel}(QR) = \{(r(u), r(v)) \mid (u,v) \in QR\}$, obstajata $v, v' \in V$, za katera velja $r(v) = w$, $r(v') = w'$ in $(v,v') \in QR$. Potem obstaja $u \in V$, za katerega velja $(v,u) \in Q$ in $(u,v') \in R$. Sledi, da je $(r(v), r(u)) = (w, r(u)) \in r_{rel}(Q)$ in $(r(u), r(v')) = (r(u), w') \in r_{rel}(R)$, in zato $(w,w') \in r_{rel}(Q) r_{rel}(R)$. Torej je $r_{rel}(QR) \subseteq r_{rel}(Q) r_{rel}(R)$. Ta vsebovanost velja tudi če r ni regularna.

- ($\supseteq$): Vzemimo poljubna $w, w' \in W$, za katera velja $(w,w') \in r_{rel}(Q) r_{rel}(R)$. Potem obstaja $z \in W$, za katerega je $(w,z) \in r_{rel}(Q)$ in $(z,w') \in r_{rel}(R)$. Ker je $(w,z) \in r_{rel}(Q)$ in $r_{rel}(Q) = \{(r(a), r(b)) \mid (a,b) \in Q\}$, obstajata $v, u_1 \in V$, za katera velja $r(v) = w$, $r(u_1) = z$ in $(v,u_1) \in Q$. Ker je $(z,w') \in r_{rel}(R)$, obstajata $u_2, v' \in V$, za katera velja $r(u_2) = z$, $r(v') = w'$ in $(u_2,v') \in R$. Vpeljimo naslednjo oznako: $N_R^+(a)$ naj označuje izhodno soseščino vozlišča $a \in V$ glede na relacijo R. r je

regularna glede na R, zato iz $r(u_1) = r(u_2) = z$ sledi $r(N_R^+(u_1)) = r(N_R^+(u_2))$. Ker je $(u_2, v') \in R$, je $v' \in N_R^+(u_2)$ in zato $w' = r(v') \in r(N_R^+(u_2))$. Potem obstaja $v'' \in N_R^+(u_1)$, da velja $r(v'') = w' = r(v')$. Ker je $v'' \in N_R^+(u_1)$, je $(u_1, v'') \in R$. Ker velja tudi $(v, u_1) \in Q$, je $(v, v'') \in QR$, in zato $(r(v), r(v'')) = (w, w') \in r_{rel}(QR)$. Torej je $r_{rel}(Q)r_{rel}(R) \subseteq r_{rel}(QR)$.

□

□

Izrek 7.54 *Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami in $r : V \to W$ dodelitev vlog. Potem velja:*

1. *Če je r regularna glede na $\mathcal{E}$, potem je regularna glede na poljubno relacijo iz $S(\mathcal{G})$.*
2. *Če je r krepko strukturna glede na $\mathcal{E}$, potem je krepko strukturna glede na poljubno relacijo iz $S(\mathcal{G})$.*

Dokaz. Dovolj je dokazati, da za poljubni relaciji $Q, R \in S(\mathcal{G})$ velja: če je r regularna (krepko strukturna) glede na Q in R, potem je regularna (krepko strukturna) glede na QR. Ker lahko poljubno relacijo iz $S(\mathcal{G})$ zapišemo kot končen produkt relacij iz $\mathcal{E}$, bo iz zgornjega po indukciji na dolžino niza sledilo, da izrek velja.

1. Naj bo r regularna glede na relaciji Q in R. Izberimo poljubna $u, v \in V$, za katera velja $r(u) = r(v)$, ter poljuben $w \in N_{QR}^+(u)$. Potem je $(u, w) \in QR$, zato obstaja $t \in V$, za katerega velja $(u, t) \in Q$ in $(t, w) \in R$. Ker je $t \in N_Q^+(u)$, r regularna glede na Q in $r(u) = r(v)$, obstaja $s \in N_Q^+(v)$, za katerega velja $r(s) = r(t)$. Ker je $(t, w) \in R$, torej $w \in N_R^+(t)$, r regularna glede na R in $r(s) = r(t)$, obstaja $z \in N_R^+(s)$, za katerega velja $r(w) = r(z)$. Ker je $z \in N_R^+(s)$, je $(s, z) \in R$. Poleg tega je $s \in N_Q^+(v)$, torej $(v, s) \in Q$. Sledi, da je $(v, z) \in QR$, torej $z \in N_{QR}^+(v)$. Ker velja tudi $r(w) = r(z)$, smo s tem dokazali $r(N_{QR}^+(u)) \subseteq r(N_{QR}^+(v))$. Z zamenjavo vlog u in v dobimo še vsebovanost v nasprotni smeri. Torej iz $r(u) = r(v)$ sledi $r(N_{QR}^+(u)) = r(N_{QR}^+(v))$. Analogno dokažemo še $r(N_{QR}^-(u)) = r(N_{QR}^-(v))$. Torej je r regularna glede na QR.

2. Naj bo r krepko strukturna glede na relaciji Q in R. Izberimo poljubna $u, v \in V$, za katera velja $r(u) = r(v)$, ter poljuben $w \in N_{QR}^+(u)$. Potem je $(u, w) \in QR$, zato obstaja $z \in V$, za katerega velja $(u, z) \in Q$ in $(z, w) \in R$. Torej je $z \in N_Q^+(u)$. Ker je r krepko strukturna glede na Q in $r(u) = r(v)$, velja $N_Q^+(u) = N_Q^+(v)$. Torej je $z \in N_Q^+(v)$, torej $(v, z) \in Q$. Ker velja tudi $(z, w) \in R$, je $(v, w) \in QR$, torej $w \in N_{QR}^+(v)$. S tem smo dokazali $N_{QR}^+(u) \subseteq N_{QR}^+(v)$. Z zamenjavo vlog u in v dobimo še vsebovanost v nasprotni smeri. Torej iz $r(u) = r(v)$ sledi $N_{QR}^+(u) = N_{QR}^+(v)$. Analogno dokažemo še $N_{QR}^-(u) = N_{QR}^-(v)$. Torej je r krepko strukturna glede na QR.

□

□

Izrek 7.55 *Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami. Če je $r : V \to W$ regularna dodelitev vlog z grafom vlog $\mathcal{R} = (W, \mathcal{F})$, potem je $r_{rel} : S(\mathcal{G}) \to S(\mathcal{R})$ surjektiven homomorfizem polgrup.*

Dokaz. Iz izreka 7.54 vemo, da je r regularna glede na poljubno relacijo iz $S(\mathcal{G})$. Potem iz leme 7.53 sledi $r_{rel}(QR) = r_{rel}(Q)r_{rel}(R)$ za poljubni relaciji $Q, R \in S(\mathcal{G})$, torej je r_{rel} homomorfizem. Vzemimo poljuben $F \in \mathcal{F}$. Potem obstaja $E \in \mathcal{E}$, za katerega velja $F = \{(r(u), r(v)) \mid (u, v) \in E\}$. Torej je $F = r_{rel}(E)$. Sledi, da je r_{rel} surjektivna. □

□

Izrek 7.56 *Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami. Če je $r : V \to W$ krepko strukturna dodelitev vlog z grafom vlog $\mathcal{R} = (W, \mathcal{F})$, potem je $r_{rel} : S(\mathcal{G}) \to S(\mathcal{R})$ izomorfizem polgrup.*

Dokaz. Najprej dokažimo, da je vsaka krepko strukturna dodelitev vlog tudi regularna. Naj bo r krepko strukturna. Vzemimo poljubni vozlišči $u, v \in V$, za kateri velja $r(u) = r(v)$. Potem je $N^+(u) = N^+(v)$ in $N^-(u) = N^-(v)$. Sledi $r(N^+(u)) = r(N^+(v))$ in $r(N^-(u)) = r(N^-(v))$, torej je r regularna. Po izreku 7.55 je r_{rel} surjektiven homomorfizem polgrup.

Za poljuben $P \in S(\mathcal{G})$ je $r_{rel}(P) = \{(r(u), r(v)) \mid (u, v) \in P\}$. Torej iz $(u, v) \in P$ sledi $(r(u), r(v)) \in r_{rel}(P)$. Dokažimo še obrat. Vzemimo poljubna $u, v \in V$, za katera velja $(r(u), r(v)) \in r_{rel}(P)$. Po definiciji $r_{rel}(P)$ obstajata $w, z \in V$, za katera velja $r(w) = r(u)$, $r(z) = r(v)$ in $(w, z) \in P$. Ker je r krepko strukturna, $r(w) = r(u)$ in $r(z) = r(v)$, velja $N^+(w) = N^+(u)$ in $N^-(z) = N^-(v)$. Ker je $(w, z) \in P$, je $z \in N^+(w) = N^+(u)$, torej $(u, z) \in P$. Potem je $u \in N^-(z) = N^-(v)$, torej $(u, v) \in P$. Torej za poljuben $P \in S(\mathcal{G})$ velja $(u, v) \in P \Leftrightarrow (r(u), r(v)) \in r_{rel}(P)$. Vzemimo sedaj poljubna $Q, R \in S(\mathcal{G})$, za katera velja $r_{rel}(Q) = r_{rel}(R)$. Potem za vsaka $u, v \in V$ velja $(u, v) \in Q \Leftrightarrow (r(u), r(v)) \in r_{rel}(Q) = r_{rel}(R) \Leftrightarrow (u, v) \in R$. Sledi $Q = R$, zato je r_{rel} injektivna. Torej je r_{rel} izomorfizem polgrup. □

□

Ali homomorfizmi polgrup reducirajo omrežja? Zgornji izreki nam dajo idejo za alternativen pristop k iskanju dodelitev vlog. V izreku 7.55 smo pokazali, da dodelitve vlog vpeljejo nove enakosti v polgrupo relacij omrežja. Kaj pa obratno? Ali identifikacija relacij inducira identifikacijo vozlišč v grafu, s katerim smo generirali polgrupo?

Torej: podan je graf z več relacijami $\mathcal{G}$ s polgrupo $S(\mathcal{G})$ in surjektiven homomorfizem polgrup $S(\mathcal{G}) \to S'$, kjer je S' neka polgrupa. Ali obstaja takšen graf z več relacijami $\mathcal{G}'$ in takšen homomorfizem grafov $\mathcal{G} \to \mathcal{G}'$, da je S' polgrupa, generirana z grafom $\mathcal{G}'$?

V splošnem je odgovor ne, saj ni vsaka polgrupa polgrupa relacij. Ostaja pa vprašanje: katere pogoje morata izpolnjevati S' in homomorfizem polgrup, da dobimo smiseln graf vlog in smiselno dodelitev vlog?

7.6.1 Winship-Pattisonova ekvivalenca vlog

Definicija 7.57 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami. Ekvivalenčna relacija $\sim$ na V je *šibka ekvivalenca vlog* grafa $\mathcal{G}$, če za vsak $u, v, w \in V$ in $E \in \mathcal{E}$ iz $u \sim v$ sledi:

- $uRw \Rightarrow \exists x : vRx$,
- $wRu \Rightarrow \exists x : xRv$.

Šibka ekvivalenca vlog se od regularne razlikuje v tem, da ne zahtevamo $x \sim w$. Če ima graf eno samo relacijo, je maksimalna šibka ekvivalenca vlog particija na izolirane točke, ponore, izvore in vozlišča s pozitivno vhodno in izhodno stopnjo.

Šibke ekvivalence vlog so ravno tiste ekvivalenčne relacije, ki so regularne glede na particijo na eno samo množico.

Definicija 7.58 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami in $S := S(\mathcal{G})$ njegova polgrupa. Ekvivalenčna relacija $\sim$ na V je *kompozicijska ekvivalenca vlog* za $\mathcal{G}$, če je šibka ekvivalenca vlog grafa z več relacijami (V, S).

Kompozicijske ekvivalence so torej poseben primer šibkih ekvivalenc vlog.

Definicija 7.59 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami in $C = (V, \mathcal{M}) := MPX(\mathcal{G})$ njegov mnogoteren graf. Ekvivalenčna relacija $\sim$ na V je *sveženjska ekvivalenca vlog* za graf $\mathcal{G}$, če je šibka ekvivalenca vlog grafa C.

Tudi sveženjske ekvivalence so poseben primer šibkih ekvivalenc vlog.

Definicija 7.60 Naj bo $\mathcal{G} = (V, \mathcal{E})$ graf z več relacijami. Ekvivalenčna relacija $\sim$ na V je *lokalna ekvivalenca vlog* ali *Winship-Pattisonova ekvivalenca vlog*, če je sveženjska ekvivalenca vlog za graf več relacijami $(V, S(\mathcal{G}))$.

Lokalne ekvivalence vlog so poseben primer sveženjskih in kompozicijskih ekvivalenc. Lokalne ekvivalence vlog v splošnem niso regularne, torej tudi šibke, kompozicijske in sveženjske ekvivalence vlog v splošnem niso regularne.

Poglavje 8

Omrežne statistike

Ajda Pirnat, Julia Cafnik, Živa Mitar

V tem razdelku bomo podrobneje predstavili mere komplesnih omrežij. To je orodje, ki nam omogoča analiziranje značilnosti velikih in kompleksnih omrežij kot so stopnja vozlišča, razdalja, število najkrajših poti, koeficient gručavosti, koeficient popačenja in tranzitivnost.

8.1 Uvod

Zaradi samega obsega velikih in kompleksnih omrežij je potrebno zmanjšati informacije, ki jih imamo. To naredimo tako, da se bo dalo opisati bistvene značilnosti točk, povezav, območij ali celotnega grafa na čim bolj enostaven način, in sicer s pomočjo mer omrežij. Mere omrežij so neke vrednosti, ki nam povedo nekaj o relevantnih in iskanih informacijah. V tej seminarski nalogi bomo predstavili najpomembnejše omrežne mere. Na podlagi predstavljenih mer bomo razvrstili mere glede na njihov osnovni tip. Pogledali si bomo tudi načine pretvarjanja različnih tipov iz enega v drugega.

8.2 Lastnosti omrežnih statistik

Omrežne mere:

- *Opisujejo bistvene značilnosti omrežij.* To je glavna naloga omrežnih mer. Določena značilnost naj bo opisana v kompaktni in priročni obliki. Ko analiziramo nek graf, bi se radi osredotočili le na določene lastnosti tega grafa s pomočjo omejenega nabora omrežnih mer. Pri tem ni pomembna natančna struktura analiziranega grafa, ampak le vrednosti njegovih omrežnih mer.

- *Razlikujejo med različnimi vrstami omrežij.* Ko analiziramo omrežje, si pogosto postavimo vprašanje, kakšnega tipa je omrežje in kateri model je primeren za generiranje takega omrežja. Med seboj primerjamo opazovan in generiran graf. To

storimo s pomočjo omrežnih mer omrežij, ki jih med seboj primerjamo. Na podlagi teh lastnosti lahko za poljubni graf določimo, kateremu razredu grafov pripada.

- *Se uporabljajo v algoritmih in omrežjih.* Nekatere omrežne mere se uporabljajo v algoritmih ali pa pri izračunih v povezavi z grafom. Glede na aplikacijo, v kateri uporabimo omrežne mere, nam lahko le te povedo nekaj o lastnostih elementov grafa.

  Koliko posamezna mera izpolnjuje te zahtevke je odvisno od aplikacije in samega omrežja. V nadaljevanju si bomo pogledali, katere so bistvene značilnosti posameznih omrežnih mer, njihovo konstrukcijo in aplikacijo.

8.3 Porazdelitev stopenj vozlišč

Najpogostejša in računsko najenostavnejša omrežna mera je *stopnja vozlišča.* V naključnem neusmerjenem grafu $G_{n,p}$, tj. graf na n vozliščih, kjer je povezava med dvema vozliščema v grafu z verjetnostjo p, je pričakovan delež vozlišč stopnje k enak:

$$\binom{n-1}{k} p^k (1-p)^{n-1-k} \quad \text{(Binomska porazdelitev)},$$

kadar je stevilo vozlišč n majhno. Kadar je n zelo velik pa je približno enak:

$$\frac{(np)^k}{k!} e^{-np} \quad \text{(Poissonova porazdelitev)}.$$

To sledi iz dejstva, da binomska porazdelitev konvergira k poissonovi, ko gre $n \to \infty$ in je $np \approx \lambda$, kjer je λ neka konstanta in p zelo majhna verjetnost.

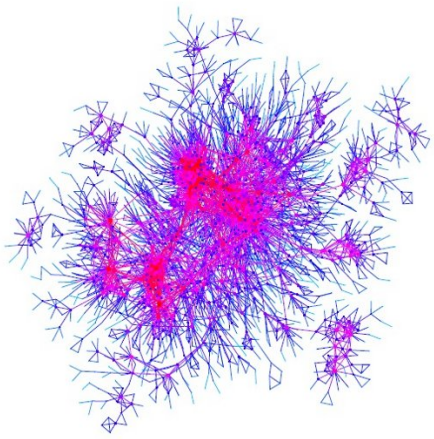
Če pogledamo grafe iz realnosti, imajo le ti največkrat stopnjo vozlišča porazdeljeno po potenčnem zakonu, tj.

$$ck^{-\gamma}, \text{ kjer je } \gamma > 0 \text{ in } c > 0.$$

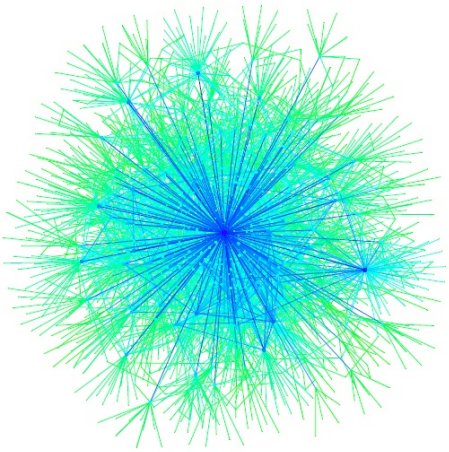
Zanimivo je, da je porazdelitev stopnje vozlišča v tem primeru neodvisna od velikosti grafa, torej od števila vozlišč. Ponavadi velja, da je $2 < \gamma < 3$, obstajajo pa tudi izjeme.

V nadaljevanju si poglejmo nekaj primerov grafov, ki imajo stopnjo vozlišča porazdeljeno po potenčnem zakonu:

- *Električno omrežje ZDA*, kjer je $\gamma \approx 4$.
- *Internet*(ruterji in avtonomni sistemi), kjer je $\gamma \approx 2,2$.
- *Graf sodelovanj med igralci.* To je graf, kjer so igralci vozlišča. Med dvema igralcema je povezava, če sta igrala v istem filmu. Ta graf ima $\gamma \approx 2.3$. Vsebuje $225,000$ vozlišč in približno 13 milijon povezav. Na sliki 8.1 imamo primer takega grafa.
- *Graf sodelovanj med matematiki:* Ta graf ima podobno osnovo kot graf sodelovanj med igralci. Ima približno $401,000$ vozlišč, kjer so vozlišča pisci matematičnih člankov. Dva matematika sta povezana, če sta napisala skupaj članek. Članek, ki ga je napisalo pet avtorjev doprinese deset povezav. Maksimalna stopnja grafa sodelovanj med matematiki je 1416, kar je število soavtorjev člankov Paula Erdősa. Vsi ti soavtorji imajo Erdősovo število 0. Vsak kdo, ki je napisal članek skupaj z avtorjem, ki ima Erdősovo število 1, ima Erdősovo število 2 itd. Maksimalno Erdősovo število je 13. Ta graf ima $\gamma \approx 2.4$. Na sliki 8.2 imamo primer takega grafa.



Slika 8.1: *Graf sodelovanj med igralci*, ki ima približno $10,000$ vozlišč.



Slika 8.2: *Graf sodelovanj med matematiki*

- *Socialna omrežja:* Sem se uvrščajo grafi socialnih omrežij kot so Facebook, MySpace, Twitter, Yahoo! Instant messaging, MSN etc., ki imajo praviloma $2 < \gamma < 3$. Na sliki 8.3 imamo primer takega grafa.

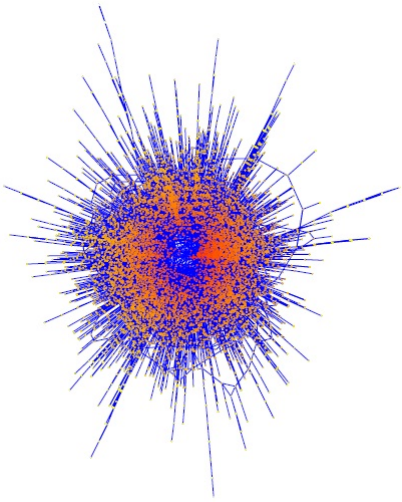
8.4 Razdalja

Razdalja je tudi ena izmed osnovnih omrežnih mer, vendar je računsko bolj zahtevna od stopnje vozlišča. V tem razdelku se bomo osredotočili na neusmerjene grafe. Naj bo razdalja med dvema poljubnima točkama u in v definirana kot $d(u,v) = \min\{|P| \mid$ P je najkrajša pot od u do v$\}$.

Če te razdalje razporedimo v matriko D, dobimo matriko velikosti $|V| \times |V|$, ki je simetrična in ji pravimo *matrika razdalje*. Vrstice in stolpci v tej matriki so indeksirani z vozlišči grafa, tako da velja:

$$D = (d(u,v))_{u,v \in V},$$

kjer nam vrstica u in stolpec v povesta razdaljo med vozlišči u in v.



Slika 8.3: *Graf Yahoo! Instant messaging grafa* To je primer grafa socialnega omrežja, ki ima 29,000 vozlišč in 39,000 povezav.

Za poljubne uteži na povezavah $\omega : E \to \mathbb{R}$, je problem iskanja najkrajše poti $\mathcal{NP}$-težak problem. Če se omejimo le na določene vrste uteži (v realnosti bolj pogost primer), se lahko problem iskanja najkrajše poti reši v polinomskem času.

8.4.1 Povprečna ali karakteristična razdalja

Povprečna ali karakteristična razdalja $\overline{d}$ je aritmetrična sredina vseh razdalj v grafu, tj.

$$\overline{d} = \frac{1}{|V^2| - |V|} \sum_{u \neq v \in V} d(u, v).$$

Če imamo nepovezan graf, je $\overline{d} = \infty$. Če poznamo matriko razdalje D, potem lahko izračunamo povprečno razdaljo v $O(n^2)$ času.

8.4.2 Radij, diameter in ekscentričnost

Če fiksiramo argument v razdalji (to pomeni, da fiksiramo določeno vozlišče) se število parametrov za to mero zmanjša. Temu pravimo *ekscentričnost* $\epsilon(u)$ vozlišča u. To je maksimalna razdalja med u in ostalimi vozlišči, tj.

$$\epsilon(u) := \max\{d(u, v) \mid v \in V\}.$$

Radij $\mathrm{rad}(G)$ je minimalna ekscentričnost vseh vozlišč, tj.

$$\mathrm{rad}(G) = \min\{\epsilon(u) \mid u \in V\}.$$

Če maksimiziramo razdaljo po obeh argumentih, dobimo *diameter* $\mathrm{diam}(G)$ grafa, tj. maksimalna razdalja med dvema poljubnima, povezanima povezavama:

$$\mathrm{diam}(G) = \max\{d(u, v) \mid u, v \in V\}.$$

Tako kot pri povprečni razdalji, lahko s pomočjo matrike razdalje D izračunamo ekscentričnost in diameter grafa v $O(n^2)$ času. Ekscentričnost posamezne točke lahko izračunamo tudi s pomočjo SSSP algoritma, ki ga bomo spoznali kasneje.

8.4.3 Okolice točk

Množici vseh točk u, ki ležijo na razdalji manjši oziroma enaki h glede na vozlišče v, pravimo *h-okolica* vozlišča v in označimo z $\text{Neigh}_h(v)$, tj.

$$\text{Neigh}_h(v) := \{u \in V \mid d(u,v) \leq h\}.$$

Velikost h-okolice označimo z:

$$N(v,h) = |\,\text{Neigh}_h(v)|.$$

Hop plot $P(h)$ nam pove, koliko je parov vozlišč (u,v), med katerima je razdalja manjša ali enaka h, tj.

$$P(h) = |\{(u,v) \in V^2 \mid d(u,v \leq h)\}| = \sum_{v \in V} N(v,h).$$

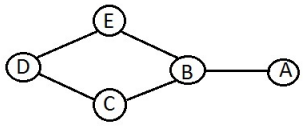
Relativni hop plot $p(h)$ nam pove delež vseh parov vozlišč v grafu, ki imajo razdaljo manjšo ali enako h:

$$p(h) = \frac{P(h)}{n^2} = \frac{1}{n^2} \sum_{v \in V} N(v,h).$$

Povprečno velikost h-okolice označimo z $\overline{\text{Neigh}}(h)$ in je definirana kot:

$$\overline{\text{Neigh}}(h) = \frac{1}{n} \sum_{v \in V} N(v,h) = \frac{P(h)}{n} = np(h).$$

Tudi v tem primeru lahko vrednost hop plot-a, relativnega hop plot-a in velikosti okolic $N(v,h)$ izračunamo s pomočjo matrike razdalje v $O(n^2)$ času. Vrednost hop plot-a interneta je raziskoval matematik Faloutsos. Ugotovil je, da je porazdeljen po potenčnem zakonu z $\gamma \approx 4.7$.



Slika 8.4: Imamo graf na petih točkah, $V = \{A, B, C, D, E\}$. Določimo vrednosti zgoraj navedenih mer razdalje: $\epsilon(B) = 2$, $\text{rad}(G) = 2$, $\text{diam}(G) = 3$, $\text{Neigh}_2(A) = \{A, B, C, E\}$, $N(A,2) = 4$, $P(h) = 10$, $p(h) = 10/25$, $\overline{\text{Neigh}}(2) = 10/5 = 2$.

8.4.4 Efektivna ekscentričnost in diameter

Efektivno eksentričnost ϵ_{eff} in *efektivni diameter* diam_{eff} izračunamo preko ekscentričnosti oz. diametra grafa. Efektivna ekscentričnost meri minimalno razdaljo pri kateri leži določen delež r vseh vozlišč od določenega izvora v, tj.

$$\epsilon_{eff}(v,r) = \min\{h \mid N(v,h) \geq rn\}.$$

Naloga efektivnega diametra je podobna, le da nima določenga izvora:

$$\text{diam}_{eff}(r) = \min\{h \mid P(h) \geq rn^2\} = \min\{h \mid p(h) \geq r\}.$$

Če poznamo N in P lahko obe omrežni meri izračunamo v $O(\log \text{diam}(G))$ času.

8.4.5 Algoritmi

SSSP (angl. Single-Source Shortest Paths) problem: Če rešimo problem SSSP za vsako vozlišče posebej, dobimo matriko razdalje D. Glede na uteži, ki jih imamo v grafu, poznamo različne algoritme za iskanje najkrajše poti:

- Če je graf neutežen, tj. $\omega(e) = 1$ za vsa vozlišča $e \in E$, potem rešimo SSSP preko iskanja v širino (angl. Breadth-first-search) v $O(n + m)$ času.
- Če imamo graf z nenegativnimi utežmi, potem rešimo SSSP preko Dijkstrovega algoritma v $O(n \log n + m)$ času.
- Če graf G nima negativnih ciklov, potem rešimo SSSP s pomočjo Bellman-Fordovega algoritma v $O(nm)$ času.

APSP (angl. All-Pairs Shortest Paths) problem: Pogoj, da lahko rešimo ta problem, je zopet predpostavka, da imamo graf z nenegativnimi cikli. Ta problem rešimo s pomočjo Floyd-Warshallovega algoritma v $O(n^3)$ času.

Drugi pristop za reševanje APSP problema je, da rešimo SSSP problem za vsako vozlišče grafa posebej. V tem primeru se računska zahtevnost spremeni:

- neutežen graf: $O(nm)$
- graf z nenegativnimi utežmi: $O(nm + n^2 \log(n))$
- graf brez negativnih ciklov: $O(n^2 m)$.

Izboljšano rešitev APSP za utežene grafe brez negativnih ciklov se doseže z Johnsonovim algoritmom. Deluje tako, da sprva izračuna razadalje iz naključno izbranega vozlišča do vseh ostalih vozlišč v grafu s pomočjo Bellman-Fordovega algoritma. Če graf ne vsebuje negativnih ciklov, algoritem ponovno izračuna težo povezav s pomočjo rezultatov Bellman-Fordovega algoritma in nato določi razdaljo med vsemi pari vozlišč s pomočjo Dijkstrovega algoritma, n-krat. Če graf vsebuje negativne cikle, se algoritem konča. Čas, ki ga za to porabi, je $O(n^2 \log n + nm)$.

8.5 Število najkrajših poti

Statistika povezana z razdaljo je *število različnih najkrajših poti med vozliščema u in v.* Definirana je kot

$$c(u, v) := |\{P \mid P \text{ je najkrajša pot od } u \text{ do } v\}|.$$

Ker je nasplošno APSP problem NP-težek, očitno tudi štetje teh poti ne more biti lažje. Enako kot prej pa obstajajo poenostavitve: če graf ne vsebuje negativnega ali ničelnega cikla, potem je problem rešljiv v polinomskem času. Vrednost $c(u, v)$ lahko v takih grafih izračunamo z modificiranim Floyd-Warshallovim ali Dijkstrinim algoritmom. Slednjega uporabimo lahko le v primeru, ko so uteži vseh povezav pozitivne.

Modificiran Floyd-Warshall algoritem postopa enako kot prej, le da so dodani določeni koraki. Pri inicializaciji dodatno definiramo še $c(u, v)$, ki bo števec najkrajših poti. $c(u, v)$ se spremeni, če pri preverjanju dodatnega vozlišča odkrijemo različico najkrajše poti (prva *if*-zanka) oz. novo najkrajšo pot (druga *if*-zanka).

Algorithm 14 Štetje različnih najkrajših pot

Vhod: Z utežmi w utežen graf $G = (V, E)$, kateri je brez ciklov negativne in ničelne dolžine.

Izhod: Razdalja $d(u, v)$ in število različnih najkrajših poti med vsakim parom vozlišč uv.

for all v ∈ V **do**
for all $u \in V$ **do**

$$d(u,v) = \begin{cases} 0 & \text{če } u = v \\ w(u,v) & \text{če } uv \in E \\ \infty & \text{sicer} \end{cases}$$

$$c(u,v) = \begin{cases} 1 & \text{če } uv \in E \\ 0 & \text{sicer} \end{cases}$$

end for
end for
for all v' ∈ V **do**
for all u ∈ V **do**
for all v ∈ V **do**
if d(u,v') + d(v',v) = d(u,v) **then**
c(u,v) = c(u,v) + c(u,v')c(v',v) (Najdemo pot enake dolžine.)
else
if $d(u, v') + d(v', v) < d(u, v)$ **then**
d(u,v) = d(u,v') + d(v',v)
c(u,v) = c(u,v')c(v',v) (Najdemo krajšo pot.)
end if
end if
end for
end for
end for

Lahko pa nas zanima *število različnih najkrajših poti v grafu, ki vsebujejo določeno povezavo e:*

$$c(e) := |\{P|P \text{ je najkrajša pot, ki vsebuje povezavo } e\}|.$$

Naj bo $e = uv \in E$. Potem v neuteženem grafu velja:

$$c(e) = \sum_{d(u',v')=d(u',u)+1+d(v,v')} c(u',u)c(v,v').$$

8.6 Koeficient popačenja

Naj imamo utežen graf $G = (V, E)$ in njemu vpeto drevo T. *Koeficient popačenja* (ang. distortion) nam pove povprečno dolžino (ceno) poti v drevesu T med dvema sosednjima vozliščema grafa G in je definiran kot

$$D(T) := \frac{1}{|E|} \sum_{uv \in E} d_T(u, v),$$

kjer je $d_T(u, v)$ pot med vozliščema u in v v drevesu T.

Globalni koeficient popačenja $D(G)$ grafa G je najmanjši koeficient popačenja izmed vseh koeficientov popačenja grafu G vpetih dreves:

$$D(G) := \min\{D(T) \mid T \text{ je grafu } G \text{ vpeto drevo}\}.$$

8.7 Koeficient gručavosti in tranzitivnost

Koeficient gručavosti leta 1998 definirata Watts in Strogatz. Za izbrano vozlišče v grafa G predstavlja verjetje (ang. likeliness), da sta poljubni dve sosednji vozlišči tega izbranega vozlišča povezana. Označimo ga s $c(v)$. Statistika $C(G)$ pa je povprečna vrednost $c(v)$ za vsa vozlišča grafa G.

Za to povprečje $C(G)$ leta 2000 Barrat in Weigt trdita, da sta našla njegovo alternativno formulacijo. To se leta 2002, ko Newmann, Strogatz in Watts upeljejo pojem tranzitivnosti, izkaže za neresnično. Tranzitivnost je namreč ekvivalentna koeficientu gručavosti $C(G)$, ki sta ga definirala Barrat in Weigt, ne pa koeficientu gručavosti $C(G)$ iz njegove prve definicije (iz leta 1998).

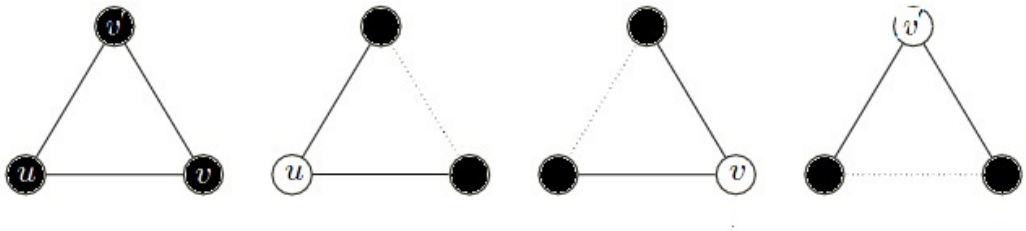
8.7.1 Osnovni pojmi in definicije

Cikel dožine tri (trikotnik) $\triangle = (V_\triangle, E_\triangle)$ je poln podgraf grafa G z natančno tremi vozlišči. Število različnih takih podgrafov v grafu G označimo z $\lambda(G)$. Ekvivalentno označimo z $\lambda(v) := |\{\triangle \mid v \in V_\triangle\}|$ število trikotnikov, ki se dotikajo vozlišča v. Opazimo: $\lambda(G) = \frac{1}{3}\sum_{v \in V} \lambda(v)$.

Drugi pojem, ki ga bomo definirali v tem razdelku je *triada*. Triado tvorita dve sosednji povezavi. *Triada v vozlišču v* je triada, kjer je vozlišče, katero jo skupno obema povezavama, vozlišče v.

Število triad v vozlišču v označimo s $\tau(v)$. Če z $d(v)$ označimo stopnjo vozlišča v, potem velja:

$$\tau(v) := \binom{d(v)}{2}.$$



Slika 8.5: *Cikel dolžine tri in triade.* Prva je triada v u, druga v v, tretja pa v v'.

Število $\tau(G) := \sum_{v \in V} \tau(v)$ pa nam pove število triad v grafu.
V okviru teh definicij *koeficient gručavosti za vozlišce* v definiramo kot

$$c(v) = \frac{\lambda(v)}{\tau(v)},$$

če je le $\tau(v)$ različen od 0. $c(v)$ vozlišča v, ki je stopnje manj kot 2, je lahko enako 1 ali 0, odvisno od dogovora. Temu v izogib definiramo množico vozlišč $V' = \{v | v \in V \text{ in } d(v) \geq 2\}$.

Koeficient gručavosti za celoten graf nato definiramo kot:

$$C(G) := \frac{1}{|V'|} \sum_{v \in V'} c(v).$$

Sedaj nam preostane le še definicija pojma tranzitivnost. *Tranzitivnost* se definira samo na ravni grafa:

$$t(G) := \frac{3\lambda(G)}{\tau(G)}.$$

Ker vsak cikel dolžine tri vsebuje točno tri triade ($3\lambda(G) \leq \tau(G)$), je tranzitivnost vedno racionalno število med 0 in 1.

8.7.2 Odnos med tranzitivnostjo in koeficientom gručavosti

Tranzitivnost $t(G)$ je bila najprej mišljena kot alternativna formulacija koeficienta gručavosti $C(G)$. Omenjeni količini se namreč ne razlikujeta, če so vsa vozlišča iste stopnje oz. če imajo vsa vozlišča enak $c(v)$. Bollobás in Riordan sta pokazala, da v splošnem med omenjenima količinama velja

$$t(G) = \frac{\sum_{v \in V'} \tau(v) c(v)}{\sum_{v \in V'} \tau(v)}.$$

Iz formule vidimo, da je tranzitivnost s številom triad utežen koeficient gručavosti. Pri tranzitivnosti so medsebojno enakovredni vsi cikli dolžine tri, pri koeficientu gručavosti pa so enakovredna vozlišča. Koeficienta se razlikujeta zato, ker so vozlišča večje stopnje verjetneje del večih ciklov dolžine tri, kot pa vozlišča manjše stopnje.

8.7.3 Izračun koeficineta gručavosti in tranzitivnosti

Da bi izračunali koeficinet gručavosti potrebujemo vrednosti $\lambda(v)$ in in $\tau(v)$ za vsako vozlišče grafa. Za izračun tranzitivnosti pa potrebujemo $\tau(v)$ za vsako vozlišče v in število trikotnikov v celotnem grafu $\lambda(G)$.

Izračun števila triad je enostaven in ima linearno časovno zahtevnost. Večji problem nastane pri izračunu števila ciklov dolžine tri.

Standardna metoda je iteracija po vozliščih (ang. node iterator) in preverjanje, ali so sosednja vozlišča izbranega vozlišča povezana ali ne. Časovna zahtevnost tega postopka je $O(d_{\max}^2 n)$, kjer je $d_{\max} = \max\{d(v) \mid v \in V\}$.

Algorithm 15 "node-iterator"

Vhod: Graf $G = (V, E)$, vozlišča poljubno urejena nad "$<$"
Izhod: Cikli dolžine tri grafa G

```
for v ∈ V do
    for vsak par sosedov u, w vozlišča v do
        if uw ∈ E then
            if u < v < w then
                output triangle {u, v, w}
            end if
        end if
    end for
end for
```

Drugi pristop je uporaba množenja matrik. Če je A matrika sosednjosti za graf G, bo potem diagonala A^3 vsebovala dvakratnik števila ciklov dolžine tri pripadajočega vozlišča. Toko dobimo časovno zahtevnost reda $O(n^3)$. Hitrost je možno dodatno izboljšati z uporabo hitrega množenja matrik in tako je časovna zahtevnost reda $O(n^\gamma)$, kjer je γ koeficient matričnega množenja. Zaenkrat je znano, da je $\gamma \leq 2,376$.

Kombinacija obeh pristopov pa je *algoritem AYZ* (Alon, Yuster, Zwick algoritm). Algoritem poteka tako, da najprej loči vozlišča na dva tipa. Taka z nizko stopnjo ($V_{low} = \{v \in V \mid d(v) \leq \beta\}$) in taka z višjo stopnjo ($V_{high} = V \backslash V_{low}$), kjer je $\beta = m^{\frac{\gamma-1}{\gamma+1}}$ in $m = |E|$. Na vozliščih z nižjo stopnjo išče cikle dolžine tri z "node-iteratorjem", na podgrafu induciranem z vozlišči višje stopnje pa s hitrim matričnim množenjem.

Algoritem AYZ za vsako vozlišče izračuna število ciklov dolžine tri $\lambda(v)$ in ga lahko implementiramo tako, da je njegova časovna zahtevnost reda $O(m^{\frac{2\gamma}{\gamma+1}})$ oz. $O(m^{1.41})$.

8.7.4 Aproksimacija števila ciklov dolžine tri

Če imamo opravka z zelo velikimi omrežji, potem je zaželjena linearna ali sublinearna časovna zahtevnost. To lahko dosežemo z uporabo slučajnega vzorčenja.

Naj bo X_i omejena slučajna spremenljivka z domeno realnih števil med 0 in M za vsak i. Naj bo k število vzorcev. Potem velja Hoeffdingova neenakost:

$$Pr\Big(\Big|\frac{1}{k}\sum_{i=1}^{k} X_i - E(\frac{1}{k}\sum_{i=1}^{k} X_i)\Big| \geq \epsilon\Big) \leq e^{\frac{-2k\epsilon^2}{M^2}}.$$

V zgornji enačbi definiramo izraz na desni strani neenačaja kot verjetnost pravilnosti $p := e^{\frac{-2k\epsilon^2}{M^2}}$, ϵ pa bomo imenovali meja napake.

Algorithm 16 AYZ

Vhod: Graf $G = (V, E)$, parameter matričnega množenja γ
Izhod: Število trikotnikov $\lambda(v)$ za vsako vozlišče

```
β = m^((γ−1)/(γ+1))
for v ∈ V do
    λ(v) = 0
    if d(v) ≤ β then
        t V_low = V_low U {v}
    else
        V_high = V_high U {v}
    end if
    for v ∈ V_low do
        for vsak par sosedov {u,w} vozlišča v do
            if povezava med u in v obstaja then
                if u, w ∈ V_low then
                    for z ∈ {v, u, w} do
                        λ(z) = λ(z) + 1/3
                    end for
                end if
            else if u, w ∈ V_high then
                for z ∈ {v, u, w} do
                    λ(z) = λ(z) + 1
                end for
            else
                for z ∈ {v, u, w} do
                    λ(z) = λ(z) + 1/2
                end for
            end if
        end for
    end for
end for
A = matrika sosednosti na induciranem podgrafu V_high
M = A^3
for for v ∈ V_high do do
    λ(v) = λ(v) + M(i,i)/2, kjer je i indeks vozlišča v
end for
```

Hoeffdingova neenakost nam pove, koliko vzorcev k rabimo, da se z verjetnostjo manjšo kot je verjetnost pravilnosti, povprečna vrednost opazovane spremenljivke ($\frac{1}{k}\sum_{i=1}^{k} X_i$) in upanje zanjo ($E(\frac{1}{k}\sum_{i=1}^{k} X_i)$) razlikujeta za več kot ϵ. Verjetnost pravilnosti je ponavadi 0.01.

Če v Hoeffdingovi neenakosti fiksiramo ϵ in p, potem, kot že rečeno, dobimo število k. Če ocenjujemo koeficient gručavosti $c(v)$ za vsako vozlišče v, potem se k nanaša na število sosednjih parov vozlišča v, za katere nas zanima, ali so povezani ali ne. Če ocenjujemo koeficient gručavosti za celoten graf, torej $C(G)$, potem pa se k lahko nanaša na število vozlišč, ki bodo vključena v izračun (V tem primeru $c(v)$-je bodisi poznamo ali pa jih že prej ocenimo.) ali pa število triad, ki jih bomo upoštevali. Slednji postopek (izbor k triad) velja tudi za tranzitivnost $T(G)$.

Za vsak konstanten ϵ in konstatno verjetnost pravilnosti p, obstaja algoritem, ki oceni koeficient gručavosti $c(v)$ za vsako vozlišče in koeficient tranzitivnosti $T(G)$ za celoten graf v času $O(n)$. Koeficient gručavosti $C(G)$ lahko ocenimo v $O(1)$.

8.8 Omrežni motivi

Pri kategorizaciji mrež si večkrat pomagamo tudi s podgrafi, ki jih lahko najdemo v njej. Pomembni tipi podgrafov so cikli, drevesa in klike (polni podgrafi). Mrežni motivi so podgrafi, ki se v realni mreži pojavljajo pogosteje, kot je statistično pričakovati. Da bi našli motiv dane mreže, potrebujemo verjetnostno porazdelitev pojavljanja posameznih podgrafov v naključnih mrežah, ki imajo enako stopnjo vseh vozlišč kot dana mreža. Nato primerjamo število pojavitev posameznega podgrafa v dani mreži s pričakovanim številom pojavitev v izboru naključnih mrež. Če je verjetnost, da se podgraf v naključni mreži pojavi vsaj tolikokrat kot v realni mreži, manjša od neke predpisane meje (običajno 0.01), potem je ta podgraf motiv dane mreže.

Na tem mestu se pojavi vprašanje, kako prešteti izbrane podgrafe danega grafa. Najbolj preprosto iskanje podgrafa s k vozlišči je, da preverimo vseh $\binom{n}{k}$ možnosti. Potem imamo $k!$ možnosti, da povemo, na katerem mestu v podgrafu je izbrano vozlišce in nato na vsakem od teh korakov še preverjanje vseh potencialnih $2\binom{k}{2}$ povezav.

Za cikle dolzine tri uporabimo že prej omenjeni algoritem AYZ, ki ga lahko le malo modificiramo in tako dobimo algoritem za iskanje podgrafov s tremi vozlišči. Njegova asimptotična časovna zahtevnost ostane nespremenjena.

Prav tako obstaja modificiran AYZ algoritem za iskanje K_4 podgrafov, ki so ga razvili Kloks, Kratsch in Müller. Ta ima časovno zahtevnost $O(m^{\frac{\gamma+1}{2}})$, kjer je $m = |E|$. Še več, na neusmerjenih grafih obstaja tudi algoritem za iskanje podgrafov z največ štirimi vozlišči, ki ima časovno zahtevnost $O(n^{\gamma} + m^{\frac{\gamma+1}{2}})$. Zaenkrat ga še niso uspeli prirediti za uporabo na usmerjenih grafih v okviru iste časovne zahtevnosti.

8.9 Vrste omrežnih statistik

S pomočjo prej predstavljenimi primeri statistik lahko identificiramo 4 osnovne tipe mer, ki jih opišemo z dvema paroma izključujočih atributov. To sta ena vrednost/porazdelitev ter globalno/lokalno.

Kljub temu da so zgoraj omenjeni atributi intuitivno jasni, bomo v nadaljevanju podali formalne definicije štirih vrst omrežnih mer.

Naj bo $\mathcal{G}$ razredov grafov (npr. uteženi, usmerjeni, povezani grafi), imejmo množico parametrov P in množico vrednosti Y. Največkrat sta množici P in Y enaki naravnim, celim ali realnim številom. Nadalje naj bo X_G množica elementov grafa G, katera lahko vsebuje vozlišča, povezave, podgrafe, poti itd.

Definicija 8.1 *Globalna mera* γ označuje (eno samo) vrednost $\gamma_G \in Y$ za vsak graf $G \in \mathcal{G}$.

Zgled 8.2 Število vozlišč/povezav, diameter (poglavje 8.4.2), koeficient gručavosti na grafu (poglavje 8.7).

Definicija 8.3 *Globalna porazdelitev* Γ označuje preslikavo $\Gamma_G : P \to Y$ za vsak graf $G \in \mathcal{G}$.

Običajno označujemo vrednosti porazdelitve z $\Gamma_G(t)$, kjer je t parameter. V primeru $P = \mathbb{N}$ je potem globalna porazdelitev Γ_G enaka zaporedju $(\Gamma_G(0), \Gamma_G(1), \ldots)$.

Zgled 8.4 Porazdelitve vhodne/izhodne stopnje (poglavje 8.3), hop plot (poglavje 8.4.3).

Zgornji statistiki pa lahko pogledamo tudi na lokalnem nivoju.

Definicija 8.5 *Lokalna mera* λ_G označuje (eno samo) vrednost $\lambda_G(x) \in Y$ za vsak graf $G \in \mathcal{G}$, kjer je $x \in X_G$ določen element grafa G. Bolj natančno, gre za preslikavo $\lambda_G : X_G \to Y$.

Zgled 8.6 Vhodna/izhodna stopnja vozlišča, utež oz. kapaciteta povezave, razdalja (poglavje 8.4), koeficient gručavosti za vozlišče (poglavje 8.7).

Definicija 8.7 *Lokalna porazdelitev* Λ označuje preslikavo $\Lambda_G : X_G \times P \to Y$ za vsak graf $G \in \mathcal{G}$ in je X_G množica elementov grafa G.

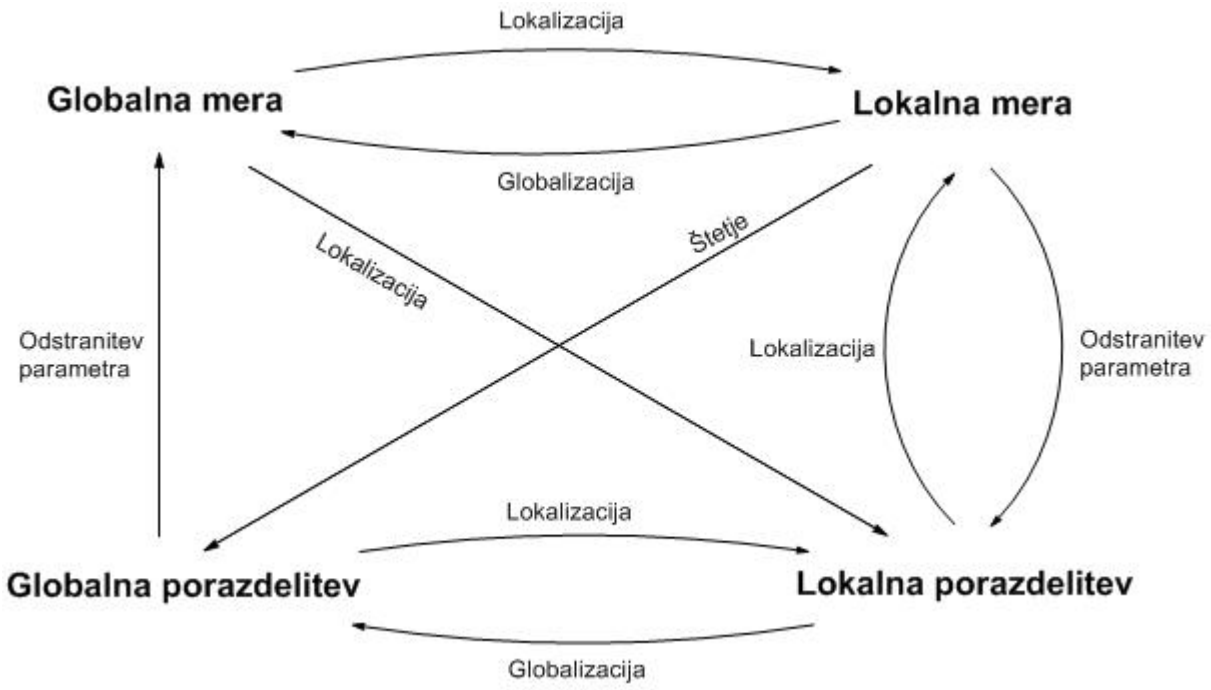
Kot v globalnem primeru lahko označimo vrednost porazdelitve z $\Lambda_G(x, t)$, kjer je x element grafa in t parameter.

Zgled 8.8 Velikost okolice (poglavje 8.4.3), efektivna ekscentričnost in diameter (poglavje 8.4.2).

8.9.1 Transformacije omrežnih statistik

Zgoraj definirane vrste omrežnih mer pa niso povsem neodvisne med seboj, saj obstajajo načini, s katerimi pridemo iz ene vrste mere v drugo. Shema na sliki 8.6 prikazuje možne transformacije iz določene mere v drugo.

V nadaljevanju predstavljamo "uradni" opis posameznih transformacij. O sami intuiciji v ozdaju posamezne tehnike ne bomo razpravljali, saj je to odvisno od konkretnega problema in vrednosti.



Slika 8.6: Transformacije vrst omrežnih mer

Globalizacija

Globalizacija je tehnika, s pomočjo katere pridemo iz lokalne mere/porazdelitve v globalno mero/porazdelitev. Kot že samo ime pove, gre pri tej transformaciji za odstranitev odvisnosti lokalne mere ali porazdelitve od elementov grafa. Do tega ponavadi pridemo z računanjem, izbiranjem ali s konstrukcijo ene vrednosti γ_G iz vrednosti $\gamma_G(x)$ za vse elemente $x \in X_G$ (v primeru mer). Najbolj splošni primeri so:

- maksimum
$$\gamma_G := \max\{\gamma_G(x) \mid x \in X_G\},$$
- minimum
$$\gamma_G := \min\{\gamma_G(x) \mid x \in X_G\},$$
- vsota
$$\gamma_G := \sum_{x \in X_G} \gamma_G(x),$$
- povprečje
$$\gamma_G := \frac{1}{|X_G|} \sum_{x \in X_G} \gamma_G(x).$$

V primeru porazdelitev je konstrukcija podobna, npr.

$$\Gamma_G(t) := \max\{\Lambda_G(x,t) \mid x \in X_G\}.$$

Predstavljeni splošni primeri jasno nakazujejo, da so primeri za globalizacijo tudi povprečna razdalja (poglavje 8.4), diameter in radij (poglavje 8.4.2), hop plot (poglavje 8.4.3) in globalno popačenje (poglavje 8.6).

Štetje

Če želimo lokalno mero λ preoblikovati v globalno porazdelitev Γ, to storimo tako, da preštejemo število elementov x grafa G, pri čemer $\lambda_G(x)$ leži v določeni množici vrednosti. V primeru diskretne mere (tj. $\lambda_G(x) \in \mathbb{NZ}$), lahko preštejemo absolutno število pojavov vrednosti t na naslednji način

$$\Gamma_G(t) := |\{x \in X_G \mid \lambda_G(x) = t\}|,$$

relativno število pojavov vrednosti t pa izračunamo kot

$$\Gamma_G(t) := \frac{|\{x \in X_G \mid \lambda_G(x) = t\}|}{|X_G|}.$$

Podobno lahko v primeru zvezne mere, tj. $\lambda_G(x) \in \mathbb{R}$, dobimo absolutno ali relativno število elementov, kjer je $\lambda_G(x) \leq t$:

$$\Gamma_G(t) := |\{x \in X_G \mid \lambda_G(x) \leq t\}|$$

ali

$$\Gamma_G(t) := \frac{|\{x \in X_G \mid \lambda_G(x) \leq t\}|}{|X_G|}.$$

Primeri iz prejšnji poglavij, ki ustrezajo tehniki štetja, pa so porazdelitve stopnje vozlišča (poglavje 8.3) ter absolutni in relativni hop plot (poglavje 8.4.3).

Odstranitev in zmanjšanje parametra(ov)

Najbolj pogosto uporabljena transformacija je zagotovo odstranitev parametra, s pomočjo katere lahko pridemo iz porazdelitve v mero (na lokalni ali globalni ravni).

Podobno kot pri preoblikovanju iz lokalne do globalne mere (ali porazdelitve) tudi tukaj izračunamo vrednost λ_G iz zaporedja vrednsoti, danega s porazdelitvijo Λ_G. Za razliko od prej pa tukaj za spremenljivko vzamemo parameter porazdelitve in ne element grafa, kot smo to storili pri globalizaciji.

Primeri so seveda podobni kot pri globalizaciji:

- maksimum
$$\lambda_G(x) := \max\{\Lambda_G(x,t) \mid t \in P\},$$
- minimum
$$\lambda_G(x) := \min\{\Lambda_G(x,t) \mid t \in P\},$$
- vsota
$$\lambda_G(x) := \sum_{t \in P} \Lambda_G(x,t),$$
- povprečje (če je P končna množica)
$$\lambda_G := \frac{1}{|P|} \sum_{t \in P} \Lambda_G(x,t).$$

Zgornji primeri predstavljajo transformacijo iz lokalne porazdelitve v lokalno mero; če pa želimo pridobiti globalno mero iz globalne porazdelitve, pa to storimo na analogen način kot zgoraj, izpustimo le spremenljivko x (mere na globalnem nivoju niso odvisne od elementov grafa). Lep primer za odstranitev parametra je tudi ekscentričnost, predstavljena v poglavju 8.4.2.

Lahko se zgodi, da ima lokalna ali globalna porazdelitev več kot en paramater, mi pa za nadaljnje izračune potrebujemo le en parameter. Le-to potem vodi do postopka, ki ga imenujemo zmanjšanje števila parametrov.

Če ima porazdelitev več parametrov, to pomeni, da je množica parametrov P kartezični produkt dveh množic, npr. $P = P' \times P''$. Porazdelitev z zmanjšanim številom parametrov potem definiramo kot

$$\Lambda'_G(x, t') := \max_{t'' \in P''} \{\Lambda_G(x, t', t'')\}.$$

Lokalizacija

Zadnja transformacija, ki jo obravnava seminarska, je lokalizacija. Na sliki 8.6 lahko vidimo, da se lokalizacija pojavi v štirih pretvorbah, katere imajo podobno idejo. Pri vseh postopkih si najprej izberemo smiselen podgraf (glede na vrsto lokalizacije), nato pa nastavimo enačbo, ki nas pripelje do željenega tipa mere. Poglejmo si vrste lokalizacije bolj podrobno.

(1) *Globalna mera $\gamma \to$ lokalna porazdelitev Λ*

Izberemo podgraf $H(x, t) \subseteq G$ za vsak element $x \in X_G$ in vsak parameter $t \in P$ in nastavimo enačbo

$$\Lambda_G(x, t) := \gamma_{H(x,t)}.$$

Da lahko zgornjo enačbo sploh definiramo, mora biti mera γ definirana na podgrafu $H(x, t)$. Naštejmo dva primera izbire podgrafa $H(x, t)$:

- krogla s polmerom t okoli x, to je podgraf induciran z okolico elementa x $\text{Neigh}_t(x)$ (vsa vozlišča v, za katere velja $d(x, v) \leq t$);
- največji/najmanjši podgraf grafa G, ki vsebuje element x in izpolnjuje določen kriterij, ki je odvisen od paramatera t, npr. največji podgraf med vsemi podgrafi, ki vsebuje x in velja, da je najmanjše število povezav, ki je potrebno, da graf postane nepovezan, večje od t.

(2) *Globalna mera $\gamma \to$ lokalna mera λ*

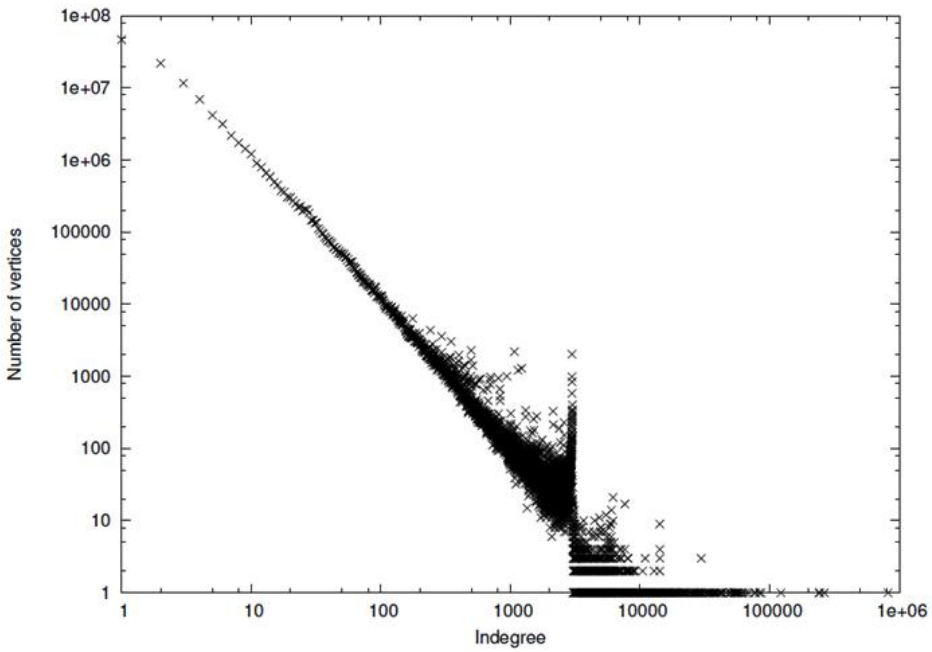
Lokalno mero λ pridobimo iz globalne mere γ tako, da izberemo podgraf $H(x)$, ki je odvisen samo od elementa x, in nastavimo enačbo

$$\lambda_G(x) := \gamma_{H(x)}.$$

Za $H(x)$ lahko na primer izberemo unijo klik [1], ki vsebujejo element x in imajo največjo velikost [2] med vsemi klikami, ki vsebujejo x.

[1] Na grafu $G = (V, E)$ je *klika* definirana kot podmnožica vozlišč $Z \subseteq V$, pri čemer mora veljati, da je vsako vozlišče podmnožice Z na grafu G povezano z vsakim drugim vozliščem iz te podmnožice.

[2] Velikost klike je moč množice Z.



Slika 8.7: Absolutna porazdelitev vhodne stopnje z logaritemsko skalo (omrežje Crawl of Web Base, 2001)

(3) *Globalna porazdelitev* $\Gamma \to$ *lokalna porazdelitev* Λ

Če izberemo podgraf $H(x)$, ki je odvisen le od elementa x, potem enačba

$$\Lambda_G(x,t) := \Gamma_{H(x)}(t)$$

vodi do lokalne porazdelitve Λ.

(4) *Lokalna mera* $\lambda \to$ *lokalna porazdelitev* Λ

Zadnji tip lokalizacije konstruira lokalno porazdelitev iz lokalne mere. Izberemo podgraf $H(x,t)$ za vsak element $x \in X_G$, ki je odvisen tudi od parametra t, in nastavimo enačbo

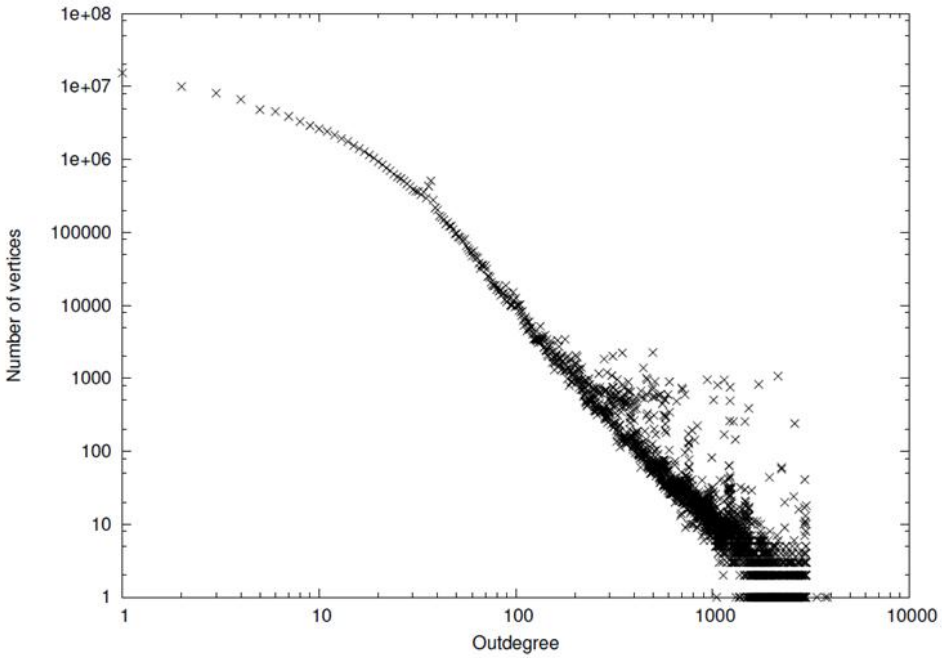
$$\Lambda_G(x,t) := \lambda_{H(x,t)}(x).$$

8.9.2 Vizualizacija

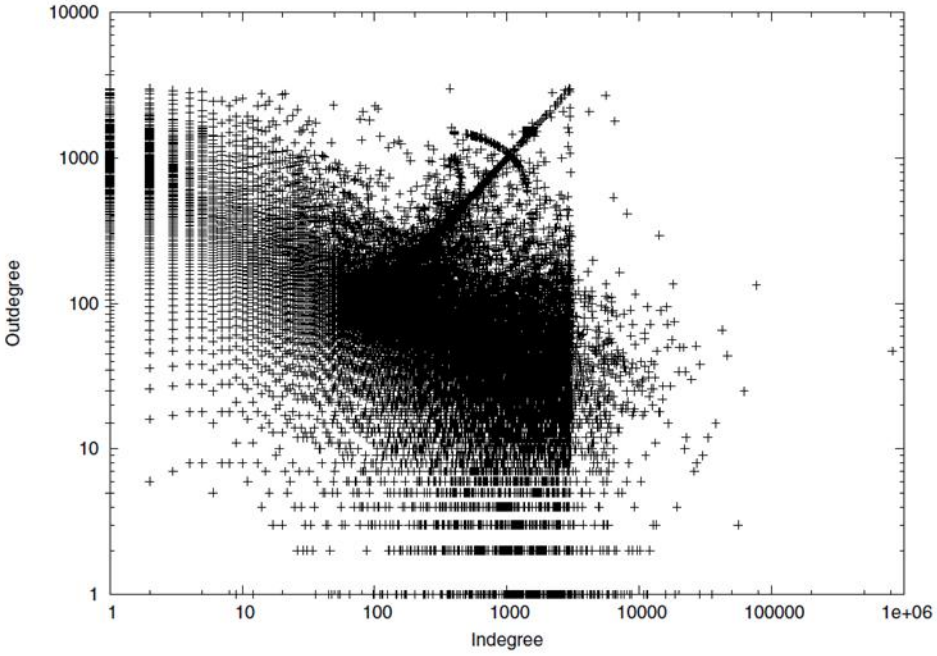
Globalna statistika γ je preprosto ena sama vrednost, torej je njena predstavitev z grafom precej trivialna.

Globalno porazdelitev $\Gamma : P \to Y$, to je preslikava iz množice parametrov v množico vrednosti, lahko predstavimo kot funkcijo na običajen način. Pri tem lahko uporabimo različne tehnike, kot sta npr. interpolacija in regresija, s katerimi dobimo opis grafa funkcije na podlagi danih parametrov. Sliki 8.7 in 8.8 prikazujeta primer vizualizacije za globalno porazdelitev. Na slikah sta prikazani absolutni porazdelitvi vhodne in izhodne stopnji v odvisnosti od števila vozlišč v omrežju WebBase Crawl. Oba diagrama imata logaritemsko skalo in prikazujeta linearno razmerje v potenčnem zakonu (poglavje 8.3).

Za lokalne mere je predstavitev na grafu bolj zapletena. Za lokalno mero $\lambda_G(x)$ izberemo zaporedje $x_1, x_2, \ldots$ elementov grafa G in grafično prikažemo pare $(i, \lambda_G(x_i))$. Vendar



Slika 8.8: Absolutna porazdelitev izhodne stopnje z logaritemsko skalo (omrežje Crawl of Web Base, 2001)



Slika 8.9: Razsevni diagram vhodne in izhodne stopnje z logaritemsko skalo (omrežje Crawl of Web Base, 2001)

je dobljen diagram močno odvisen od izbranega zaporedja, zato bi bil boljši pristop štetje in nato s pomočjo globalne porazdelitve ali mere priti do diagrama.

Različne lokalne mere istega grafa, ki temeljijo na isti množici elementov, lahko predstavimo s pomočjo t.i. razsevnega diagrama. Imejmo dve lokalni meri λ_G in λ'_G na isti množici elementov X_G. Razsevni diagram teh dveh mer je sestavljen iz parov $(\lambda_G(x), \lambda'_G(x))$

za vse elemente $x \in X_G$ in prikazuje razmerje med λ_G in $\lambda_G^{'}$. V primeru, da je dobljena slika nestrukturirani oblak, se zdi, da meri nista povezani; v nasprotnem primeru pa se zdi, da sta meri medsebojno povezani (na nekem danem grafu).

Na sliki 8.9 je prikazan razsevni diagram med vhodno in izhodno stopnjo v omrežju WebBase Crawl na logaritemski skali. Vsak križec predstavlja eno ali več vozlišč, ki imajo ustrezno kombinacijo vhodne in izhodne stopnje. Glede na dobljeni diagram se zdi, da vhodna in izhodna stopnja nista povezani.

8.9.3 Vzorčenje lokalnih mer

Kot smo že videli, so omrežne mere računsko precej zahtevne na velikih grafih, zato se zdi smiselno, da uporabimo približke.

Predpostavimo, da je σ_G lokalna ali globalna mera, pridobljena iz lokalne mere λ_G, ki temelji na elementih $x \in X_G$. Splošno sprejeta tehnika za aproksimacijo σ_G je *vzorčenje*. Temelji na tem, da pri $\lambda_G(x)$ uporabimo samo nekatere vzorce elementov $x \in X_{vzorec} \subset X_G$ za izračun σ_G namesto, da bi vzeli vse elemente $x \in X_G$.

V splošnem ima vzorčenje (globalne) mere σ_G iz lokalne mere $\lambda_G : X_G \to Y$ naslednjo obliko:

1. Izberemo množico vzorcev X_{vzorec} iz množice X_G. To izvedemo popolnoma naključno ali pa s kakšno posebno strategijo.

2. Izračunamo $\lambda_G(x)$ za vsak $x \in X_{vzorec}$.

3. Izračunamo σ_G iz množice $\{\lambda_G(x) \mid x \in X_{vzorec}\}$.

Zgoraj naveden postopek je opisan precej splošno, zato so seveda glede na konkreten problem potrebne in koristne razne prilagoditve. Na primer, sama strategija izbire množice vzorcev pogosto določa kakovost aproksimacije. V vsakem primeru pa je potrebno kakovost aproksimacije potrditi bodisi z eksperimentom ali bodisi analiza na določenih modelih grafov. Očitno je, da naraščanje števila elementov v vzorcu vodi do boljše aproksimacije.

Poglejmo si primer vzorčenja. Hop plot grafa lahko dokaj enostavno ocenimo z uporabo vzorčenja (glej algoritem 20). Algoritem v primeru $l = n$ (n je moč množice X_G) vrne pravo vrednost $\bar{P}$. Časovna zahtevno algoritma je $\mathcal{O}(ln^2 \log n)$, prostorska zahtevnost pa je enaka $\mathcal{O}(n)$ (poleg grafa). Na žalost pa ni znano, kakšna je stopnja napake same aproksimacije.

8.10 Zaključek

Kompleksna omrežja so težko predstavljiva, zato je potrebno skriti vse dane informacije. V ta namen imamo tako imenovane omrežne mere, ki nam opisujejo bistvene značilnosti danega omrežja in s tem poenostavijo njegovo razumevanje.

V tem poglavju smo predstavili nekaj najpomembnejših omrežnih mer, npr. razdalja, radij, diameter, okolice točk, koeficient popačenja ter koeficienta gručavosti in tranzitivnosti. Ogledali smo si tudi algoritem za izraČun števila najkrajših poti v omrežju in algoritem, ki vrne cikle dolžine tri danega grafa. Na koncu poglavja pa smo predstavljene mere razvrstili glede na njihov splošen tip in tako definirali globalno porazdelitev in mero

Algorithm 17 Preprost primer vzorčenja: Hop plot

Vhod: Graf $G = (V, E)$.
Izhod: Aproksimativna vrednost hop plot za graf G.

```
Nastavi P(h) = 0 za h = 0, 1, 2, ..., diam(G)
for i ∈ {1, ..., l} do
    Izberi neuporabljeno vozlišče u_i
    Izračunaj razdaljo d(u_i, v) za vse v ∈ V s pomočjo SSSP za u_i
    for v ∈ V do
        for h ∈ {d(u_i, v), ..., diam(G)} do
            P(h) = P(h) + 1
        end for
    end for
end for
p(h) = P(h)/ln
```

ter lokalno porazdelitev in mero. Razložili smo tudi, kako lahko prehajamo iz ene vrste mer v druge ter si na koncu ogledali grafične prikaze osnovnih tipov mer.

Poglavje 9

Primerjava omrežij

Teja Klemenčič, Jerneja Pikelj, Maja Smrke

V tem seminarskem delu se bomo srečali s primerjavo omrežij/grafov. Podrobneje bosta predstavljena izomorfizem in podobnost grafov. Vprašanje izomorfizma se pogosto pojavlja v matematični kemiji, ko nas zanima kako podobni sta si dve molekularni strukturi (spojini). Prav tako se s tem vprašanjem srečujejo biologi, ko primerjajo biološka omrežja. Uporablja pa se tudi pri elektronski avtomatizaciji načrtovanj, npr. pri načrtovanju elektronskega vezja, pri prepoznavanju simetrije in pri iskanju napak v teksturi.

Podana bomo imeli grafa $G_1(V_1, E_1)$ in $G_2(V_2, E_2)$, kjer sta V_1, V_2 množici točk in E_1, E_2 množici povezav danih grafov. Opravka bomo imeli z grafi brez zank in vzporednih povezav. Takim grafom pravimo enostavni grafi. Temeljno vprašanje, ki se tukaj pojavi je, ali imata dva grafa enako strukturo, torej ali med njima obstaja izomorfizem. Izomorfizem grafov določa ali sta dva grafa, ki na prvi pogled izgledata povsem drugačna, v resnici enaka. V primeru da ugotovimo, da grafa nimata enake strukture, pa bomo želeli podati izjavo o tem, kako oziroma koliko sta si pravzaprav podobna. Opisali bomo tudi nekaj algoritmov za iskanje izomorfizma, ter jih predstavili na primerih. Kot najbolj uporaben in najhitrejši algoritem za iskanje izomorfizma velja omeniti McKay's Nauty algoritem.

9.1 Izomorfizem grafov

Podana imamo enostavna grafa $G_1(V_1, E_1)$ in $G_2(V_2, E_2)$. Zanimalo nas bo, kako podobna sta dana grafa oziroma ali sta morda celo enaka, z drugimi besedami ali sta izomorfna. Za začetek si najprej poglejmo nekaj definicij in trditev [24].

Definicija 9.1 Bijektivni preslikavi $\Phi : V_1 \to V_2$, za katero je za vsaki točki $u, v \in V_1$: $\{u, v\} \in E_1 \Leftrightarrow \{\Phi(u), \Phi(v)\} \in E_2$, pravimo *izomorfizem grafov*.

Definicija 9.2 Grafa G_1 in G_2 sta *izomorfna*, oznaka $G_1 \simeq G_2$, če med njima obstaja kakšen izomorfizem.

Definicija 9.3 V primeru, ko je graf G_2 kar enak grafu G_1, izomorfizmu grafov pravimo *avtomorfizem grafov*. Če množico avtomorfizmov danega grafa G opremimo z operacijo

komponiranja preslikav, dobimo grupo, ki ji pravimo *grupa avtomorfizmov* grafa G in jo označimo z $\mathrm{Aut}(G)$.

Definicija 9.4 Lastnosti grafa, ki jo imajo poleg grafa samega tudi vsi z njim izomorfni grafi, pravimo *invarianta grafa*. Ta lastnost se pri izomorfizmu ohranja.

Preproste grafovske invariante so:

- število točk in povezav
- število podgrafov izbrane vrste (npr. trikotniki, štirikotniki, ...)
- stopnje točk
- število komponent
- dvodelnost
- število mostov

Najenostavnejša in v mnogih primerih najmočnejša invarianta je število točk.

Posledica 9.5 *Izomorfizem ohranja število vozlišč, število povezav, stopnje vozlišč, število trikotnikov, itd.*

Trditev 9.6 *Izomorfnost je ekvivalenčna relacija* ($\simeq$).

Pri iskanju izomorfizma pa si včasih lahko delo poenostavimo, če upoštevamo naslednjo proposition:

Trditev 9.7 *Grafa G_1 in G_2 sta izomorfna natanko tedaj, ko sta izomorfna njuna komplementa G_1^c in G_2^c.*

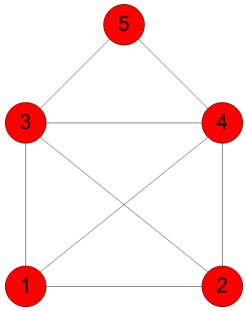
Pri iskanju izomorfizma bomo imeli opravka tudi z matriko sosednosti, zato si poglejmo spodnjo definicijo.

Definicija 9.8 Imamo graf G, množico točk $V(G) = \{v_1, \ldots, v_n\}$ in množico povezav $E(G) = \{e_1, \ldots, e_m\}$. *Matrika sosednosti* $A(G)$ neusmerjenega grafa je kvadratna matrika velikosti $n \times n$, definirana takole:

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix},$$

kjer je

$$a_{ij} = \begin{cases} 1 & ; \text{če } v_i \sim v_j, \\ 0 & ; \text{sicer.} \end{cases}$$



Torej, če sta točki v grafu sosedni, imamo v matriki vrednost 1, sicer pa 0. Iz matrike sosednosti zlahka rekonstruiramo pripadajoči graf. Za neusmerjen graf je matrika sosednosti simetrična. Poglejmo si definicijo in uporabo na primeru.
Pripadajoča matrika sosednosti A je:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix},$$

kjer prva vrstica/stolpec pripada točki 1, druga točki 2, itd.

Na pameten in hiter način dokazati izomorfizem dveh grafov je zelo težko. Znanstveniki kljub trajnim raziskavam niso našli polinomskega algortima za iskanje izomorfizma. V resnici nihče ne zna točno določiti, ali problem določanja izomorfizma spada v razred P ali NP polnih problemov.

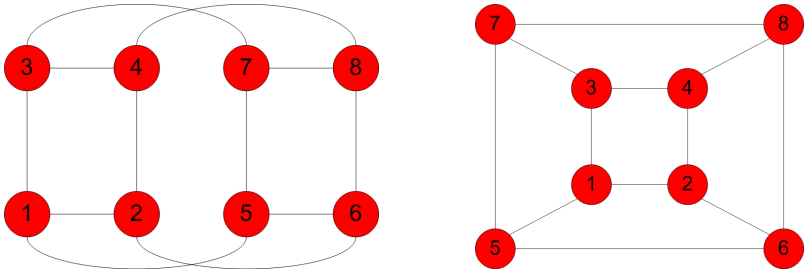
9.1.1 Iskanje izomorfizma

Postopek za iskanje izomorfizma:

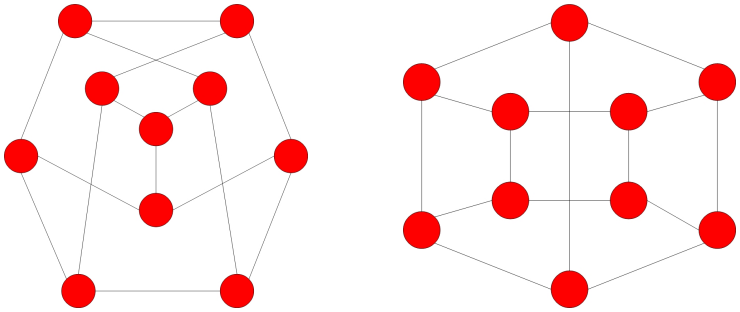
- preverimo enakost valenc grafov,
- preverimo invariante grafov,
- poiščemo primerno bijektivno preslikavo točk,
- primerjamo sosednost vseh točk z ostalimi: $u \sim v \Leftrightarrow \Phi(u) \sim \Phi(v)$.

Če imata dva grafa enake zgoraj naštete lastnosti, potem sta izomorfna. V praksi pa ni enostavno pokazati ali sta dva grafa izomorfna. Lažje je dokazati, da grafa nista izomorfna s pomočjo grafovskih invariant, v katerih se obravnavana grafa ločita. Poglejmo si sedaj primer dveh izomorfnih grafov.

Opazimo, da sta oba grafa na sliki dvodelna. Prvi z razbitjem $A = \{1, 2, 3, 4\}$ in razbitjem $B = \{5, 6, 7, 8\}$. Drugi graf podobno. Vse točke v obeh grafih imajo enako stopnjo 3, povezav je 12 in v obeh primerih vidimo da je točka 1 povezana z enakimi točkami $2, 3, 5$. Podobno ostale točke. Iz tega sledi da smo našli izomorfizem med danima grafoma. Poglejmo sedaj še primer dveh neizomorfnih grafov.



Slika 9.1: Izomorfna grafa



Slika 9.2: Neizomorfna grafa

Vse točke v obeh grafih imajo stopnjo 3 in v obeh grafih je 15 povezav. Če dobro pogledamo, vidimo, da drugi graf na sliki vsebuje cikel dolžine 4, prvi graf pa ne. Podobno vsebuje drugi graf dva cikla dolžine 5, prvi graf pa mnogo več. Iz tega sledi, da grafa nista izomorfna. Prvi graf je sicer izomorfen Petersenovemu grafu, drugi graf pa 5-strani prizmi $C_5 \square K_2$.

Za iskanje izomorfizma splošnih grafov obstajata v glavnem dve metodi:

- *direktna*, tako da vzamemo dva grafa, ki ju lahko primerjamo in poskusimo najti izomorfizem. Prednost te metode je, da če obstaja veliko izomorfizmov, je naša naloga najti le enega.
- preko *kanonične oznake* C, ki je funkcija na množici vseh grafov, tako da sta G_1 in G_2 izomorfna, če in samo če velja $C(G_1) = C(G_2)$.

Na drugi metodi temelji McKay's Nauty algoritem, ki je tudi najbolj praktičen in najhitrejši algoritem za iskanje izomorfizma grafov in ga bomo spoznali v nadaljevanju. Sedaj pa si najprej poglejmo algoritem sestopanja, ki temelji na direktni metodi.

9.1.2 Preprost algoritem sestopanja

Algoritem, ki za iskanje izomorfizma grafov uporablja grafovsko invarianto točk. Naj bosta $G_1 = (V = \{v_1, \ldots, v_n\}, E_1)$ in $G_2 = (W = \{w_1, \ldots, w_n\}, E_2)$ grafa, za katera preverjamo ali sta izomorfna, [25].

Vhodni podatki:

- naj bo R niz z linearnim redom '<'
- naj inv:$V \to R$ označuje invarianto točk, npr. $\text{inv}(v) = d(v)$ in $R = \mathbf{N}$

- naj bo $\Pi(V, \text{inv}) = (V_1, \ldots, V_k)$ točkovna delitev V v povezavi z inv, tj.

$$\forall v, w \in V_i : \text{inv}(v) = \text{inv}(w) \text{ in } \forall v \in V_i, w \in V_j, i < j : \text{inv}(v) < \text{inv}(w).$$

Izhodni podatek:

- preslikava Φ, tako da je $V \to W$, $1 \leq i \leq n$, izomorfizem med grafoma G_1 in G_2. Če taka preslikava ne obstaja, vrne algoritem '$NON - isomorphic$'.

Algorithm 18 ISOMORPH(G_1, G_2, $(V_1, \ldots, V_m)$, $(W_1, \ldots, W_m)$, Φ')

VHOD: Grafa $G_1 = (V = \{v_1, \ldots, v_n\}, E_1)$, $G_2 = (W = \{w_1, \ldots, w_n\}, E_2)$, delitvi točk $(V_1, \ldots, V_m)$, $(W_1, \ldots, W_m)$ z lastnostjo $\bigcup V_i \subset V$, $\bigcup W_i \subset W$ in $|V_i| = |W_i|$, izomorfizem Φ' med podgrafi določenimi z $V \setminus \bigcup V_i$ in $W \setminus \bigcup W_i$.
IZHOD: Φ, če je Φ' mogoče razširiti na izomorfizem Φ med grafoma G_1 in G_2, sicer vrne algoritem 'NON-isomorphic'
if $(V_1, \ldots, V_m) = \emptyset$ **then return** Φ'
end if
Izračunaj $V_i \in \{V_j || V_j| \leq |V_l|, 1 \leq l \leq m\}$
Naj bosta $V_i = \{v_{i1}, v_{i2}, \ldots\}, W_i = \{w_{i1}, w_{i2}, \ldots\}$
for $j = 1, \ldots, |V_i|$ **do**
if Φ' razširjen z $i1 \to ij$ izomorfizem med podgrafi določenimi z $V \setminus \bigcup V_k \cup \{v_{i1}\}$ in $W \setminus \bigcup W_k \cup \{w_{i1}\}$ **then**
$branch = ISOMORPH(G_1, G_2, (V_1, \ldots, V_i \setminus v_{i1}, \ldots, V_m), (W_1, \ldots, W_i \setminus w_{i1}, \ldots, W_m), \Phi' \cup \{i1 \to ij\})$
if $branch \neq$ 'NON-isomorphic' **thenreturn** $branch$
end if
end ifreturn 'NON-isomorphic'
end for

Delitvi točk $\Pi(V, \text{inv}) = (V_1, \ldots, V_k)$ in $\Pi(W, \text{inv}) = (W_1, \ldots, W_{k'})$ sta določeni že na začetku. Če opazimo, da je $k \neq k'$ ali da $|V_i| \neq |W_i|$, za $1 \leq i \leq k$, potem dva grafa ne moreta biti izomorfna, ker ne zadoščata pogoju invariante točk inv. Recimo, da privzamemo, da je $k = k'$ in $|V_i| = |W_i|$. Torej imata množici V_i in W_i enako moč množice. Algoritem preverja izomorfnost podgrafov danih grafov in gre korak za korakom, dokler se bodisi ne ustavi, ker je na koncu in je uspešno preveril že vse točke, bodisi se ustavi, če so preverjene možnosti neuspešne. Izomorfizem podgrafov označimo z Φ', kjer je Φ' bijekcija med dvema skupinama točk $\{1, \ldots, n\}$. Velja, da katerikoli izomorfizem Φ med grafoma G_1 in G_2 preslika točke iz V_i v točke iz W_i. To je dovolj, da fiksiramo/določimo to preslikavo in preverjamo dalje. Te preslikave so določene v for zanki algoritma $ISOMORPH(G_1, G_2, (V_1, \ldots, V_m), (W_1, \ldots, W_m), \Phi')$. Če obstaja izomorfizem Φ, potem je $\Phi(v_{i1}) \in W_i$ in na ta način je preverjanje vseh preslikav $v_{i1} \to w_{ij}$ dovolj za določitev izomorfizma.

9.1.3 McKay-ev Nauty algoritem

Eden izmed načinov, kako izračunati kanonično oznako grafa G je McKay-ev nauty algoritem [19]. Še preden pa si pogledamo algoritem, najprej definirajmo McKay-evo idejo kako definirati kanonično oznako.

Definicija 9.9 (Kanonična oznaka grafa G) Naj bo dan $G = (V, E)$ neusmerjen graf, kjer je $V = \{v_1, v_2, \ldots, v_n\}$ množica vozlišč. $\mathrm{Adj}(G, \delta)$ je matrika sosednosti grafa G glede na razvrstitev vozlišč $v_{\delta(1)}, v_{\delta(2)}, \ldots, v_{\delta(n)}$, kjer je δ permutacija $\{1, \ldots, n\}$. Potem je C_{adj} - kanonična oznaka definirana z

$$C_{adj}(G) = \min_{\delta \in S_n} \mathrm{Adj}(G, \delta).$$

Na $\mathrm{Adj}(G, \delta)$ pa gledamo kot n^2-bitno binarno število dobljeno z združevanjem vrst matrike sosednosti.

Oznaki $C_{adj}(G_1)$ in $C_{adj}(G_2)$ sta enaki če in samo če sta grafa G_1 in G_2 izomorfna. To velja zato, ker je minimum matrike sosednosti enolično določen in sta dva grafa izomorfna le če obstajajo razvrstitve vozlišč grafov iz katerih dobimo iste matrike sosednosti. Naiven način, kako izračunati $C_{adj}(G)$ bi pogledal $n!$ razvrstitev vozlišč in za vsakega posebej še primerjal dve matriki velikosti $n \times n$. Torej je takšen način računanja $C_{adj}(G)$ je tudi za majhne vrednosti n časovno neustrezen. Zato McKay v svojem *Nauty* algoritmu uporablja različne pristope za čim hitrejši izračun oznake $C(G)$. V splošnem $C(G)$ ne bo enak $C_{adj}(G)$, saj McKay-nov algotitem ne pregleda vseh $n!$ razvrstitev vozlišč grafa G, ampak le del in potem izračuna minimum iz dobljenih vzorcev matrik sosednosti. Vzorec bomo dobili s postopkom refiniranja, njegova velikost pa bo odvisna od strukture grafa in ponavadi precej manjša od $n!$. Za iskanje vseh razvrstitev vozlišč, McKay-ev algoritem uporablja iskalno drevo $\mathcal{T}$ v katerem vsak list predstavlja eno izmed izbranih razvrstitev. Nato algoritem prehaja po drevesu $\mathcal{T}$ in preučuje pripadajoče matrike sosednosti obiskanih listov. Vendar pa algoritem ne obišče vseh listov. Teorija grup, natančneje vse kat vemo o avtomorfizmu grup Aut(G) nam pove, da lahko iz pregleda drevesa $\mathcal{T}$ izključimo nekatera poddrevesa. Tako izključimo poddrevo, za katerega vemo, da vodi do razvrstitve vozlišč iz katerih dobimo matrike sosednosti, ki niso manjše od tistih, ki smo jih že našli. Tako bomo z uporabo algebre in predvsem teorije grup lahko videli, da je oznaka grafa C dobljena po zgoraj navedenem postopku obrezovanja drevesa res kanonična. Obstaja še en način kako odstraniti odvečne liste iz drevesa $\mathcal{T}$, vendar je zelo abstrakten in ga bomo omenili le na koncu v zadnjem poglavju.

Osnove teorije grup

Z S_n označimo grupo premutacij z n elementi. Element $\delta \in S_n$ je potemtakem bijekcija med množicama $\{1, \ldots, n\}$ in $\{1, \ldots, n\}$. Očtno obstaja $n!$ takšnih bijektivnih preslikav. Produkt dveh elementov f in g v grupi funkcij je definiran kot kompozicija: $f \cdot g = f \circ g$. Za končno grupo $\mathcal{G}$ in podmnožice elementov $\mathcal{F} \subseteq \mathcal{G}$ je produkt grup $\mathcal{F}$ in $\mathcal{G}$ podgrupa $\langle \mathcal{F} \rangle$ definirana kot

$$\langle \mathcal{F} \rangle = \{f \in \mathcal{G} |\ \exists\ m\ \exists\ f_1, f_2, \ldots, f_m \in \mathcal{F} : f = f_1 \cdot f_2 \cdots f_m\}.$$

Elementi grupe$\mathcal{F}$ pa imenujemo generatorji za $\langle \mathcal{F} \rangle$.

Grupa $\mathcal{G}$ deluje na množici $\mathcal{M}$ z funkcijo $\sigma : \mathcal{G} \times \mathcal{M} \to \mathcal{M}$, če za vsak nevtralan element $e \in \mathcal{G}$ in vse $f, g \in \mathcal{G}$ ter $x \in \mathcal{M}$ velja, da je $\sigma(e, x) = x$ in $\sigma(f \cdot g, x) = \sigma(f, \sigma(g, x))$, potem $\mathcal{G}$ in σ inducirata ekvivalenčno relacijo na $\mathcal{M}$ v naslednjem smislu:

$$x \sim y \iff \exists f \in \mathcal{G} : \sigma(f, x) = y.$$

Ekvivalenčni razred za x, tj. množico $\{\sigma(f, x) | f \in \mathcal{G}\}$ imenujemo tudi orbita x-a. Množico vseh ekvivalenčnih razredov za $\mathcal{M}$ glede na $\mathcal{G}$ in σ pa imenujemo orbita particij. V našem

primeru bo podgrupa $\Phi \subseteq \mathrm{Aut(G)}$ avtomorfizma grup grafa $G = (V, E)$ delovala na množici vozlišč V. Za avtomorfizem $\phi \in \Phi$ ter vozlišče $v \in V$ je funkcija σ definirana kot $\sigma(\phi, v) = \phi(v)$.

Iskalno drevo $\mathcal{T}$

Dan imamo neusmerjen graf $G = (V, E)$ in želimo izračunati njegovo kanonično oznako $C(G)$. Najprej nastavimo začetne oznake vozlišč $V = \{v_1, \ldots, v_n\}$, kjer je n število vseh vozlišč. Tako lahko vpeljemo formalno definicijo razdelitev vozlišč na particije.

Definicija 9.10 (Razdelitev vozlišč na particije) Razdelitev vozlišč na particije grafa G je urejen seznam $\Pi = (V_1, V_2, \ldots, V_r)$, kjer so $V_i \subseteq V$ podmnožice množice vozlišč ali celice za katere velja:

1. $V_i \cap V_j = \emptyset$ za vse $1 \leq i \neq j \leq r$
2. $\bigcup_{i \in \{1,\ldots,r\}} V_i = V$
3. $|V_i| \geq 1$ za vse $1 \leq i \leq r$.

Število celic r Π je dobimo kot $|\Pi|$. Particija vozlišč je enotna če je $r = 1$ in diskretna če $r = n$.

Vsako vozlišče drevesa $\mathcal{T}$ ustreza eni izmed particij vozlišč s katero identificiramo vozlišče. Za določitev particij moramo najprej predstaviti še izpopolnjevalni postopek f ali refiniranje. Za particijo vozlišč Π je $f(\Pi)$ izpopolnitev, tako da velja: za vsako celico V' v $f(\Pi)$ je V v Π, kjer je $V' \subseteq V$. Refiniranje je urejeno tako, da so vozlišča, ki imajo enake sosednosti, grupirana skupaj. Za vozlišče $v \in V$ in množico vozlišč $W \subset V$ naj bo $d(v, W)$ število vozlišč v W, ki so sosedna z v.

Recimo, da želimo poiskati izpopolnitev $f(\Pi)$ enotne particije $\Pi = (V)$. Na prvem koraku števila $d(v, V)$ izračunamo za vsako vozlišče v grafa G, kar pomeni, da je $d(v, V)$ kar stopnja posameznega vozlišča. Nato vozlišča združimo v skupine glede na njihovo stopnjo tako, da so vozlišča z isto stopnjo v isti skupini. Kot rezultat prvega koraka refiniranja dobimo particijo $\Pi_1 = (W_1, W_2, \ldots, W_j)$ v kateri ima vsak par vozlišč, ki pripada isti skupini isto stopnjo in za vozlišči $v \in W_k$ ter $w \in W_l$ velja da je $d(v, V) < d(w, V)$ samo če je $k < l$. V naslednjem koraku vsako celico iz Π_1 izpolnemo glede na Π_1, torej isti postopek ponovimo še enkrat na Π_1. Za vsako vozlišče v celice W_i izračunamo vektor $\eta(v) = (d(v, W_1), d(v, W_2), \ldots, d(v, W_j))$ in vozlišča W_i razdelimo na podskupine glede na izračunane $\eta(v)$ tako da jih primerjamo leksikografsko. Ko to storimo za vse celice W_i, dobimo particijo Π_2. Π_3 je potem izpopolnitev particije Π_2. Postopek pa ponavljamo vse dokler ni Π_{i+1} prava izpolnitev particije Π_i. Postopek je formalno opisan še v algoritmu:

Particija $f(\Pi) = (V_1, V_2, \ldots, V_{r'})$ izpolnjuje naslednjo lastnost: za vsak par (ne nujno različnih) celic V_i, V_j iz $f(\Pi)$ in vsak par vozlišč $v, w \in V_i$ mora veljati, da je $d(v, V_j) = d(w, V_j)$. Particijo s to lastnostjo imenujemo **pravična particija**. Pravimo tudi, da sta dve vozlišči **strukturno ekvivalentni**, če ležita v isti celici iz $f(\Pi)$.

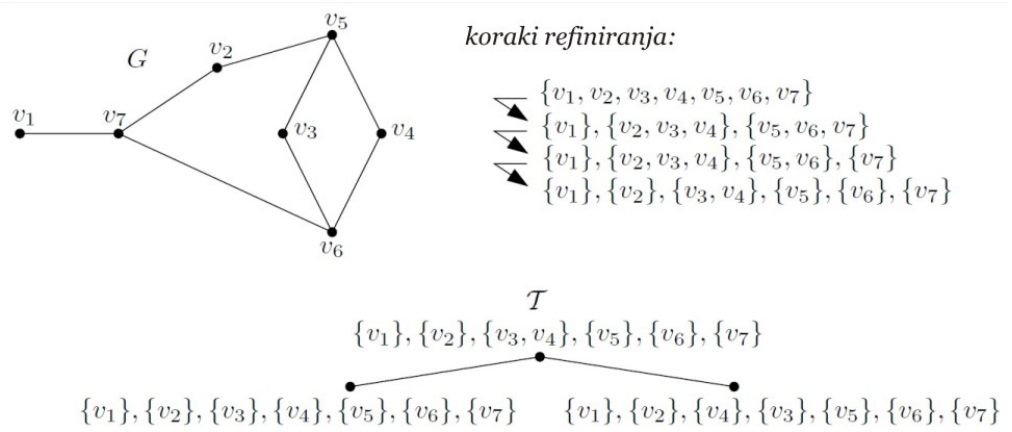
Sedaj lahko natančno opišemo vozlišča iskalnega drevesa $\mathcal{T}$. Vsa vozlišča ustrezajo pravičnim particijam. Koren vozlišča je $\Pi = (V_1, V_2, \ldots, V_r)$ usteza izpolnitvi enotne particije $f((V))$. Če je Π že diskretna particija Π nima potomcev in ima drevo $\mathcal{T}$ eno samo vozlišče, drugače pa potomce Π dobimo na naslednji način: naj bo $V_i = \{v'_1, v'_2, \ldots, v'_m\}$ prva netrivialna celica v Π tj. prva celica, ki več kot le eno samo vozlišče. Potem ima Π

Algorithm 19 Refiniranje

Vhod: Razdelitev vozlišč $\Pi = (V_1, \ldots, V_r)$
  Nastavi $\Pi_{nov} = \Pi$
  repeat
    $\Pi_{star} = \Pi_{nov}$
    Naj bo $\Pi_{star} = (V_1, \ldots, V_r)$
    for i=1 to r' **do**
      for all $v \in V_i$ **do**
        Izračunaj $\eta(v) = (d(v, V_1), \ldots, d(v, V_{r'}))$
      end for
      Particija V_i v $W_1, \ldots, W_j$ tako da za $v \in W_k,\ w \in W_l : \eta(v) < \eta(w) \iff k < l$
      Zamenjaj V_i v Π_{nov} z $W_1, \ldots, W_j$
    end for
  until $\Pi_{star} = \Pi_{nov}$ **return** Π_{nov}

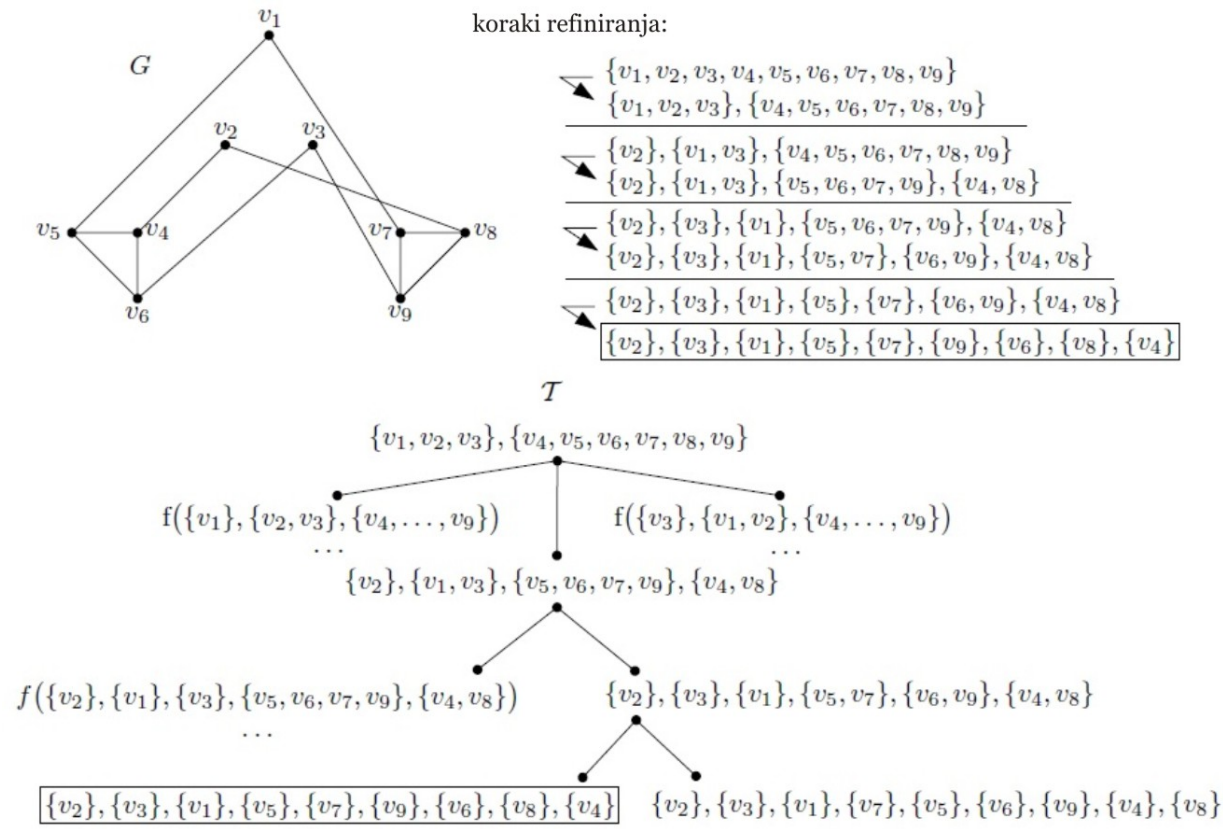
m potomcev. Označimo jih z $f(\Pi \backslash v'_1), f(\Pi \backslash v'_2), \ldots, f(\Pi \backslash v'_m)$, kjer je $f(\Pi \backslash v'_j)$ okrajšana oznaka za $f((V_1, \ldots V_{i-1}, \{v'_j\}, V_i \backslash \{v'_j\}, V_{i+1}, \ldots, V_r))$. Kar pomeni, da je posamezen potomec sestavljen tako, da iz celice vzamemo eno vozlišče $v' \in V_i$ in iz njega naredimo samostojno celico $\{v'\}$. To ima smisel le, če je Π pravična particija.

Za vsako drugo vozlišče $\Pi' \in \mathcal{T}$, ki ni diskretna particija ponovimo isti postopek kot smo ga naredili za Π. Tako vsi listi iskalnega drevesa $\mathcal{T}$ predstavljajo diskretne particije. Vrstni red vozlišč v teh particijah $(\{v_{\delta(1)}\}, \ldots, \{v_{\delta(n)}\})$, $\delta \in S_n$ določi matriko sosednosti, ki pripada listu. Glavni namen f je narediti drevo $\mathcal{T}$ čim manjše v pomenu sktrukturno ekvivalentnih vožlišč glede na pripadanjočo particijo. Vendar pa je velikost drevesa $\mathcal{T}$ odvisna od strukture grafa G. V naslednjih dveh primerih je podan izračun dreves $\mathcal{T}$ za dva grafa in opazimo lahko, da prvemu pripada drevo s tremi vozlišči, drugemu pa bistveno večje drevo.



Slika 9.3: Graf G in pripadajoče iskalno drevo $\mathcal{T}$. Na začetku sta vozlišči v_3 in v_4 strukturno ekvivalentni.

Izrek 9.11 *Naj bosta dana G_1 in G_2 neusmerjena grafa in naj bosta $C(G_1)$ in $C(G_2)$*



Slika 9.4: Graf G in del pripadajočega iskalnega drevesa $\mathcal{T}$. Na začetku so ekvivalentna vozlišča $\{v_1, v_2, v_3\}$ in $\{v_4, \ldots, v_9\}$.

oznaki grafov, ki smo ju dobili s pomočjo iskalnih dreves. Potem velja

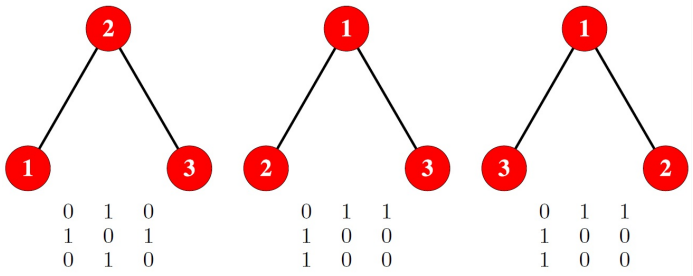
$$C(G_1) = C(G_2) \Longleftrightarrow G_1 \simeq G_2.$$

Po eni strani je očitno, da neizomorfna grafa G_1 in G_2 ne moreta imeti enake oznake $C(G_i)$, saj je vsaka matrika sosednosti G_1 različna od matrik sosednosti, ki pripadajo grafu G_2. Po drugi strani pa nam da jasen predpis kako definirati iskalno drevo $\mathcal{T}$ idejo, da izomorfna grafa res dobita enako oznako.

Uporaba avtomorfizma za rezanje $\mathcal{T}$

Nauty algoritem nam ne vrne iskalnega drevesa $\mathcal{T}$ eksplicitno. Namesto tega ga razčleni $\mathcal{T}$ v posebnem vrstnem redu od zgoraj navzdol in hkrati poskuša izločiti čim več poddreves. Pravzaprav particija, ki pripada nekimu vozlišču ni izračunana vse dokler ga ne obiščemo. Ko algoritem obišče list l izračuna še propadajočo matriko sosednosti A_l. Sproti pa si tudi shranjuje minimalno matriko sosednosti A_{min}, ki smo jo našli do tekočega koraka. Ko algoritem prvič pride do lista l_1 nastavi A_{min} na A_{l_1}. Pri vsakem naslednjem listu pa sproti preveri ali je $A_l < A_{min}$ in če je, spremeni A_{min} na A_l. Tako A_{min} vsebuje oznako $C(G)$. Poleg tega si algoritem sproti shrani tudi podgrupo $\Phi_t(G)$ avtomorfizma grupe $\mathcal{G}$. Označili jo bomo z $\Phi_t(G)$ in zanjo velja, da $\Phi_t(G) = \langle \phi_1, \ldots, \phi_{i(t)} \rangle$, kjer so $\phi_1, \ldots, \phi_{i(t)}$ vsi avtomorfizmi, ki jih poznamo do časa t. V grafu najdemo avtomorfizem ϕ takrat, ko dva lista inducirata enako matriko sosednosti. Naj bosta $w_1, w_2, \ldots, w_n$ ter $w'_1, w'_2, \ldots, w'_n$

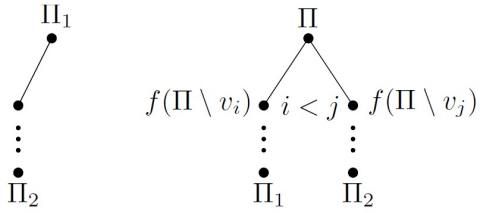
vrstna reda vozlišč v vsakem izmed dveh listov. Potem je $\phi : w_i \to w_i'$ za $i = 1, \ldots, n$ avtomorfizem. Enostaven primer pa je prikazan tudi na sliki.



Slika 9.5: Avtomorfizem prepoznamo po obliki matrike sosednosti, ki pripada vsakemu izmed grafov glede na oznake vozlišč. Če sta matriki enaki, dobimo avtomorfizem.

Da bi videli kako obrezati iskalno drevo $\mathcal{T}$ potrebujemo najprej še nekaj osnovnih definicij. Ena izmed njih je definition linearnega reda na vozliščih drevesa $\mathcal{T}$, ki nam označi vozlišča od zgoraj navzdol. Če je Π notranje vozlišče $\mathcal{T}$, potem definiramo še poddrevo s korenom v potomcu Π in ga označimo z $\mathcal{T}(\Pi \backslash v_i)$.

Definicija 9.12 (Linearni vrstni red vozlišč iskalnega drevesa $\mathcal{T}$) Naj bosta Π_1 in Π_2 dve različni vozlišči $\mathcal{T}$ in naj bo Π njun zadnji skupni prednik. Definiramo $\Pi_1 \preceq \Pi_2$, če je $\Pi_1 = \Pi$ ali pa če za vozlišči v_i in v_j, ki pripadata prvi netrivialni celici iz Π in $\Pi_1 \in \mathcal{T}(\Pi \backslash v_i)$ ter Π in $\Pi_2 \in \mathcal{T}(\Pi \backslash v_j)$ kjer $i < j$. Drugače pa je $\Pi_1 \succeq \Pi_2$.



Slika 9.6: Linearni red vozlišč iskalnega drevesa $\mathcal{T}$. V obeh primerih velja, da je $\Pi_1 \preceq \Pi_2$.

Očitno je relacija $\preceq$ linearna, to lahko opazimo tudi iz slike. Nauty algoritem obiskuje vozlišča iskalnega drevesa v vrstnem redu, ki ga določa linearni vrstni red. V naslednjem koraku bomo definirali še ekvivalenčno relacijo na vozliščih $\mathcal{T}$.

Definicija 9.13 (Ekvivalenčna realcija na vozliščih $\mathcal{T}$) Naj bo dana particija $\Pi_1 = (V_1, \ldots, V_m) \in \mathcal{T}$ ter $\Pi_2 = (W_1, \ldots, W_m) \in \mathcal{T}$. Potem je $\Pi_1 \sim \Pi_2$, če in samo če obstaja avtomorfizem $\phi \in \mathrm{Aut(G)}$ in permutacija $\delta \in S_m$ tako da velja $\phi(V_i) = W_{\delta(i)}$ za $i = 1, \ldots, m$. Potem pravimo, da ϕ določa $\Pi_1 \sim \Pi_2$.

Izrek 9.14 *Naj bo $\Pi_1 \sim \Pi_2 \in \mathcal{T}$ in naj bosta $\mathcal{T}_1$, $\mathcal{T}_2$ poddrevesa $\mathcal{T}$ s korenom v Π_1 in Π_2. Potem za vsako vozlišče $\Pi_1' \in \mathcal{T}_1$ obstaja vozlišče $\Pi_2' \in \mathcal{T}_2$ da je $\Pi_1' \sim \Pi_2'$.*

Kot direktno posledico izreka dobimo, da lahko poddrevo $\mathcal{T}_2$ s korenom v $\Pi_2 \in \mathcal{T}$ izločimo iz nadalnje preiskave, če vemo, da obstaja vozlišče Π_1 in velja $\Pi_1 < \Pi_2$ ter $\Pi_1 \sim \Pi_2$. To sledi iz dejstva, da so listi iz drevesa $\mathcal{T}_2$ ekvivalentni listom drevesa s korenom v Π_1. Tako smo že pregledali matrike sosednosti, ki bi jih inducirali listi drevesa $\mathcal{T}_2$. Videti moramo le kako uporabimo $\Phi_t(G)$ za iskanje notranjih vozlišč drevesa, ki so si med seboj ekvivalentni. Za vozlišče $v \in V$ je $\{\phi(v) \mid \phi \in \Phi_t(G)\}$ njegova orbita glede na $\Phi_t(G)$. Naj bo Θ_t particija V ob času t. Algoritem lahko dostopa do Θ_t v vsakem času t. Na začetku je Θ_t diskretna particija, tj. $\Theta_0 = \{v_1\}, \ldots, \{v_n\}$. Na vsakem koraku algoritma odkrijemo kak nov avtomorfizem in tako posodobimo Θ_t. To pomeni, da postaja Θ_t vse bolj groba particija ko odkrivamo nove avtomorfizme in s tem širimo $\Phi_t(G)$, vozlišča pa postajajo med seboj ekvivalentna glede na $\Phi_t(G)$. Tako za vsako vozlišče $\Pi \in \mathcal{T}$ vemo, kateri izmed njegovih potomcev so ekvivalentni.

Izrek 9.15 *Naj bo $\Pi = (V_1, \ldots, V_r) \in \mathcal{T}$ in $V_i = \{v'_1, \ldots, v'_m\}$ prva netrivialna celica Π. Če obstajata $v_i, v_j \in V_i$, ki ležita v isti orbiti particije Θ_t, obstaja avtomorfizem $\phi \in \Phi_t(G)$ in velja $f(\Pi \backslash v'_i) \sim f(\Pi \backslash v'_j)$.*

Ta izrek lahko uporabimo za obrezovanje drevesa $\mathcal{T}$ na dva načina. Prvi način je precej enostaven. Ko algoritem doseže neko vozlišče $\Pi \in \mathcal{T}$ z netrivialno celico vozlišč V_i, Θ_t inducira particijo V_i-ja na celice tako, da vsak par vozlišč v posamezni celici leži v isti orbiti. To particijo označimo z $\Theta_t \wedge V_i$. Po prejšnjem izreku potem velja, da moramo obravnavati le enega izmed potomcev $\mathcal{T}(\Pi \backslash v')$ za vsako celico $\Theta_t \wedge V_i$. Za v' izberemo tisto vozlišče iz V_i, ki ima najmanjšo stopnjo oz. najmanjši začetni indeks izmed vseh vozlišč. Z drugimi besedami, obravnavati moramo tiste potomce tistih vozlišč z najmanjšo stopnjo v $\Theta_t \wedge V_i$.

Drug način rezanja je malo bolj zapleten. Predpostavimo lahko, da algoritem pride do vozlišča $\Pi \in \mathcal{T}$ v času t_1. Tako kot prej z V_i označimo prvo netrivialno celico particije Π in naj bosta $v_i, v_j \in V_i$ vozlišči, ki ne ležita v isti orbiti glede na Θ_{t_1}. To pomeni, da bo algoritem obiskal obe poddrevesi $\mathcal{T}(\Pi \backslash v_i)$ in $\mathcal{T}(\Pi \backslash v_j)$ na podlagi informacij, ki jih bo dobil iz Θ_{t_1}. Brez škode za splošnost lahko privzamemo, da ima vozlišče v_i manjši indeks kot v_j, kar pomeni, da bomo $\mathcal{T}(\Pi \backslash v_i)$ preučili prej kot $\mathcal{T}(\Pi \backslash v_j)$. Algoritem v času t_2 najde nov avtomorfizem ϕ' tako, da da za avtomorfizem $\phi \in \Phi_{t_2}(G)$ velja $\phi(v_i) = v_j$. To velja, saj v_i in v_j ležita na isti orbiti glede na Θ_{t_2}. (Kjer ϕ ni nujno samo zase nov avtomorfizem ϕ' ampak je lahko kompozitum ϕ' in še drugih avtomorfizmov, ki jih poznamo že od prej.) Tako imamo dovolj informacij, da algoritem iz pregleda izključi pregled poddrevesa $\mathcal{T}(\Pi \backslash v_j)$. Seveda, tega ne moremo upoštevati, če smo v času t_2 že pregledali $\mathcal{T}(\Pi \backslash v_j)$. Sicer pa lahko pregled tega poddrevesa preskočimo ali pa prekinemo, če smo že začeli z pregledovanjem. Tako se vrnemo nazaj v Π, saj smo našli avtomorfizem, da Θ_t dovoljuje ta korak. Pravzaprav se lahko algoritem vrne še bolj nazaj, k prednikom Π, saj ima Θ_t dovolj informacij in lahko končamo pregled celotnega poddrevesa, ki vsebuje Π. Ko najdemo kak nov avtomorfizem, algoritem v hipu skoči nazaj po vozliščih drevesa, čim višje kot lahko, da dobi nove informacije.

Naslednje pomembno vprašanje je, kako veliko naj bo število matrik, ki jih shranjujemo medtem, ko pregledujemo drevo. Shranjevanje le teh in primerjanje med sabo vzame veliko časa in prostora. Seveda, če algoritem shranjuje veliko število matrik sosednosti, bomo tudi odkrili veliko več avtomorfizmov. Prav tako lahko iskalno drevo obrežemo bolj učinkovito, kar pa bo zmanjšalo časovno zahtevnost. McKay je v praksi testiral algoritem le z dvema matrikama. Na vsakem koraku ima nauty algoritem shranjeno matriko prvega

lista, ki ga je obiskal A_{l_1} in A_{min}.

Algorithm 20 Nauty algoritem

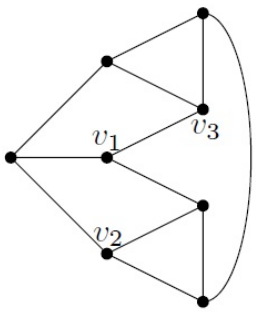
Vhod: Graf $G = (V, E)$ in začetno inicializacijo vozlišč $V = \{v_1, \ldots, v_n\}$
$\Phi(G) = id$
$\Theta = \{v_1\}, \ldots, \{v_n\}$
Izvajaj $(f((V)))$ **return** A_{min}
Izvajaj $(\Pi = (V_1, \ldots, V_r))$
if r=n **then**
Označi $V_1 = \{v'_1\}, \ldots, V_n = \{v'_n\}$ z razvrstitvijo vozlišč $v'_1, \ldots, v'_n$
Izračunaj matriko sosednosti A_Π, inducirano z razvrstitvijo vozlišč $v'_1, \ldots, v'_n$
if A_{l_1}=nil **then**
$A_{l_1} = A_\Pi$, razvrstitev_vozlišč$(A_{l_1})= v'_1, \ldots, v'_n$
$A_{min} = A_\Pi$, razvrstitev_vozlišč$(A_{min})= v'_1, \ldots, v'_n$
else
if $A_{l_1} > A_\Pi$ **then**
$A_{min} = A_\Pi$, razvrstitev_vozlišč$(A_{min})= v'_1, \ldots, v'_n$
else
$\phi = nil$
if $A_{l_1} = A_\Pi$ **then**
Izračunaj avtomorfizem ϕ, ki ga inducira razvrstitev vozlišč A_{l_1} in $v'_1, \ldots, v'_n$
end if
if $A_{min} \neq A_{l_1}$ in $A_{min} = A_\Pi$ **then**
Izračunaj avtomorfizem ϕ, ki ga inducira razvrstitev vozlišč A_{min} in $v'_1, \ldots, v'_n$
end if
if $\phi \neq nil$ **then**
$\Phi(G) = \langle\{\Phi(G) \cup \phi\}\rangle$
Posodobi Θ
Preveri in skoči nazaj
end if
end if
end if
else
naj bo $V_i = \{v'_q, \ldots, v'_m\}$ prva netrivialna celica Π
naj bo $v"_q, \ldots, v"_{m'}$ najmanjša celicam ki predstavlja $\Theta \wedge V_i$
for $j = 1, \ldots, m'$ **do**
Izvajaj $(f(\Pi \backslash v"_j))$
end for
end if

9.1.4 Zahtevnost iskanja izomorfizmov oziroma kako prelisičiti nauty algoritem

Rečemo lahko, da zahtevnost iskanja izomorfizma med grafi še ni dokončno znana. Predpostavimo lahko, da verjamemo da je zahtevnost polinomska in skušamo najti tak al-

goritem, ki bo problem izomorfizma rešil v polinomskem času. Ideja takega algoritma bi bila zelo podobna McKay-evem nauty algoritmu. V tej seminarski nalogi poskušamo prikazati dejansko težavnost iskanja izomorfizma med grafi. Pokazali bomo da tudi tak način iskanja izomorfizmov ni dober. Načeloma želimo ohraniti idejo *nauty* algoritma, vendar uporabimo drug način refiniranja tako, da izračunamo le en list iskalnega drevesa. Oznaka $C(G)$ je potemtakem spet definirana kot matrika sosednosti, ki jo inducira vrstni red vozlišč dobljenega lista. Kot smo omenili že prej oznaki dveh neizomorfnih grafov G_1 in G_2 ne bosta nikoli enaki, saj je matrika sosednosti grafa G_1 različna od matrike sosednosti grafa G_2. Da se prepričamo ali izomorfna grafa res prepoznamo kot izomorfna želimo zagotoviti naslednje: Naj bo Π_k list grafa G_1, ki smo ga izračunali in določa $C(G_1)$ in naj bo $\Pi'_{k'}$ list grafa G_2 in določa $C(G_2)$. Naj bodo še $\Pi_1, \ldots, \Pi_k$ in $\Pi'_1, \ldots, \Pi'_{k'}$ particije vozlišč, ki smo jih izračunali, da smo dobili Π_k ter $\Pi'_{k'}$. Potem mora veljati, da $k = k'$ in se za vsak $i = 1, \ldots, k$ Π_i ujema z Π'_i v številu celic in velikosti vsake celice. Na koncu za $\Pi_i = (V_1 \ldots, V_r)$ in $\Pi'_i = (V'_1 \ldots, V'_r)$ in za vsak par $V_j = \{v_1, \ldots, v_m\}$, $V'_j = \{v'_1, \ldots, v'_m\}$ celic zahtevamo še, da velja: za vse $(v, v') \in V_j \times V'_j$ med G_1 in G_2 obstaja izomorfizem ϕ in je $\phi(v) = v'$. Zadnji pogoj je zadosten, da v iskalnem drevesu lahko izračunamo le en list. Potem je vseeno kateri izmed vozlišč v in v' vzamemo iz prvih netrivialnih celic Π_i in Π'_i, jih definiramo kot nove ekvivalenčne razrede in refiniramo glede na nove particije. Če želimo videti, da to res drži, moramo upoštevati še, da če se res izkaže, da sta $C(G_1)$ in $C(G_2)$ enaka, ko pripadajoča vrstna reda vozlišč listov iskalnega drevesa inducirata izomorfizem ϕ med grafoma G_1 in G_2. Ko definiramo $\{v\}$ in $\{v'\}$ kot nova ekvivalenčna razreda preprosto pomeni, da nastavimo $\phi(v) = \phi(v')$ in refiniramo glede na te informacije. Če pa slučajno zares obstaja izomorfizem, ki slika v v v' od zgoraj, ga bomo tudi s tem pogojem našli.

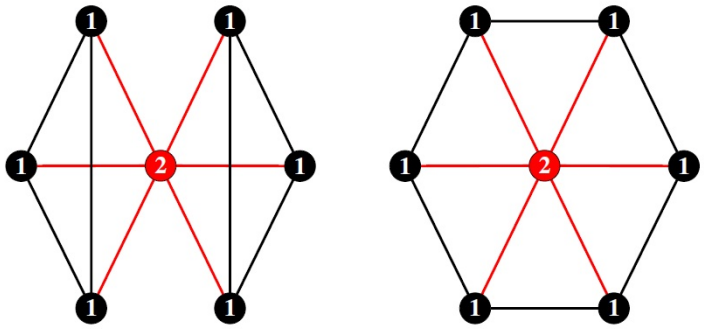
Iskanje izomorfizma grafa v polinomskem času je na način, ki je opisan zgoraj, res težak problem. Za boljšo predstavo dejanskega problema bosta v nadaljevanju predstavljena dva protiprimera, ki nam bosta pokazala, da tudi nauty algoritem ni dovolj dober za iskanje izomorfizmov. Najprej pogledamo postopek refiniranja f nauty algoritma. 3-regularni graf G zagotavlja, da nam f ne pomaga rešiti problema izomorfizma v polinomskem času. Za ta graf $G = (V, E)$ velja da je $d(v, V) = d(w, V)$ za vsak par vozlišč $v, w \in V$, saj je G regularen graf. Tako enotne particije z f ne moremo več refinirati naprej. Če imamo dve kopiji grafa G, G_1 ter G_2 in iz prvega vzamemo $v_1 \in V$ in izračunamo pripadajoči $C(G_1)$ ter analogno za drugega za vozlišče v_2 dobimo, da grafa G_1 in G_2 nista izomorfna. Tako očitno ne obstaja izomorfizem, ki bi slikal v_1 v v_2. Vozlišče v_1 je od vseh ostalih oddaljeno za 2, medtem ko je razdalje med v_2 in v_3 enaka tri.



Slika 9.7: 3-regularni graf G.

Sedaj bi radi videli, kaj se zgodi če namesto f vzamemo kakšen drug postopek refiniranja, ki uporabi več informacij kot le oddaljenost od drugih celic. Kot že rečeno je bila ideja f razdeliti množico vozlišč v ekvivalenčne razrede tako, da je imel vsak par vozlišč v posamezni celici enako število sosedov v vseh ostalih celicah.

Za $v \in V$, $W \subset V$ in i naravno število naj bo sedaj $d_i(v, W)$ število vozlišč v W na razdalji i do v. Poskusili bomo nadgraditi f in refinirati tako dolgo dokler bo veljalo: za vsaki dve vozlišči v in w iz posamezne celice morajo biti števila $d_i(v, W)$ in $d_i(w, W)$ enaka glede na vsako celico W in vsako naravno število i.. Tako 3-regularni graf ni več proti primer za postopek refiniranja f. Kljub temu pa tudi ta izpopolnjen postopek ni dober saj je njegov protiprimer prikazan na drugem grafu. Oznaka vsakega vozlišča v ustreza ekvivalenčnemu razredu, kateremu pripada vozlišče po refinizaciji enotne particije. Vsako vozlišče označeno z 1 ima dve vozlišči tipa 1 in eno vozlišče tipa 2 na razdalji 1 ter tri vozlišča tipa 1 na razdalji 2, medtem ko ima vsako vozlišče z oznako 2 šest vozlišč tipa 1 na razdalji 1. Iz slike pa je očitno tudi, da ne obstaja izomorfizem, ki bi preslikal vozlišče tipa 2 iz prvega v drugi graf.



Slika 9.8: Grafa z dvema komponentama.

9.2 Podobnost grafov

Izomorfizem grafov preverja, če imata dva grafa isto strukturo. Ker je to omejevalni kriterij, bi bilo bolj naravno gledati, če sta si dva grafa med seboj podobna. S tem se ukvarja teorija podobnosti grafov oziroma imenovana tudi teorija ujemanja (graph matching). Ta primerja dva grafa in da neko mero podobnosti, ki jo imenujemo razdalja. Na kakšni razdalji je en graf podoben drugemu.

Pomembna prednost podobnosti nad izomorfizmom je sposobnost obvladovanja z napakami, recimo na vhodnih podatkih, ki se pogosto pojavijo pri zbiranju dejanskih podatkov. Take napake lahko spremenijo izomorfne grafe v neizomorfne, zato je uporaba izomorfizma neprimerna.

Veliko aplikacij uvaja označevanje točk ali povezav, npr. v molekorskih strukturah je označevanje določeno z vrsto elementov. Točke in povezave z različnimi oznakami včasih ne smemo med sabo primerjati. Ker nas zanimajo le strukturne podobnosti, v nadaljevanju obravnavamo vse grafe kot neoznačene.

Mera podobnosti grafov mora izpolnjevati določene lastnosti. Na primer, oddaljenost grafa G_1 do grafa G_2 biti enaka kot od G_2 do G_1, razdalja med izomorfnimi grafi mora biti enaka 0. Meri s takimi lastnosti imenujemo mera razdalje med grafi.

Definicija 9.16 Naj bodo G_1, G_2 in G_3 grafi. Funkcija $d : G_1 \times G_2 \to R_0$ se imenuje *mera razdalje med grafi*, če veljajo naslednje lastnosti:

1. Refleksivnost: $d(G_1, G_2) = 0 \Leftrightarrow G_1 \cong G_2$
2. Simetričnost: $d(G_1, G_2) = d(G_2, G_1)$
3. Trikotniška neenakost: $d(G_1, G_2) + d(G_2, G_3) \geq d(G_1, G_3)$.

Mere razdalje med grafi je težko računati. Že sama lastnost refleksivnost nam dejansko pove rešitev za izomorfizem grafov. Zato v praksi malo omilimo lastnosti mer ali pa računamo le z približkom mere.

Zaradi enostavnosti bomo pogledali le neusmerjene povezane grafe. Vse trditve se lahko razširijo tudi v nepovezane grafe s tem, da gledamo samo njihove povezane komponente.

Predstavili bomo tri načine merjenja podobnosti. Mero, ki temelji na velikosti največjega skupnega podgrafa. Druga mera pa temelji na razliki dolžin ustreznih poti. Tretji pristop določa razdaljo med dvema grafoma v smislu koliko operacij preurejanja je potrebnih za preoblikovanje enega grafa v drugega.

9.2.1 Urejevalna razdalja

Osnovna in fleksibilna metoda za ujemanje strukturnih objektov je urejevalna razdalja. Ta je določena kot minimalno število operacij potrebnih za preoblikovanje enega objekta v drugega. Znan primer je urejevalna razdalja niza (minimalno število operacij (zbriši, vstavi, zamenjaj), ki spremeni en niz v drugega).

Pri urejevalni razdalji grafov tipične operacije vključujejo vstavljanje, brisanje in nadomestitev točk in povezav. Splošno ni določeno katere operacije so nam dovoljene in koliko je strošek posamezne operacije. Dober izbor dovoljenih operacij je zelo odvisen od aplikacije. Lahko imamo operacije, ki se bolje prilegajo posebnim zahtevam in tem določimo poljubne nenegativne stroške. Razdalja se določi kot minimalni strošek vseh zaporedij operacij, ki preoblikujejo en graf v drugega.

Razdaljo je mogoče najbolj učinkovito izračunati na enostavnih nizih dovoljenih operacij.

Zgled 9.17 Na prvem primeru pokažemo operacije s katerimi je lahko upravljati, ampak ne vodijo do konkretnih rezultatov. Dovoljenje so naslednje operacije:

- dodajanje točke - grafu dodamo novo (izolirano) točko
- izbris točke - grafu izbrišemo (izolirano) točko
- dodajanje povezave - grafu dodamo novo povezavo med poljubnimi točkami
- brisanje povezave - povezava je izbrisana iz grafa

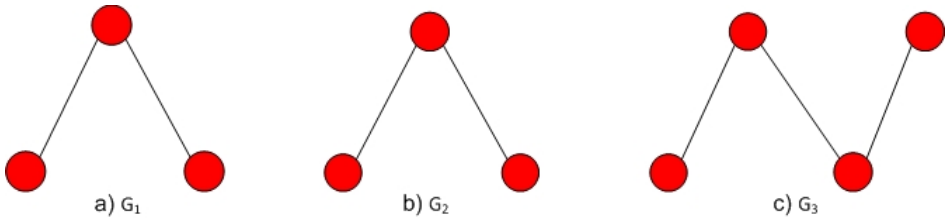
Strošek operacij s točkami je enak 1, medtem pri operacijah s povezavami ni stroškov. Razdalja definirana s temi specifikacijami je enaka razliki številu točk pri obeh grafih:

$$d_{prim1} = ||V(G_1)| - |V(G_2)||.$$

To pomeni, da so na primer pot, zvezda in klika na enakem številu točk podobni glede na tako definirano razdaljo.

Zgled 9.18 (Paradopoulos, Manolopoulos [21]) Na voljo imamo uporabo naslednjih operacij, kjer ima vsaka strošek ena:

- dodajanje točke - grafu dodamo novo (izolirano) točko
- izbrisanje točke - grafu izbrišemo (izolirano) točko
- posodobitev povezave - ena končna točka na povezavi se spremeni



Slika 9.9: Primer urejevalne razdalje

Z uporabo teh operacij lahko na grafih na sliki 9.9 vidimo, da sta potrebni dve operaciji, da se G_1 ujema z G_2, torej dve posodobitvi povezave, medtem ko so tri operacije potrebne, da se G_1 ujema z G_3: eno dodajanje točke in dve posodobitvi povezave. Tako je pri teh operacijah G_1 bolj podoben G_2 kot G_3.

Ker je računanje smiselne urejevalne razdalje grafa težko, lahko pogoje ujemanja rahlo sprostimo. Pri dveh grafih G_1 in G_2, namesto, da spreminjamo G_1 v G_2, G_1 spremenimo v graf z istim številom točk in povezav ter zaporedjem stopenj točk kot graf G_2. Torej, da gledamo samo velikost in zaporedje stopenj.

9.2.2 Razlika v dolžini poti

Naslednja mera podobnosti je primer mere urejevalne razdalje. Definicija je kar enostavna, ampak računanje je lahko zahtevnejše. Grobo povedano, se ukvarjamo z vsoto razlik v dolžini ustreznih poti. Ta mera ima smisel samo za grafe z enakim številom točk in zato bomo v nadaljevanju primerjali le-te.

Definicija 9.19 Naj bosta G_1 in G_2 izomorfna povezana grafa z izomorfizmom $\phi : V(G_1) \to V(G_2)$. Dve točki grafa G_1 sta *sosednji*, če sta njihovi izomorfni točki v G_2 sosednji. Z drugimi besedami:

$$\forall u, v \in V(G_1) : \{u, v\} \in E(G_1) \Leftrightarrow \{\phi(u), \phi(v)\} \in E(G_2).$$

Ekvivalentna formulacija razširja to lastnost povezave od razdalje ene točke do razdalj poljubnih parov točk:

$$\forall u, v \in V(G_1) : d_{G_1}(u, v) = d_{G_2}(\phi(u), \phi(v)). \tag{9.1}$$

Naj bosta G_1 in G_2 poljubna povezana grafa z enakim številom točk in $\sigma : V(G_1) \to V(G_2)$ bijekcija. Potem ni nujno, da (9.1) drži. Namesto tega lahko uporabimo razliko teh dolžin poti, da definiramo podobnost dveh grafov glede na bijekcijo σ.

Definicija 9.20 Za dva povezana grafa G_1 in G_2 z istim številom točk in bijekcijo σ: $V(G_1) \to V(G_2)$ definiramo *σ-dolžino* d_σ z

$$d_\sigma(G_1, G_2) = \sum_{\{u,v\} \in V(G_1) \times V(G_1)} |d_{G_1}(u, v) - d_{G_2}(\sigma(u), \sigma(v))|,$$

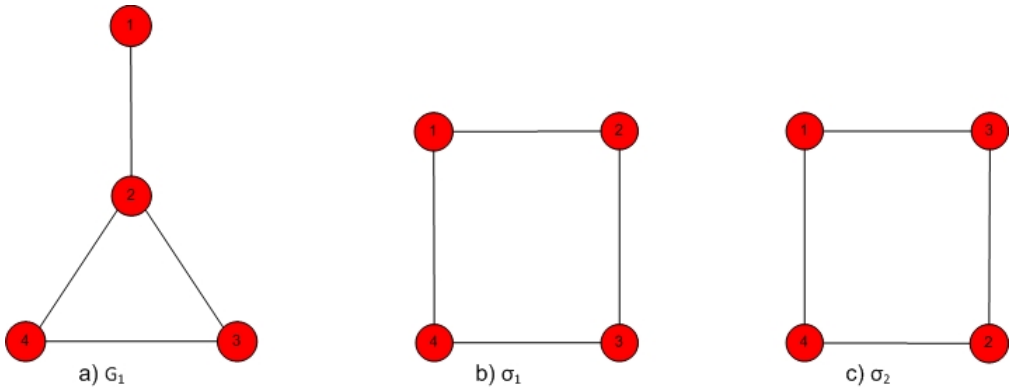
kjer gre vsota po vseh parih točk grafa G_1.

Ker podobnost dveh grafov ni odvisna od preslikave med dvema množicama točk, je razdalja definirana kot minimum po vseh možnih bijekcijah med $V(G_1)$ in $V(G_2)$.

Definicija 9.21 Za dva povezana grafa G_1 in G_2 z istim številom točk definiramo *dolžino poti* d_{pot} kot

$$d_{\text{pot}}(G_1, G_2) = \min_{\sigma \in \Lambda} d_\sigma(G_1, G_2),$$

kjer je Λ množica vseh bijekcij med $V(G_1)$ in $V(G_2)$.



Slika 9.10: Primer razlike v dolžini poti

Zgled 9.22 Naj bo G_1 graf kot na sliki 9.10 in naj bo G_2 cikel na 4 točkah. Na prvi pogled izgleda, kot da obstaja 4! bijektivnih preslikav iz $V(G_1)$ na $V(G_2)$. Ampak, ker imata grafa zelo simetrično strukturo, obstajata le dve neenakovredni preslikavi glede na razdaljo poti. Ti vidimo na sliki 9.10. Preslikavi $\sigma : V(G_1) \to V(G_2)$ sta definirani z $\sigma_i j = j$ za $j = 1, \ldots, 4$. Sedaj lahko za vsak par točk določimo razliko med dolžino v grafu G_1 in dolžino ustrezne slike v G_2 in dobimo, da je

$$d_{\sigma 1}(G_1, G_2) = 2 \text{ in } d_{\sigma 1}(G_1, G_2) = 4.$$

Tako je $d_{\text{pot}}(G_1, G_2) = 2$.

Dolžina poti je mera

Iz same definicije opazimo, da velja $d_{pot}(G_1, G_2) = d_{pot}(G_2, G_1)$. Iz enačbe (9.1) takoj dobimo, da je $d_{pot}(G_1, G_2) = 0$ za izomorfne grafe. Po drugi strani $d_{pot}(G_1, G_2) = 0$ pomeni, da obstaja izomorfizem $\phi : V(G_1) \to V(G_2)$.
Ostane nam še trikotniška neenakost. Naj bodo G_1, G_2 in G_3 povezani grafi z $|V(G_1)| = |V(G_2)| = |V(G_3)|$ in $\alpha : V(G_1) \to V(G_2)$, ter $\beta : V(G_2) \to V(G_3)$ bijekciji z $d_\alpha(G_1, G_2) =$

$d_{pot}(G_1, G_2)$ in $d_\beta(G_2, G_3) = d_{pot}(G_2, G_3)$. Potem je $\beta \circ \alpha : V(G_1) \to V(G_3)$ tudi bijekcija in velja

$$\begin{aligned}
d_{pot}(G_1, G_3) &\leq d_{\beta\circ\alpha}(G_1, G_3) \\
&= \sum_{\{u,v\}\in V(G_1)\times V(G_1)} |d_{G_1}(u,v) - d_{G_3}((\beta\circ\alpha)(u), (\beta\circ\alpha)(v))| \\
&\leq \sum_{\{u,v\}\in V(G_1)\times V(G_1)} |d_{G_1}(u,v) - d_{G_2}(\alpha(u), (\alpha(v))| \\
&+ \sum_{\{u,v\}\in V(G_1)\times V(G_1)} |d_{G_2}(\alpha(u), \alpha(v)) - d_{G_3}((\beta\circ\alpha)(u), (\beta\circ\alpha)(v))| \\
&= d_\alpha(G_1, G_2) + d_\beta(G_2, G_3) \\
&= d_{pot}(G_1, G_2) + d_{pot}(G_2, G_3).
\end{aligned}$$

Izračun dolžine poti

Računanje dolžine poti pri dveh grafih poteka v treh korakih. Najprej izračunamo poti vseh parov točk v obeh grafih. To je v resnici za vse točke problem najkrajše poti v obeh grafih. Potem izračunamo σ-razdalje za dane bijekcije σ v času $O(n^2)$. Na koncu pa moramo določiti le še minimalno bijekcijo glede na dolžino poti.

9.2.3 Največji skupni podgrafi

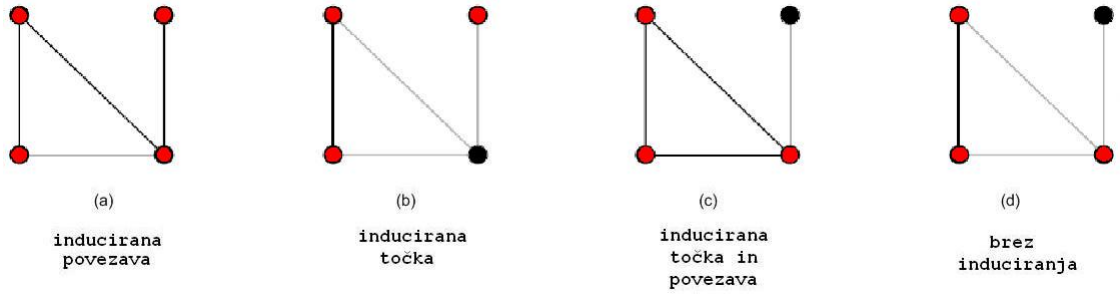
V tem razdelku gledamo mero podobnosti glede na velikost največjega skupnega podgrafa. Spomnimo se definicije induciranih podgrafov. Graf $G' = (V', E')$ je *podgraf* grafa $G = (V, E)$, če $V' \subseteq V$ in $E' \subseteq E$. *Inducirani podgraf* je takrat, ko E' vsebuje vse povezave $e \in E$, ki povezujejo točke v V'. Inducirani podgraf grafa G dobimo z odstranjevanjem točk iz $V(G)$. Ko odstranimo točko, izginejo tudi povezave, ki imajo to točko za krajišče.

Definicija 9.23 Naj bosta G_1 in G_2 neusmerjena grafa. Če obstaja inducirani podgraf $G_2' \subseteq G_2$, da velja $|V(G_2')| = |V(G_1)|$ in ϕ je izomorfizem grafov med G_1 in G_2', potem injektivno funkcijo $\phi : V(G_1) \to V(G_2)$ imenujemo *podgraf izomorfizma* od G_1 do G_2,

Definicija 9.24 Naj bosta G_1 in G_2 neusmerjena grafa. Graf S je *skupni inducirani podgraf* grafa G_1 in G_2, če obstaja podgraf izomorfizma iz S do G_1 in G_2.

Definicija 9.25 Naj bosta G_1 in G_2 neusmerjena grafa. Skupni inducirani podgraf S od G_1 in G_2 je največji, če ne obstaja noben drug skupni podgraf, ki bi imel več točk kot S. *Največji skupni inducirani podgraf* označimo z $\text{mcis}(G_1, G_2)$ (maximum common induced subgraph).

Poleg (točkovno) induciranega podgrafa poznamo tudi povezavno inducirani podgraf. Graf $G' = (V', E')$ je *povezavno inducirani podgraf* grafa $G = (V, E)$, če $E' \subseteq E$ in V' vsebuje točke povezav v E'. Povezavno inducirani graf ne vsebuje izoliranih točk. Slika 9.11 prikazuje primerjavo med točkovno in povezavno indiciranimi grafi na enostavnem primeru. Prejšnje definicije za inducirane grafe lahko prenesemo tudi na povezavno inducirane grafe.



Slika 9.11: Primerjava med točkovno in povezavno indiciranimi grafi

Definicija 9.26 Naj bosta G_1 in G_2 neusmerjena grafa. Če obstaja povezavno inducirani podgraf $G_2' \subseteq G_2$, da velja $|V(G_2')| = |V(G_1)|$ in ϕ je izomorfizem grafov med G_1 in G_2', potem injektivno funkcijo $\phi : V(G_1) \to V(G_2)$ imenujemo *podgraf izomorfizma* od G_1 do G_2,

Definicija 9.27 Naj bosta G_1 in G_2 neusmerjena grafa. Graf S je *skupni povezavno inducirani podgraf* grafa G_1 in G_2, če obstaja podgraf povezav izomorfizma iz S do G_1 in G_2.

Definicija 9.28 Naj bosta G_1 in G_2 neusmerjena grafa. Skupni povezavno inducirani podgraf S od G_1 in G_2 je največji, če ne obstaja noben drug skupni povezavni podgraf, ki bi imel več točk kot S. *Največji skupni povezavno inducirani podgraf* označimo z $\mathrm{mces}(G_1, G_2)$ (maximum common edge subgraph).

Opomba 9.29 Maksimalni skupni podgrafi po definiciji niso enolični ali povezani. MCIS ali MCES za neprazne grafe je sestavljen z vsaj ene točke ali ene povezave. Inducirani podgrafi se uporabljajo pri definiciji mer razdalj pri grafih.

Definicija 9.30 Naj bosta G_1 in G_2 neusmerjena grafa. MCIS razdaljo definiramo kot

$$d_{\mathrm{mcis}}(G_1, G_2) = 1 - \frac{|V(\mathrm{mcis}(G_1, G_2))|}{\max(|V(G_1)|, |V(G_2)|)} \tag{9.2}$$

in razdaljo MCES kot

$$d_{\mathrm{mces}}(G_1, G_2) = 1 - \frac{|V(\mathrm{mces}(G_1, G_2))|}{\max(|V(G_1)|, |V(G_2)|)}. \tag{9.3}$$

Računanje MCIS in MCES

Iskanje največje skupnega podgrafa je NP-poln problem [22]. Kjub temu je predlaganih nekaj natančnih algoritmov, ki temeljijo na iskanju vseh podgrafov ali razmerju med maksimalnimi skupnimi podgrafi in največji skupni kliki.

Prvi način je predlagal McGregor [23] in je zelo podoben "search-and-backtrack" pristopu pri iskanju izomorfizmov grafa. Algoritem najde skupne podgrafe s tem, da začne pri eni sami točki v vsakem grafu in iterativno dodaja točke (in incident povezave), ki ne kršijo pogojev skupnega podgrafa. Če ni več mogoče dodati nobeno novo točko, se velikost trenutnega podgrafa primerja s prejšnjim najdenim in "backtracking" se nadaljuje

za testiranje nove veje iskalnega drevesa. Na koncu algoritma dobimo največji skupni podgraf.

Drugi način iskanja temelji na dejstvu, da MCIS dveh grafov pripada največji kliki modularnemu produktu grafov. Klika je polno povezan podgraf. Maksimalna klika je največji induciran podgraf, ki je klika. Maksimalna klika ni nujno enolična. Modularni produkt $G_1 \diamond G_2$ grafa G_1 in G_2 je definiran na množici točk

$$V(G_1 \diamond G_2) = V(G_1) \times V(G_2)$$

in dve točki $(u_i, v_i), (u_j, v_j) \in G_1 \diamond G_2$ sta sosednji, če velja ali

$$(u_i, u_j) \in E(G_1) \text{ in } (v_i, v_j) \in E(G_2)$$

ali

$$(u_i, u_j) \notin E(G_1) \text{ in } (v_i, v_j) \notin E(G_2).$$

V skladu s tem, MCES dveh grafov ustreza največji kliki v modularnem produktu. [20]

Poglavje 10

Spektralna teorija grafov

Katarina Stupica, Teja Šegula, Jaka Špeh

Strukturne značilnosti grafov lahko raziskujemo s pomočjo računanja lastnih vrednosti nekaterih matrik. V spektralni teoriji se največkrat pojavljajo matrika sosednosti, Laplaceova matrika in normalizirana Laplaceova matrika. S pomočjo spektra teh matrik lahko nekaj povemo o podgrafih, povezanosti, dvodelnosti, diametru in kromatičnem številu grafa. Pomembne pa so tudi metode, s katerimi čim hitreje izračunamo spekter.

10.1 Temeljne lastnosti

V tem razdelku bomo predstavili tri različne matrike, ki so pomembne v spektralni teoriji in nekaj osnovnih lastnosti grafa, ki jih lahko dobimo iz njihovih spektrov. Pokazali bomo, da je koristno izračunati spektre vseh matrik, saj samo iz ene matrike ne moremo dobiti vseh lastnosti grafa.

10.1.1 Spekter grafa

Definicija 10.1 Naj bo $G = (V, E)$ poljuben graf (lahko tudi usmerjen multigraf) z vozlišči $V = \{1, \ldots, n\}$. *Matrika sosednosti* $A = (a_{ij})_{i,j}$ ima elemente

$$a_{ij} := \text{večkratnost povezave } (i, j) \in E.$$

Spekter grafa je spekter njegove matrike sosednosti.

Primer izračuna matrike sosednosti na usmerjenem multigrafu je na sliki 10.1. Matrika sosednosti je očitno odvisna od številčenja vozlišč. To pa ne velja za njen spekter, saj se ob menjavi vozlišča i in j v matriki sosednosti A zamenjata i-ta in j-ta vrstica ter i-ti in j-ti stolpec, kar ne vpliva na izračun $\det(A - \lambda I_n)$.

V tem poglavju se bomo večinoma ukvarjali z enostavnimi, neusmerjenimi grafi brez zank. Matrika sosednosti za take grafe je simetrična 0/1 matrika. Spomnimo se linearne algebre, ki pove, da ima realna simetrična matrika realen spekter, ortonormirane lastne vektorje ter se jo da diagonalizirati. To torej velja tudi za matriko sosednosti.

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$

Slika 10.1: Primer izračuna matrike sosednosti.

Slika 10.2: Primer grafa z utežmi, ki predstavljajo komponente lastnega vektorja.

Kako bi lahko interpretirali lastni par (λ, x) matrike A? Seveda velja $Ax = \lambda x$, vendar lahko na to enačbo pogledamo tudi drugače. Naj bo $w \in \mathbb{C}^n$ poljuben vektor in $\omega : V \to \mathbb{C}$ funcija, ki preslika vsak $i \in V$ v komponento $w_i = \omega(i)$. Množico sosedov vozlišča i označimo z $N(i)$. Tako lahko i-to komponento vektorja Aw zapišemo kot $\sum_{j \in N(i)} \omega(j)$. Torej velja, da ima A lastno vrednost λ natanko tedaj, ko obstaja neničelna utežna funkcija $\omega : V \to \mathbb{C}$, da za vsak $i \in V$ velja $\lambda\omega(i) = \sum_{j \in N(i)} \omega(j)$. V našem primeru je matrika sosednosti simetrična, zato se lahko omejimo na realne utežne funkcije. Predpostavimo lahko tudi to, da je maksimalna teža nenegativna. Če bi namreč veljalo $\max\{\omega(i);\ i \in V\} < 0$, potem bi bilo $\omega(i) < 0$ za vsak $i \in V$ in bi lahko namesto ω vzeli kar $-\omega$. Da bo zgornja interpretacija bolj jasna, si jo lahko ogledamo na primeru iz slike 10.2.

$$\begin{aligned}
-2 \cdot \omega(1) &= -2 = -1 + 0 - 1 = \omega(2) + \omega(3) + \omega(4) \\
-2 \cdot \omega(2) &= 2 = 1 + 0 + 1 = \omega(1) + \omega(3) + \omega(5) \\
-2 \cdot \omega(3) &= 0 = 1 - 1 - 1 + 1 = \omega(1) + \omega(2) + \omega(4) + \omega(5) \\
-2 \cdot \omega(4) &= 2 = 1 + 0 + 1 = \omega(1) + \omega(3) + \omega(5) \\
-2 \cdot \omega(5) &= -2 = -1 + 0 - 1 = \omega(2) + \omega(3) + \omega(4)
\end{aligned}$$

Zgornji izračuni nam povedo, da je ena lastna vrednost matrike sosednosti grafa na sliki 10.2 enaka -2. S pomočjo take interpretacije lastnih vrednosti lahko dokažemo naslednjo trditev.

Trditev 10.2 *Naj bo $G = (V, E)$ graf z n vozlišči in maksimalno stopnjo Δ. Naj ima*

njegova matrika sosednosti A lastne vrednosti $\lambda_1 \leq \ldots \leq \lambda_n$. Potem velja:

1. $\lambda_n \leq \Delta$.
2. *Če je G unija disjunktnih grafov G_1 in G_2, potem je* $\text{spekter}(G) = \text{spekter}(G_1) \cup \text{spekter}(G_2)$.
3. *Če je G cikel, potem je* $\text{spekter}(G) = \{2\cos(\frac{2\pi k}{n});\ k = 1, \ldots, n\}$.
4. *Če je $G = K_{n_1,n_2}$, kjer je $n_1 + n_2 = n$, je $\lambda_1 = -\sqrt{n_1 n_2}, \lambda_2 = \cdots = \lambda_{n-1} = 0, \lambda_n = \sqrt{n_1 n_2}$.*
5. *Če je G poln graf K_n, potem je $\lambda_1 = \cdots \lambda_{n-1} = -1$ in $\lambda_n = n - 1$.*

Dokaz. 1. Naj bo ω neničelna utežna funkcija, za katero velja $\lambda_n \omega(i) = \sum_{j \in N(i)} \omega(j)$ za vsak $i \in V$. Naj bo i_0 vozlišče z maksimalno težo. Potem velja $\lambda_n \omega(i_0) = \sum_{j \in N(i_0)} \omega(j) \leq \Delta\omega(i_0)$, od koder sledi $\lambda_n \leq \Delta$.

2. ($\subseteq$) Naj bo ω neničelna utežna funkcija za lastno vrednost $\lambda \in \text{spekter}(G)$. Ker ω ni identično enaka nič, potem mora biti ω neničelna na G_1 ali G_2, torej je ω zožena na G_1 ali G_2 utežna funkcija za $\lambda \in \text{spekter}(G_1)$ ali $\lambda \in \text{spekter}(G_2)$.
($\supseteq$) Naj bo sedaj ω neničelna utežna funkcija za lastno vrednost $\lambda \in \text{spekter}(G_1)$ (podobno za G_2). Če sedaj razširimo ω tako, da je $\omega(i) = 0$ za vse $i \in V \setminus V(G_1)$, dobimo neničelno utežno funkcijo za $\lambda \in \text{spekter}(G)$.

3. Za dokaz te točke začasno uporabimo kompleksne uteži. Naj bodo povezave cikla $(1, n)$ in $(i, i+1)$ za $i = 1, \ldots, n-1$. Za vsak $k = 1, \ldots, n$ naj bo $\tau_k := e^{2\pi i k/n}$. Definirajmo kompleksno utežno funkcijo $\omega(j) := \tau_k^{j-1}$. Potem je za vsak $j \in V$

$$\sum_{\ell \in N(j)} \omega(\ell) = \omega(j-1) + \omega(j+1) = \tau_k^{j-2} + \tau_k^j = (\tau_k^{-1} + \tau_k) \cdot \tau_k^{j-1} = (\tau_k^{-1} + \tau_k) \cdot \omega(j).$$

Torej je $\tau_k^{-1} + \tau_k = e^{-2\pi i k/n} + e^{2\pi i k/n} = 2\cos(\frac{2\pi k}{n})$ lastna vrednost grafa G.

4. Ker je matrika A simetrična, jo lahko diagonaliziramo. Torej je A podobna diagonalni matriki, ki ima lastne vrednosti po diagonali. Zato ima isti rang kot diagonalna matrika, ta rang pa je enak številu neničelnih lastnih vrednosti. Matrika sosednosti za graf K_{n_1,n_2} je enaka $\begin{pmatrix} A_1 & A_2 \\ A_2 & A_1 \end{pmatrix}$, kjer je A_1 matrika samih ničel, A_2 pa matrika samih enic. Očitno je njen rang enak 2, torej ima A samo dve neničelni lastni vrednosti. Iz linearne algebre vemo, da je vsota lastnih vrednosti matrike enaka sledi matrike. Torej je $\lambda_1 + \lambda_n = 0$ in lastne vrednosti matrike A so $\lambda_1 = -c, \lambda_2 = \cdots = \lambda_{n-1} = 0, \lambda_n = c$ za neko število c. Poglejmo sedaj karakteristični polinom $\det(A - \lambda I_n) = (-c - \lambda)\lambda^{n-2}(c - \lambda) = \lambda^n - c^2\lambda^{n-2}$. Spomnimo se, kako izračunamo determinanto matrike.

$$\det A = \sum_{\pi \in S_n} \text{sign}(\pi) \cdot a_{1,\pi(1)} \cdot \cdots \cdot a_{n,\pi(n)}$$

Prvi del vsote v karaterističnem polinomu se pojavi, če vzamemo permutacijo, kjer izberemo vse elemente na diagonali. Drugi del se pojavi, če izberemo $n - 2$ diagonalnih

elementov in 2 izven diagonale, ki sta enaka 1. Ko izberemo ta dva elementa, natanko določimo permutacijo. To lahko naredimo na $n_1 \cdot n_2$ načinov. Torej je takih permutacij c^2 (vsaka je z negativnim predznakom) in velja $c^2 = n_1 n_2$. Tako velja $\lambda_1 = -\sqrt{n_1 n_2}$ in $\lambda_n = \sqrt{n_1 n_2}$.

5. Matrika sosednosti grafa K_n je enaka $A = J - I_n$, kjer je J matrika samih enic. Naj bo λ lastna vrednost matrike J, torej $Jx = \lambda x$. Sledi $(J - I_n)x = (\lambda - 1)x$. Torej je lastna vrednost matrike A enaka $\lambda - 1$. Izračunajmo lastne vrednosti matrike J. Ker ima matrika J rang 1, ima $n - 1$ ničelnih lastnih vrednosti. Ker je sled matrike J enaka n, je njena neničelna lastna vrednost enaka n. Zato so lastne vrednosti matrike A enake $\lambda_1 = \cdots = \lambda_{n-1} = -1, \lambda_n = n - 1$.

□

Dokazano je bilo tudi, da je $\lambda_n = \Delta$ natanko tedaj, ko ima graf G Δ-regularno komponento. Poleg tega sta najmanjša in največja lastna vrednost povezani, saj velja $\lambda_1 \geq -\lambda_n$. Enakost velja natanko tedaj, ko ima graf dvodelno komponento z največjo lastno vrednostjo, ki je enaka λ_n [208].

S pomočjo spektra matrike sosednosti lahko v grafu tudi preštejemo število sprehodov določene dolžine.

Trditev 10.3 *Naj bo G usmerjen multigraf, ki ima lahko tudi zanke in A njegova matrika sosednosti z lastnimi vrednostmi λ_i. Potem velja:*

1. $(A^k)_{ij}$ = *število (i,j)-sprehodov dolžine k.*
2. *Lastne vrednosti matrike A^k so λ_i^k.*

Dokaz. 1. To točko lahko pokažemo z indukcijo na k. Za $k = 1$ dobimo kar matriko sosednosti, na (i,j)-tem mestu je seveda število sprehodov dolžine 1 med vozliščema i in j. Predpostavimo sedaj, da trditev drži za A^{k-1}. Ker je $A^k = A^{k-1}A$, velja $A^k_{ij} = \sum_{\ell=1}^{n} A^{k-1}_{i\ell} A_{\ell j}$. Tako dobimo vse sprehode dolžine $k-1$ od vozlišča i do ℓ, pomnožene s sprehodi dolžine 1 od vozlišča ℓ do j. Skupaj torej dobimo število sprehodov dolžine k od vozlišča i do j.

2. Za lastni par (λ_i, x_i) matrike A velja $Ax_i = \lambda_i x_i$. Od tod sledi $A^k x_i = \lambda_i^k x_i$, torej je λ_i^k lastna vrednost matrike A^k.

□

Z uporabo te trditve lahko pokažemo naslednjo posledico.

Posledica 10.4 *Naj ima matrika sosednosti grafa G lastne vrednosti λ_i. Potem velja:*

1. $\sum_{i=1}^{n} \lambda_i$ = *število zank v G.*
2. $\sum_{i=1}^{n} \lambda_i^2 = 2 \cdot |E|$.
3. $\sum_{i=1}^{n} \lambda_i^3 = 6 \cdot$ *število trikotnikov grafa G.*

Dokaz. 1. Spomnimo se, da je vsota lastnih vrednosti ravno sled matrike. V matriki sosednosti pa se na diagonali pojavlja ravno število zank za posamezno vozlišče.

2. Vsota kvadratov lastnih vrednosti je ravno sled matrike A^2, na njeni diagonali pa se pojavljajo sprehodi dolžine 2, ki se začnejo in končajo v istem oglišču. Če preštejemo vse povezave v grafu, bomo dobili ravno take sprehode, pri tem vsak sprehod štejemo dvakrat.

3. Vsota kubov lastnih vrednosti je sled matrike A^3, na njeni diagonali so ravno cikli dolžine 3. Če preštejemo število trikotnikov v grafu, dobimo ravno take cikle. Sprehod je določen z izbiro prvega oglišča (3 možnosti) in izbiro smeri (2 možnosti). Vsak trikotnik torej štejemo $2 \cdot 3 = 6$-krat.

□

S pomočjo spektra grafa pa lahko določimo tudi dvodelnost grafa.

Trditev 10.5 *Graf G je dvodelen natanko tedaj, ko se lastne vrednosti njegove matrike sosednosti pojavljajo v takih parih λ, λ', da je $\lambda = -\lambda'$.*

Dokaz. ($\Rightarrow$) Naj bo ω neničelna utežna funkcija za lastno vrednost λ grafa G. Ker je graf dvodelen, lahko njegova vozlišča razbijemo na dva dela V_1 in V_2. Naj bo $\omega' : V \to \mathbb{R}$ definirana kot

$$i \mapsto \begin{cases} \omega(i), & \text{če } i \in V_1 \\ -\omega(i), & \text{če } i \in V_2. \end{cases}$$

Potem za vsa vozlišča $i \in V_1$ velja

$$-\lambda\omega'(i) = -\lambda\omega(i) = -\sum_{j \in N(i)} \omega(j) = \sum_{j \in N(i)} \omega'(j),$$

za vsa vozlišča $i \in V_2$ pa

$$-\lambda\omega'(i) = \lambda\omega(i) = \sum_{j \in N(i)} \omega(j) = \sum_{j \in N(i)} \omega'(j).$$

Torej je tudi $-\lambda$ lastna vrednost grafa G.
($\Leftarrow$) Če velja $\lambda_i = -\lambda_j$, potem je za vsak lih k tudi $\lambda_i^k = -\lambda_j^k$. Sledi, da je $\mathrm{sl}(A^k) = \sum_{i=1}^k \lambda_i^k = 0$. Sled matrike A^k pa šteje število ciklov dolžine k v grafu G. Graf torej nima lihih ciklov, zato je dvodelen.

□

Videli smo, da lahko nekatere lastnosti grafa dobimo kar iz spektra matrike sosednosti. Dva izomorfna grafa imata očitno isti spekter, obratno pa ne moremo trditi. Na sliki 10.3 vidimo dva neizomorfna grafa, ki imata iste lastne vrednosti $\lambda_1 = -2, \lambda_2 = \lambda_3 = \lambda_4 = 0$ in $\lambda_5 = 2$. Tudi nekaterih drugih strukturnih značilnosti ne moremo določiti s pomočjo spektra matrike sosednosti, na primer povezanosti. V ta namen raziskujemo tudi spekter drugih matrik, ki jih lahko dobimo iz grafa.

Slika 10.3: Dva neizomorfna grafa z istim spektrom.

10.1.2 Laplaceova matrika

Definicija 10.6 Naj bo $G = (V, E)$ neusmerjen multigraf (lahko tudi z zankami) z matriko sosednosti A. Naj bo $D = \mathrm{diag}(d(1), \ldots, d(n))$ diagonalna matrika stopenj vozlišč. *Laplaceova matrika* L je definirana kot $L := D - A$. Če je G enostaven neusmerjen graf, potem so elementi matrike L enaki

$$l_{ij} = \begin{cases} -1, & \text{če } (i,j) \in E \\ d(i), & \text{če } i = j \\ 0 & \text{sicer.} \end{cases}$$

Na Laplaceovo matriko lahko pogledamo tudi drugače. *Orientacija* σ grafa G je preslikava, ki vsaki povezavi $e = (i, j)$ priredi smer, tako da pove, katero vozlišče je začetek povezave. *Incidenčna matrika* B orientiranega grafa (G, σ) ima elemente

$$b_{i,e} = \begin{cases} 1, & \text{če je } i \text{ začetek povezave } e \\ -1, & \text{če je } i \text{ konec povezave } e \\ 0 & \text{sicer.} \end{cases}$$

Pokažemo lahko, da neodvisno od izbire orientacije σ velja $L = BB^T$. Velja tudi naslednja lema.

Lema 10.7 *Za vsak* $x \in \mathbb{C}^n$ *velja*

$$x^T L x = x^T B B^T x = \sum_{(i,j)\in E} (x_i - x_j)^2.$$

Dokaz. Ker vemo, da velja $L = BB^T$, lahko zapišemo $x^T LX = x^T BB^T x = (B^T x)^T (B^T x)$. Vsaki povezavi (i, j)(vrstici v matriki B^T) pa pripada $\pm(x_i - x_j)$ v vektorju $B^T x$. Torej je

$$(B^T x)^T (B^T x) = \sum_{(i,j)\in E} (x_i - x_j)^2.$$

□

Ker je matrika sosednosti A realna in simetrična, to velja tudi za matriko L. Naj bodo lastne vrednosti matrike L $\lambda_1(L) \leq \cdots \leq \lambda_n(L)$. Spet lahko lastne vektorje $x \in \mathbb{R}^n$

interpretiramo z utežno funkcijo $\omega : V \to \mathbb{R}$, $i \mapsto x_i$. Velja, da je λ lastna vrednost matrike L, če obstaja taka neničelna utežna funkcija ω, da za vsak $i \in V$ velja

$$\lambda\omega(i) = \sum_{j \in N(i)} (\omega(i) - \omega(j)),$$

saj je $\sum_{j \in N(i)} \omega(i) = \omega(i) \cdot d(i)$. Spet lahko predpostavimo, da je maksimalna utež nenegativna. Če na zgornjo enačbo pogledamo za $i \in V$ z maksimalno težo, opazimo, da je $\lambda\omega(i) \geq 0$. Torej so vse lastne vrednosti nenegativne. Če za utežno funkcijo vzamemo $\omega \equiv 1$, dobimo $\lambda = \lambda\omega(i) = \sum_{j \in N(i)} (\omega(i) - \omega(j)) = 0$. Torej je ena lastna vrednost enaka 0, njen lastni vektor pa je vektor samih enic.

Spekter Laplaceove matrike nam odkrije nekaj novih lastnosti grafa, med njimi tudi povezanost.

Trditev 10.8 *Naj ima graf G Laplaceovo matriko L. Graf je sestavljen iz k povezanih komponent natanko tedaj, ko velja $\lambda_1(L) = \cdots = \lambda_k(L) = 0$ in $\lambda_{k+1}(L) > 0$.*

Dokaz. ($\Rightarrow$) Naj bo B incidenčna matrika grafa G za poljubno orientacijo. Za vsako komponento C grafa G definiramo $z(C) \in \mathbb{R}^n$

$$z(C)_i := \begin{cases} 1, & \text{če } i \in V(C) \\ 0, & \text{sicer.} \end{cases}$$

Potem je množica $Z := \{z(C);\ C \text{ je komponenta grafa } G\}$ linearno neodvisna, ker imamo k povezanih komponent, ki so med seboj ločene. Velja tudi $Lz(C) = BB^T z(C) = 0$. Imamo torej k linearno neodvisnih lastnih vektorjev za lastno vrednost 0.
($\Leftarrow$) Naj velja, da je $z \in \mathbb{R}^n$ tak, da je $Lz = BB^T z = 0$. Potem iz $z^T BB^T z = 0$ sledi $B^T z = 0$, kar pomeni, da mora biti z konstanten na vsaki povezani komponenti. Torej je z linearna kombinacija elementov iz množice Z in imamo toliko povezanih komponent kot je linearno neodvisnih lastnih vektorjev za lastno vrednost 0.

□

S pomočjo Laplaceove matrike lahko določimo tudi število vpetih dreves, kar je omenjeno v [208], [210] in [221].

Izrek 10.9 *Naj bo G graf z Laplaceovo matriko L. Potem velja:*

1. *Število vpetih dreves v grafu G je enako* $\det(L_i)$*, pri tem je L_i podmatrika matrike L, kjer izbrišemo i-to vrstico in i-ti stolpec.*
2. *Število vpetih dreves je enako*
$$\frac{1}{n} \prod_{i \geq 2} \lambda_i(L).$$

Čeprav lahko s pomočjo Laplaceove matrike določimo dodatne lastnosti grafa, pa ne moremo določiti nekaterih drugih lastnosti, na primer dvodelnosti. To lahko vidimo na primeru iz slike 10.4, kjer sta dva grafa z istim Laplaceovim spektrom, desni je dvodelen, levi pa ne. Radi bi imeli oboje, povezanost in dvodelnost grafa, zato nam tukaj pride prav normalizirana Laplaceova matrika.

Slika 10.4: Dva grafa z istim Laplaceovim spektrom.

10.1.3 Normalizirana Laplaceova matrika

Definicija 10.10 *Normalizirana Laplaceova matrika* $\mathcal{L}$ grafa G je definirana kot $\mathcal{L} = D^{-1/2}LD^{-1/2}$. Pri tem je $D^{-1/2}$ diagonalna matrika, kjer je na i-tem mestu $d(i)^{-1/2}$, če je $d(i) > 0$ in 0 sicer. Če je G enostaven graf, ima $\mathcal{L}$ elemente:

$$l_{ij} = \begin{cases} 1, & \text{če } i = j \text{ in } d(i) > 0 \\ -\frac{1}{\sqrt{d(i)d(j)}}, & \text{če } (i,j) \in E \\ 0 & \text{sicer.} \end{cases}$$

λ je lastna vrednost matrike $\mathcal{L}$ natanko tedaj, ko obstaja neničelna funkcija $\omega : V \to \mathbb{C}$, da velja

$$\lambda\omega(i) = \frac{1}{\sqrt{d(i)}} \sum_{j \in N(i)} \left(\frac{\omega(i)}{\sqrt{d(i)}} - \frac{\omega(j)}{\sqrt{d(j)}} \right) \text{ za vsak } i \in V.$$

V zvezi z Laplaceovo matriko je znan naslednja trditev, dokaz lahko najdemo v [209].

Trditev 10.11 *Naj bo G graf z normalizirano Laplaceovo matriko $\mathcal{L}$. Potem velja:*

1. $\lambda_1(\mathcal{L}) = 0,\ \lambda_n(\mathcal{L}) \leq 2$.

2. *G je dvodelen natanko tedaj, ko je za vsako lastno vrednost $\lambda(\mathcal{L})$ tudi $2-\lambda(\mathcal{L})$ lastna vrednost matrike $\mathcal{L}$.*

3. *Če je $\lambda_1(\mathcal{L}) = \cdots = \lambda_k(\mathcal{L}) = 0$ in $\lambda_{k+1}(\mathcal{L}) \neq 0$, potem ima G natanko k povezanih komponent.*

10.1.4 Primerjava spektrov

Če je G d-regularen graf, potem lahko primerjamo spektre matrik A, L in $\mathcal{L}$. Če je spekter$(A) = \{\lambda_1, \ldots, \lambda_n\}$, potem je spekter$(L) = \{d - \lambda_n, \ldots, d - \lambda_1\}$ in spekter$(\mathcal{L}) = \{1 - \frac{\lambda_n}{d}, \ldots, 1 - \frac{\lambda_1}{d}\}$.
V splošnem pa ni preprostega razmerja med spektri, lahko samo omejimo lastne vrednosti Laplaceove matrike z lastnimi vrednostmi matrike sosednosti. Za to potrebujemo znan izrek iz linearne algebre, ki ga bomo samo navedli, dokaz pa lahko zasledimo v [211].

Izrek 10.12 *Courant-Fischerjev izrek. Naj bo $M \in \mathbb{R}^{n\times n}$ realna simetrična matrika z lastnimi vrednostmi $\lambda_1 \leq \ldots \leq \lambda_n$. Potem za vsak $k = 1, \ldots, n$ velja*

$$\lambda_k = \min_{\substack{U \leq \mathbb{R}^n \\ \dim(U) = k}} \max_{\substack{x \in U \\ x \neq \mathbf{0}}} \frac{x^T M x}{x^T x}.$$

Sedaj lahko dokažemo spodnjo trditev.

Trditev 10.13 *Naj bo A matrika sosednosti in L Laplaceova matrika grafa G. Naj bosta Δ in δ maksimalna in minimalna stopnja vozlišč v grafu G ter $\lambda_k(A)$ k-ta najmanjša lastna vrednost matrike A in $\lambda_{n+1-k}(L)$ k-ta največja lastna vrednost matrike L. Potem velja:*

$$\delta - \lambda_k(A) \leq \lambda_{n+1-k}(L) \leq \Delta - \lambda_k(A).$$

Dokaz. Dokažimo prvo neenakost, drugo dokažemo na podoben način. Ker je $\lambda_{n+1-k}(L)$ k-ta največja lastna vrednost matrike L, je $\delta - \lambda_{n+1-k}(L)$ k-ta najmanjša lastna vrednost matrike $\delta I_n - L = A - (D - \delta I_n)$. Ta matrika se od matrike A razlikuje le na diagonali, kjer odštevamo nenegativno vrednost $d(i) - \delta$. Definirajmo $r(x) := \frac{x^T(D-\delta I_n)x}{x^T x}$. Očitno velja, da je število $r(x) \geq 0$. Sedaj dvakrat uporabimo Courant-Fischerjev izrek in dobimo

$$\begin{aligned}
\delta - \lambda_{n+1-k}(L) &= \lambda_k(\delta I_n - L) \\
&= \lambda_k(A - (D - \delta I_n)) \\
&= \min_{\substack{U \leq \mathbb{R}^n \\ \dim(U) = k}} \max_{\substack{x \in U \\ x \neq \mathbf{0}}} \frac{x^T(A - (D - \delta I_n))x}{x^T x} \\
&= \min_{\substack{U \leq \mathbb{R}^n \\ \dim(U) = k}} \max_{\substack{x \in U \\ x \neq \mathbf{0}}} \frac{x^T A x}{x^T x} - r(x) \\
&\leq \min_{\substack{U \leq \mathbb{R}^n \\ \dim(U) = k}} \max_{\substack{x \in U \\ x \neq \mathbf{0}}} \frac{x^T A x}{x^T x} \\
&= \lambda_k(A).
\end{aligned}$$

□

Za kasnejšo uporabo navedimo še eno posledico in trditev v zvezi z največjo in drugo najmanjšo lastno vrednostjo Laplaceove matrike.

Posledica 10.14 *Za največjo lastno vrednost λ_n realne simetrične matrike $M \in \mathbb{R}^{n\times n}$ velja*

$$\lambda_n = \max_{\substack{x \in \mathbb{R}^n \\ x \neq \mathbf{0}}} \frac{x^T M x}{x^T x}.$$

Za drugo najmanjšo lastno vrednost Laplaceove matrike L velja

$$\lambda_2 = \min_{x \perp \mathbf{1}} \frac{x^T L x}{x^T x}.$$

Graf	spekter(A)	spekter(L)	spekter($\mathcal{L}$)
$G = P_n$	$2\cos(\frac{\pi k}{n+1})$, $k = 1, \ldots, n$	$2 - 2\cos(\frac{\pi(k-1)}{n})$, $k = 1, \ldots, n$	$1 - \cos(\frac{\pi(k-1)}{n-1})$, $k = 1, \ldots, n$
$G = C_n$	$2\cos(\frac{2\pi k}{n})$, $k = 1, \ldots, n$	$2 - 2\cos(\frac{2\pi k}{n})$, $k = 1, \ldots, n$	$1 - \cos(\frac{2\pi k}{n})$, $k = 1, \ldots, n$
$G = K_{1,n}$	$-\sqrt{n}, \sqrt{n}$, $0(n-2$-kratna$)$	$0, n$, $1(n-2$-kratna$)$	$0, 2$, $1(n-2$-kratna$)$
$G = K_{n_1,n_2}$	$-\sqrt{n_1 n_2}, \sqrt{n_1 n_2}$, $0(n-2$-kratna$)$	$0, n_1(n_2-1$-kratna$)$, $n_2(n_1-1$-kratna$), n$	$0, 2$, $1(n-2$-kratna$)$
$G = K_n$	$-1(n-1$-kratna$)$, $n-1$	$0, n(n-1$-kratna$)$	$0, \frac{n}{n-1}(n-1$-kratna$)$

Tabela 10.1: Primeri spektrov za elementarne grafe.

Trditev 10.15 *Naj bo L Laplaceova matrika grafa $G = (V, E)$ in $\lambda_1 \leq \ldots \leq \lambda_n$ njene lastne vrednosti. Potem velja*

$$\lambda_2(L) = n \min \left\{ \frac{\sum_{\{i,j\} \in E} (x_i - x_j)^2}{\sum_{\{i,j\} \in \binom{V}{2}} (x_i - x_j)^2};\ x \in \mathbb{R}^n \text{ nekonstanten} \right\}$$

in

$$\lambda_n(L) = n \max \left\{ \frac{\sum_{\{i,j\} \in E} (x_i - x_j)^2}{\sum_{\{i,j\} \in \binom{V}{2}} (x_i - x_j)^2};\ x \in \mathbb{R}^n \text{ nekonstanten} \right\}.$$

10.1.5 Primeri spektrov

V tabeli 10.1 so našteti spektri matrike sosednosti A, Laplaceove matrike L in normalizirane Laplaceove matrike $\mathcal{L}$ za nekatere osnovne primere grafov. Vsi grafi imajo n vozlišč.

Obstajajo pa tudi grafi, ki imajo isti spekter glede na vse tri vrste matrik, matrike sosednosti A, Laplaceove matrike L in normalizirane Laplaceove matrike $\mathcal{L}$. To lahko vidimo na primeru grafa na sliki 10.5.

10.2 Numerične metode

Če želimo uporabiti spekter poljubnega grafa, ga moramo izračunati. Pri tem se pojavi kup vprašanj. Kako dobiti lastne vrednosti? Kakšne metode imamo na voljo? Kako hitre in prostorsko zahtevne so?

Slika 10.5: Dva grafa z istim spektrom glede na vse tipe matrik.

10.2.1 Metode za izračun lastnih vrednosti majhnih polnih matrik

Zatakne se že pri prvem vprašanju. Namreč ni jasno, kako za poljubno matriko M hitro izračunati celotni spekter. Eno pot nam pokaže dejstvo, da so lastne vrednosti ničle karakterističnega polinoma $\det(M - \lambda I_n)$. Ta pot nas stane $\mathcal{O}(n!)$, zato jo kaj hitro zavržemo. Spomnimo se, da se ubadamo z matrikami, ki so realne in (v primeru neusmerjenih grafov) simetrične. Simetrične matrike lahko s podobnostnimi transformacijami preoblikujemo v diagonalno matriko $M \to P^{-1}MP$. To nam porodi idejo. Če nas zanimajo samo lastne vrednosti in ne lastni vektorji, je dovolj preoblikovati M v trikotno matriko. (Matriko, ki ima vse elemente pod (ali nad) diagonalo enake nič.) V tem primeru so diagonalni elementi kar lastne vrednosti.

To storimo v dveh korakih. Najprej iterativno aproksimiramo P (in P^{-1}). Aproksimiramo ju s pomočjo produkta "atomskih" transformacij P_i, ki določene izvendiagonalne elemente postavijo na nič. (Poslužimo se lahko Jacobijeve transformacije, Givensovih rotacij in Hausholderjevih zrcaljenj.) Metode potrebujejo $\mathcal{O}(n^3)$ korakov. Rezultat je prehodna matrika $\widetilde{P}$, da je $M_1 = (m_{ij}) := \widetilde{P}^{-1}M\widetilde{P}$ tridiagonalna (v primeru simetrične M) oziroma Hessenbergova matrika (v splošnem).

Slika 10.6: Zgoraj vidimo transformacijo matrike M v (zgornjo) Hessenbergovo matriko, ki ima neničelne elemente v zgornjem trikotniku in na prvi poddiagonali. Potem je ilustriran isti korak za simetrično matriko M. Rezultat je tridiagonalna matrika M_1 ($m_{ij} = 0$, če $|i - j| > 1$).

Drugi korak je QR iteracija. Ideja je, da lahko za vsako realno matriko M' izračunamo

njen QR razcep $M' = QR$, kjer je Q ortogonalna in R zgornje trikotna matrika. Sedaj zmnožimo faktorje v obratnem vrstnem redu in dobimo $M'' := RQ = Q^T QRQ = Q^T M'Q$. Pri tem se zgornja Hessenbergova oblika ali simetričnost in tridiagonalnost ohranja. Algoritem je sestavljen iz zaporednih transformacij

$$M_i := Q_i R_i,$$
$$M_{i+1} := R_i Q_i = Q_i^T M_i Q_i.$$

Pravilnost pove naslednji izrek.

Izrek 10.16

1. *Če ima M lastne vrednosti, ki so različne po absolutnih vrednostih, potem M_s konvergira proti zgornje trikotni matriki, ko gre $s \to \infty$.*

2. *Če ima M lastno vrednost $|\lambda_i|$ večkratnosti p, potem M_s, ko gre $s \to \infty$, konvergira proti matriki, ki je skoraj zgornje trikotna. Vsebuje namreč diagonalni blok reda p, katerega lastne vrednosti konvergirajo proti λ_i.*

Dokaz najdemo v [227]. Izrek 10.16 nam omogoča, da matriko z lastno vrednostjo, ki ima večkratnost večjo kot 1, razcepimo na podmatrike in te posebej diagonaliziramo. Za tridiagonalne matrike ena QR iteracija potrebuje $\mathcal{O}(n)$ korakov. S premiki in implicitno QR metodo lahko dosežemo zadovoljivo konvergenco v $\mathcal{O}(n)$ iteracijah. Skupaj je zatevnost drugega koraka enaka $\mathcal{O}(n^2)$. Torej je zahtevnost izračuna vseh lastnih vrednosti enaka $\mathcal{O}(n^3)$. Več o izračunu lastnih vrednosti polnih matrik lahko preberemo v [228].

10.2.2 Metode za izračun nekaj lastnih vrednosti velike razpršene matrike

Če imamo opravka z matrikami, ki so zelo velike, časovna in prostorska zahtevnost postopka opisanega v prejšnjem razdelku skokovito narasteta. Izkaže se, da je v takem primeru matrika M pogosto razpršena in zadošča izračunati le nekaj (zunanjih) lastnih vrednosti.

Groba ideja je, da iz velike razpršene matrike konstruiramo manjšo matriko, za katero znamo hitro izračunati njene lastne vrednosti. Dobljene lastne vrednosti razglasimo za lastne vrednosti razpršene matrike.

Naj bo M simetrična matrika. Izberimo si poljuben normiran vektor x_1. Poglejmo si podprostor $\mathcal{M}$, določen z $(x_1, Mx_1, \ldots, M^{i-1}x_1)$, za $i \in \mathbb{N}$. Za prostor $\mathcal{M}$ konstruirajmo bazo $(x_1, x_2, \ldots, x_i)$. Naj bo X_i $n \times i$ matrika, ki ima za stolpce vektorje $x_1, x_2, \ldots, x_i$. Potem je $T = X_i^T M X_i$ tridiagonalna matrika. Njene lastne vrednosti so aproksimacija za i lastnih vrednosti M, ki se nahajajo na robu spektra. Opisali smo i-ti korak *Lanczosovega algoritma* [229]. Navadno ponovno zaženemo algoritem vsakih k korakov, za nek določen $k \ll n$, dokler nimamo zadovoljive konvergence.

Algoritem: Lanczosov algoritem.
Vhod: Matrika M.
Izhod: r lastnih vrednosti, ki se nahajajo na robu spektra M.

Slika 10.7: Iz velike razpršene matrike A konstruiramo manjšo matriko B. (Videli bomo, da je B v primeru nesimetrične A enaka zgornji Hessenbergovi matriki (zgoraj desno), v primeru simetrične A pa je B tridiagonalna (spodaj desno).)

1. *Začetek:* Izberi število korakov k, število lastnih vrednosti r in začetni vektor x_1. Naj bo $\beta_0 := x_1^T x_1$, $x_1 = x_1/\beta_0$.

2. *Lanczosov korak:*
   for $i = 1$ **to** k **do**
   $y := Mx_i$
   $\alpha_i := x_i^T y$
   $x_{i+1} := y - \alpha_i x_i - \beta_{i-1} x_{i-1}$
   $\beta_i := x_{i+1}^T x_{i+1}$
   $x_{i+1} := x_{i+1}/\beta_i$
   end
   Naj bo $X_i := [x_1, x_2, \ldots, x_i]$.

3. *Izračun lastnih vrednosti:* Izračunaj lastne vrednosti $T := X_i^T M X_i$.

4. *Test konvergence in ponovni zagon:* Če prvih r stolpcev T zadošča pogoju konvergence, potem vrni pripadajoče lastne vrednosti in končaj. Sicer ponovno zaženi algoritem s primerno izbranim x_1.

Za lastne vrednosti, ki ne ležijo na robu spektra M uporabimo metodo "premakni in obrni". Namesto, da računamo lastne vrednosti matrike M, računamo lastne vrednosti za matriko $(M - \mu I_n)^{-1}$, kjer je $\mu \in \mathbb{C}$. Sedaj bodo najprej skonvergirale lastne vrednosti, ki so v bližini μ.

Minimalno število iteracij potrebnih za izračun željene množice lastnih vrednosti je neznano. V praksi se izkaže, da je konvergenca relativno hitra.

Če nimamo simetrične matrike, lahko uporabimo *Arnoldijevo metodo* [230]. Razlika med njo in Lanczosovo metodo je v tem, da v ortogonalizaciji pri Arnoldiju uporabimo vse vektorje, pri Lanczosu pa le zadnje tri (zaradi simetričnosti). Zaradi tega je pri Arnoldiju matrika $T = X_i^T M X_i$ zgornja Hessenbergova (slika 10.7).

10.3 Podgrafi in operacije na grafih

Kaj nam spekter grafa pove o njegovih podgrafih? Ali so lastne vrednosti podgrafov grafa G zastopane v spektru G? Ali lahko na podlagi lastnih vrednosti sklepamo, da določen graf ni podgraf (ali vsaj induciran podgraf) danega grafa? Kaj se zgodi s spektrom dveh grafov, ki sta združena bodisi s kartezičnim bodisi z direktnim produktom?

V tem razdelku bomo pokazali ideje, kako odgovoriti na ta vprašanja. Obravnavali bomo samo spekter matrike sosednosti, čeprav veliko rezultatov velja tudi za Laplaceov spekter.

10.3.1 Izrek o prepletanju

Omejimo se na vprašanje: kakšna je povezava med spektrom grafa in spektrom njegovih induciranih podgrafov.

Slika 10.8: Grafi K_2, $K_{1,6}$ in K_4.

Zgled 10.17 Spekter K_2 je enak $\{-1, 1\}$. Jasno je, da je K_2 induciran podgraf v $K_{1,6}$ in K_4. Spekter $K_{1,6}$ je $\{-\sqrt{6}, 0, 0, 0, 0, \sqrt{6}\}$. Opazimo, da se lastne vrednosti K_2 ne pojavijo v spektru $K_{1,6}$. Po drugi strani pa velja, da je spekter K_4 enak $\{-1, -1, -1, 1\}$. Vidimo, da sta tu zastopani obe lastni vrednosti K_2. Torej ni nujno, da se lastne vrednosti induciranega podgrafa grafa G pojavijo v spektru G.

Vendar ni vse tako brezupno. Velja, da se lastne vrednosti prepletajo med seboj. Naj bo G graf na n vozliščih in naj bo H induciran podgraf G z $n-1$ vozlišči ($H := G - v$ za nek $v \in V(G)$). Naj bodo $\lambda_1, \lambda_2, \ldots, \lambda_n$ lastne vrednosti za matriko sosednosti grafa G in $\mu_1, \mu_2, \ldots, \mu_{n-1}$ lastne vrednosti za matriko sosednosti grafa H. Potem velja

$$\lambda_i \leq \mu_i \leq \lambda_{i+1}, \quad \forall i \in \{1, 2, \ldots, n-1\}.$$

Sledi, da če ima G lastno vrednost večkratnosti k, potem ima H lastno vrednost večkratnosti $k-1$.

Za induciran podgraf H na m vozliščih lahko z indukcijo posplošimo rezultat

$$\lambda_i \leq \mu_i \leq \lambda_{i+(n-m)}, \quad \forall i \in \{1, 2, \ldots, m\}. \tag{10.1}$$

Pokazali bomo splošnejši rezultat iz katerega bo sledilo 10.1.

Izrek 10.18 *Naj bosta $n, m \in \mathbb{N}$ in $S \in \mathbb{R}^{n \times m}$ taka matrika, da velja $S^T S = I_m$. Naj bo $A \in \mathbb{R}^{n \times n}$ simetrična matrika in $B := S^T A S$. Potem za lastne vrednosti A ($\lambda_1, \lambda_2, \ldots, \lambda_n$) in lastne vrednosti B ($\mu_1, \mu_2, \ldots, \mu_n$) velja lastnost prepletanja in sicer*

$$\lambda_i \leq \mu_i \leq \lambda_{i+(n-m)}, \quad \forall i \in \{1, 2, \ldots, m\}. \tag{10.2}$$

Kako iz izreka sledi enačba 10.1. Naj bo A matrika sosednosti grafa G. Matrika sosednosti induciranega podgrafa H je podmatrika A, ki jo dobimo tako, da izbrišemo i-to vrstico in i-ti stolpec A za vsako vozlišče $i \in V(G)\backslash V(H)$. Tako matriko lahko dobimo iz A na isti način, kot je v izreku 10.18 B dobljena iz A. Če so $i_1, i_2, \ldots, i_k$ vozlišča iz $V(G)\backslash V(H)$, potem za S^T izberemo matriko, ki ima izbrisamo i_j-to vrstico za $j = 1, 2, \ldots, k$.

Dokaz. Matrika S določa injektivno preslikavo iz $\mathbb{R}^m$ v $\mathbb{R}^n$. Za podmnožico $U \subseteq \mathbb{R}^m$ s $S(U)$ označimo sliko U preslikave S, zapisano kompaktno

$$S(U) := \{Su \mid u \in U\}.$$

Če je $U \subseteq \mathbb{R}^m$ i-dimenzionalni podprostor ($i \leq m$), potem je $S(U)$ prav tako i-dimenzionalni podprostor v $\mathbb{R}^n$, ker je S injektivna preslikava. Spomnimo se karakterizacije lastnih vrednosti iz izreka 10.12 . Za vsak $i \in \{1, 2, \ldots, m\}$ velja

$$\begin{aligned}
\lambda_i &= \min_{\substack{U\subseteq\mathbb{R}^n \\ \dim(U)=i}} \max_{\substack{x\in U \\ x\neq 0}} \frac{x^TAx}{x^Tx} \leq \min_{\substack{U\subseteq S(\mathbb{R}^m) \\ \dim(U)=i}} \max_{\substack{x\in U \\ x\neq 0}} \frac{x^TAx}{x^Tx} = \min_{\substack{U\subseteq\mathbb{R}^m \\ \dim(U)=i}} \max_{\substack{x\in S(U) \\ x\neq 0}} \frac{x^TAx}{x^Tx} \\
&= \min_{\substack{U\subseteq\mathbb{R}^m \\ \dim(U)=i}} \max_{\substack{x\in U \\ x\neq 0}} \frac{(Sx)^TA(Sx)}{(Sx)^T(Sx)} = \min_{\substack{U\subseteq\mathbb{R}^m \\ \dim(U)=i}} \max_{\substack{x\in U \\ x\neq 0}} \frac{x^T(S^TAS)x}{x^Tx} = \min_{\substack{U\subseteq\mathbb{R}^m \\ \dim(U)=i}} \max_{\substack{x\in U \\ x\neq 0}} \frac{x^TBx}{x^Tx} \\
&= \mu_i.
\end{aligned}$$

S tem smo pokazali prvo neenakost v 10.2.

Isti argumet uporabimo na matriki $-A$. Dobimo, da za vsak $k \in \{0, 1, \ldots, m-1\}$ velja $-\lambda_{n-k} \leq -\mu_{m-k}$, kar pomeni

$$\mu_{m-k} \leq \lambda_{n-k}.$$

Ko vstavimo $k := m - i$ dobimo še drugo neenakost v 10.2.

□

Zgled 10.19 Iz tabele 10.1 lahko preberemo, da so lastne vrednosti P_n enake

$$\left\{2\cos\left(\frac{\pi k}{n+1}\right) \mid k = 1, 2, \ldots, n\right\}.$$

Slika 10.9: Poti P_5, P_4, P_3 in P_2 (levo) ter njihove pripadajoče lastne vrednosti (desno).

Na sliki 10.9 (leva stran) so narisane poti P_5, P_4, P_3 in P_2. Jasno je, da je naslednik induciran graf predhodnika. Prav tako so na sliki (desna stran) numerično izračunane lastne vrednosti. Vidi se prepletanje, ki ga zagotavlja izrek 10.18.

Posledica 10.20 *Naj bodo $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ lastne vrednosti grafa G in naj bodo $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_m$ lastne vrednosti grafa H ($m \leq n$). Če je $\mu_1 < \lambda_1$ ali $\lambda_n < \mu_m$, potem H ni induciran podgraf grafa G.*

Posledica jasno sledi iz pred dokazanega izreka o prepletanju. Na sliki 10.10 je viden njen geometrijski pomen.

Slika 10.10: Lastne vrednosti G ležijo na intervalu med λ_1 in λ_n. Lastne vrednosti H ležijo na intervalu med μ_1 in μ_m. Na sliki so tri možnosti, kjer H ne more biti induciran podgraf grafa G. (Velja $\mu_1 < \lambda_1$ (rdeča in oranžna) oziroma $\mu_m < \lambda_n$ (modra in oranžna).)

Zgled 10.21 Naj bo G graf, ki ima vse lastne vrednosti strogo manjše od 2. Iz tabele 10.1 lahko preberemo, da imajo vsi grafi C_j ($j \in \mathbb{N}$) največjo lastno vrednost enako 2. Zato v G ne obstaja inducirani podgraf, ki bi bil cikel. Za vsak cikel v grafu G lahko najdemo enostaven cikel, iz katerega lahko konstruiramo induciran enostaven cikel. To pomeni, da G nima ciklov.

O G-ju lahko povemo še več. Nima vozlišča, ki bi imelo stopnjo strogo večjo kot 3. Saj bi v nasprotnem G vseboval induciran podgraf $K_{1,j}$ za $j \geq 4$, ki pa ima največjo lastno vrednost enako $\sqrt{j}$. To ni mogoče.

10.3.2 Grafting

Oglejmo si vlogo podgrafov še z drugega zornega kota. Bodita G in H grafa. Radi bi spremenili G v taki meri, da bi modificiran G vseboval nekatere lastne vrednosti H.

Prvi pristop je jasen. Vzamemo disjunktno unijo obeh grafov. Ni težko videti, da je spekter $G \cup H$ enak uniji spektra G in H (v mislih imamo spekter matrike sosednosti). Vendar ta pristop odpove, če hočemo ohranjati povezanost.

Postavimo si skromno zahtevo. Poizkusimo preoblikovati G tako, da mu dodamo samo eno lastno vrednost. Naj bo λ lastna vrednost H, ki jo želimo dodati, in x pripadajoči lastni vektor. Najprej obravnavajmo primer, ko obstaja i_0, da je $x_{i_0} = 0$. Konstruirajmo graf G' kot unijo G in H, kjer $i_0 \in V(G)$ identificiramo z nekim poljubnim vozliščem $j_0 \in V(G)$. Formalno definiramo G' kot

$$V(G') := V(G) \cup (V(H) \backslash \{i_0\}),$$
$$E(G') := E(G) \cup E(H - i_0) \cup \{\{j_0, i\} \mid i \in V(H), \{i_0, i\} \in E(H)\}.$$

Rečemo, da je H *grafted* v G vzdolž i_0 in j_0.

Poiščimo lastni vektor za λ v G'. Uporabimo kombinatorično interpretacijo lastnih parov. Za vsako vozlišče G v G' utež nastavimo na 0. Ostala vozlišča pa naj imajo enake uteži, kot so jih imele v grafu H (za lastno vrednost λ). Tako dobimo lastni vektor G' za

λ. Na sliki 10.11 je predstavljen primer za $G = K_3$ in $H = K_{1,4}$. Velja omeniti, da poleg λ dobimo še nekaj dodatnih lastnih vrednosti.

Slika 10.11: Na sliki je G' za $G = K_{1,4}$ in $H = K_3$. Vemo, da -1 leži v spektru matrike K_3. Novi graf G' ima lastno vrednost -1, čeprav je $K_{1,4}$ nima. Na sliki so označene uteži za novo lastno vrednost.

Dolžni smo še obravnavati primer, ko lastni vektor nima ničelne komponente. Izberimo si vozlišče $i_0 \in V(H)$ in naredimo dve kopiji H^+ in H^- grafa H. Vozlišča v kopijah označimo z i^+ (za H^+) in i^- (za H^-). Ustvarimo novo vozlišče i_1 in povežimo grafa H^+ ter H^- z dvema povezavama $\{i_0^+, i_1\}$ in $\{i_1, i_0^-\}$. Nastali graf označimo s $\widetilde{H}$.

Naj bo (λ, x) lastni par za graf H. Potem je vektor $\widetilde{x}$, definiran kot

$$\widetilde{x}_{i^+} := x_i \text{ in } \widetilde{x}_{i^-} := -x_i, \forall i \in V(H)$$

ter

$$\widetilde{x}_{i_1} := 0,$$

lastni vektor za λ v matriki sosednosti $\widetilde{H}$.

Zgled 10.22 Graf P_3 ima lastno vrednost $\sqrt{2}$ za lastni vektor $(1/\sqrt{2}, 1, 1/\sqrt{2})^T$. Za i_0 izberemo srednje vozlišče. Na sliki 10.12 je prikazana simetrična konstukcija $\widetilde{P_3}$.

V primeru samih neničelnih komponent lastnega vektorja za graf H, konstruiramo $\widetilde{H}$. Sedaj imamo graf z isto lastno vrednostjo s tem, da ima pripadajoči lastni vektor ničelno komponento. Zato je $\widetilde{H}$ lahko grafted v G vzdolž i_1 in poljubnega vozlišča iz $V(G)$. Takšni konstrukciji pravimo *simetrični graft*. Opazimo, če je H drevo, je tudi $\widetilde{H}$ drevo. Simetrični grafti dreves imajo pomembno vlogo v analizi spektra naključnih grafov [231].

Obe naši konstrukciji sta bili vzdolž enega vozlišča. Obstaja naravna posplošitev za več vozlišč.

Slika 10.12: Konstrukcija $\widetilde{P}_3$, ki ima lastno vrednost $\sqrt{2}$. Uteži so označene na sliki.

10.3.3 Operacije na grafih

Kartezični produkt grafov G_1 in G_2 je graf $G := G_1 \,\square\, G_2$, ki ima za vozlišča $V(G) = V(G_1) \times V(G_2)$, povezave pa so določene s predpisom

$$(i_1, i_2) \sim_G (j_1, j_2) \iff (\{i_1, j_1\} \in E(G_1) \wedge i_2 = j_2) \vee (i_1 = j_1 \wedge \{i_2, j_2\} \in E(G_2)),$$

za $i_1, j_1 \in V(G_1)$ in $i_2, j_2 \in V(G_2)$.

V direktnem produktu $G := G_1 \times G_2$ so vozlišča prav tako $V(G) = V(G_1) \times V(G_2)$. Povezava $(i_1, i_2) \sim_G (j_1, j_2)$ pa obstaja natanko tedaj, ko je $\{i_1, j_1\} \in E(G_1)$ in $\{i_2, j_2\} \in E(G_2)$. Slika 10.13 predstavlja kartezični in direktni produkt P_4 samega s sabo.

Slika 10.13: Kartezični in direktni produkt poti na štirih vozliščih same s sabo.

Za lastne vrednosti kartezičnega in direktnega produkta velja naslednja trditev.

Trditev 10.23

1. $\text{spekter}(G_1 \,\square\, G_2) = \text{spekter}(G_1) + \text{spekter}(G_2)$.
2. $\text{spekter}(G_1 \times G_2) = \text{spekter}(G_1) \cdot \text{spekter}(G_2)$.

10.4 Meje parametrov grafa

V teoriji grafov nas pogosto zanimajo posamezni parametri, ki vsakemu grafu pripišejo neko vrednost. Ker v velikih omrežjih težko določimo njihove natančne vrednosti, si pomagamo z različnimi ocenami. Pogledali si bomo, kako s pomočjo lastnih vrednosti matrike sosednosti in Laplaceove matrike določimo meje nekaterih parametrov v grafu.

Imamo graf $G = (V, E)$. Naj bo n število vozlišč v grafu. Z λ_1 bomo označevali najmanjšo, z λ_2 drugo najmanjšo in z λ_n največjo lastno vrednost matrike sosednosti. Z $\lambda_1(L)$ pa bomo označevali najmanjšo, z $\lambda_2(L)$ drugo najmanjšo in z $\lambda_n(L)$ največjo lastno vrednost Laplaceove matrike.

Povprečna stopnja

Prvi parameter, ki si ga bomo ogledali, je povprečna stopnja vozlišč v grafu. Naj bo G graf na n vozliščih. Stopnja $d(i)$ vozlišča i je število povezav, ki ima krajišče v vozlišču i. Z $\overline{d}$ označimo *povprečno stopnjo* vozlišč v grafu, ki je definirana kot

$$\overline{d} := \frac{1}{n} \sum_{i \in V} d(i).$$

Spodnja lema nam pove, kako je povprečna stopnja povezana z največjo lastno vrednostjo matrike sosednosti.

Lema 10.24 *Naj bo G graf in $\overline{d}$ njegova povprečna stopnja. Potem velja*

$$\overline{d} \leq \lambda_n.$$

Dokaz. Naj bo $y := 1_n$ $n-$dimenzionalen vektor z vsemi koordinatami enakimi 1. Potem je

$$\lambda_n = \max_{\substack{x \in \mathbb{R}^n \\ x \neq 0_n}} \frac{x^T A x}{x^T x} \geq \frac{y^T A y}{y^T y} = \frac{\sum_{i \in V} d(i)}{n} = \overline{d}.$$

□

Zgled 10.25 Na sliki 10.14 imamo graf s 100 vozlišči. Da si stvari poenostavimo, ga poimenujemo kar zvezda in zanj v nadaljevanju uporabljamo oznako Z. Njegove lastne vrednosti matrike sosednosti so enake $\lambda_1 = -9.9212, \lambda_2 = -9.3660, \ldots, \lambda_{100} = 50.4085$. Lastne vrednosti Laplaceove matrike grafa so enake $\lambda_1(L) = 0, \lambda_2(L) = 35.7313, \ldots, \lambda_{100}(L) = 65.1941$. Za povprečno stopnjo tega grafa velja:

$$\overline{d} \leq 50.4085.$$

Diameter in povprečna razdalja

Diameter je največja razdalja med dvema različnima vozliščema v grafu. Lastne vrednosti Laplaceove matrike nam dajo meje za diameter povezanega grafa, glej [214] in [220].

Slika 10.14: Zvezda.

Izrek 10.26 *Za diameter grafa G veljata naslednji neenakosti:*

$$\left\lceil \frac{4}{n\lambda_2(L)} \right\rceil \leq \mathrm{diam}(G) \leq 2 \cdot \left\lfloor \frac{\mathrm{Arcosh}(n-1)}{\mathrm{Arcosh}(\frac{\lambda_n(L)+\lambda_2(L)}{\lambda_n(L)-\lambda_2(L)})} \right\rfloor + 1.$$

Naslednji parameter, ki nas bo zanimal je *povprečna razdalja*, ki jo označimo z $\overline{\rho}$. To je povprečje vseh razdalj med različnimi vozlišči. Spodnji izrek najdemo v [220].

Izrek 10.27 *Naj bo n število vozlišč v grafu, $\lambda_2(L)$ druga najmanjša lastna vrednost Laplaceove matrike ter Δ maksimalna stopnja vozlišča v grafu. Povprečna razdalja je omejena z*

$$\frac{1}{n-1}\left(\frac{2}{\lambda_2(L)} + \frac{n-2}{2}\right) \leq \overline{\rho}(G) \leq \frac{n}{n-1}\left\lceil \frac{\Delta + \lambda_2((L)}{4\lambda_2(L)} \ln(n-1) \right\rceil.$$

Zgled 10.28 Na sliki 10.15 imamo kubični poliedrski graf na 66. vozliščih. Zanj bomo v nadaljevanju uporabljali oznako KPG. Njegove lastne vrednosti matrike sosednosti so enake $\lambda_1 = -2.7448, \lambda_2 = -2.7086, \ldots, \lambda_{66} = 3$. Lastne vrednosti Laplaceove matrike tega grafa so enake $\lambda_1(L) = 0, \lambda_2(L) = 0.213, \ldots, \lambda_{66}(L) = 5.7448$.

Slika 10.15: Kubični poliedrski graf.

Za diameter in povprečno razdaljo kubičnega poliedrskega grafa s slike 10.15 veljajo naslednje ocene:

$$\begin{aligned}
\left\lceil \frac{4}{66 \cdot 0.213} \right\rceil \leq\ & \operatorname{diam}(\mathrm{KPG}) & \leq 2 \left\lfloor \frac{\operatorname{Arcosh}(65)}{\operatorname{Arcosh}(\frac{5.7448+0.213}{5.7448-0.213})} \right\rfloor + 1 \\
1 \leq\ & \operatorname{diam}(\mathrm{KPG}) & \leq 25, \\
\frac{1}{65}\left(\frac{2}{0.213} + \frac{64}{2}\right) \leq\ & \overline{\rho}(\mathrm{KPG}) & \leq \frac{66}{65} \left\lceil \frac{3+0.213}{4 \cdot 0.213} \ln(65) \right\rceil \\
0.636764 \leq\ & \overline{\rho}(\mathrm{KPG}) & \leq 16.2462.
\end{aligned}$$

Povezanost

Vemo že, da je graf povezan natanko tedaj, ko je $\lambda_2(L)$ neničelna. Med $\lambda_2(L)$ in lastnostmi povezanosti torej obstaja tesna povezava, zato $\lambda_2(L)$ imenujemo tudi algebraična povezanost. Vemo, da v primeru, ko je δ minimalna stopnja vozlišča v grafu G, velja, da je $\kappa(G) \leq \eta(G) \leq \delta$. V [216] lahko najdemo ocene, ki nam jih poda naslednji izrek.

Izrek 10.29 *Naj bo G graf z maksimalno stopnjo vozlišča v grafu enako Δ in naj bo $\omega = \frac{\Pi}{n}$. S $\kappa(G)$ označimo minimalno število vozlišč, z $\eta(G)$ pa minimalno število povezav, ki jih moramo odstraniti, da postane G nepovezan. Potem velja:*

1. $\lambda_2(L) \leq \kappa(G) \leq \eta(G)$,

2. $\lambda_2(L) \geq 2\eta(G)(1 - \cos\omega)$,

3. $\lambda_2(L) \geq 2(\cos\omega - \cos 2\omega)\eta(G) - 2\cos\omega(1 - \cos\omega)\Delta(G)$.

Izoperimetrično število

Naj bo $G = (V, E)$ graf in X, Y podmnožici množice vozlišč V. *Izoperimetrično število* je mera, ki nam pove, najmanj koliko povezav moramo v omrežju odstraniti, da ločimo največjo možno podmnožico vozlišč X od preostalega večjega dela Y. Izoperimetrično število nam je v veliko pomoč, če želimo na primer konstruirati dobro povezano omrežje. Pove namreč, ali ima graf ozko grlo, kar pomeni, da lahko najdemo dve veliki podmnožici množice vozlišč, ki sta med seboj povezani z majhnim številom povezav.

Definicija 10.30 Naj bo X neprazna podmnožica množice vozlišč V in Y njen komplement. Z $E(X, Y)$ označimo množico povezav, ki povezujejo X in Y. Izoperimetrično število je definirano kot

$$i(G) := \min\left\{ \frac{|E(X,Y)|}{\min\{|X|, |Y|\}}; \emptyset \neq X \subsetneqq V, Y = V \setminus X \right\}.$$

Če bo parameter $i(G)$ majhno število, imamo ozko grlo. V primeru, ko je $i(G)$ velik, so vse možne delitve množice vozlišč na dva dela med seboj povezane z velikim številom povezav. Če vzamemo $X = \{v\}$, kjer je v vozlišče z najnižjo stopnjo $d(v) = \delta(G)$, vidimo, da je $i(G) \leq \delta(G)$. Če je G nepovezan, dobimo $i(G) = 0$.

Za izoperimetrično število velja neenakost

$$i(G) \geq \frac{\lambda_2(L)}{2}.$$

Ko je $\lambda_2(L) \leq 2$, poda boljšo mejo naslednji izrek.

Izrek 10.31 *Izoperimetrično število je navzdol omejeno z lastnimi vrednostmi Laplaceove matrike z*

$$i(G) \geq \min\left\{1, \frac{\lambda_2(L)\lambda_n(L)}{2(\lambda_n(L) + \lambda_2(L) - 2)}\right\}.$$

Zgornji izrek najdemo v [215]. Poglejmo si še zgornjo mejo za izoperimetrično število, obravnavano v [219].

Izrek 10.32 *Naj bo G graf, ki ni poln graf in ima maksimalno stopnjo Δ. Potem je*

$$\lambda_2(L) \leq \Delta.$$

Dokaz. Če je G nepovezan, potem res velja $\lambda_2(L) = 0 \leq \Delta$. Naj bo sedaj G povezan. Njegovo matriko sosednosti označimo z A. Ker smo predpostavili, da graf ni poln, graf G vsebuje pot na treh vozliščih P_3 kot induciran podgraf. Vemo, da je $\lambda_2(A(P_3)) = 0$. S pomočjo izreka o prepletanju dobimo $0 = \lambda_2(A(P_3)) \leq \lambda_{n-1}(A)$. Po trditvi 10.13 velja $\lambda_2(L) \leq \Delta - \lambda_{n-1}(A) \leq \Delta$.

□

Izrek 10.33 *Naj bo $G = (V, E)$ graf različen od K_1, K_2 ali K_3. Potem je*

$$i(G) \leq \sqrt{\lambda_2(L)(2\Delta - \lambda_2(L))}.$$

Dokaz tega izreka je zanimiv primer, kako lahko dobimo netrivialne meje Laplaceovih lastnih vrednosti in ga lahko najdete v podpoglavju 10.4.1. Sedaj le še izračunajmo izoperimetrično število za dva grafa.

Zgled 10.34 Za graf zvezde s slike 10.14 je $17.8657 \leq i(Z) \leq \sqrt{56.1601(2 \cdot 62 - 56.1601)} = 56.1601$.
Kubični poliedrski graf s slike 10.15 ima $\lambda_2(L) = 0.213 \leq 2$. Zanj velja $\min\{1, 0.154586\} = 1 \leq i(KPG) \leq \sqrt{0.213(2 \cdot 3 - 0.213)} = 1.11024$.

Razširitev

Velikokrat je dobra razširitev vozlišč zaželena lastnost omrežja. Pogosta definicija, ki zajema lastnost, da imajo vse majhne množice vozlišč velike soseščine, je sledeča.

Definicija 10.35 Naj bo $N(S)$ množica sosedov podmnožice vozlišč S brez množice S. Potem je *razširitev vozlišč* enaka

$$c_V := \min\left\{\frac{|N(S)|}{|S|}; S \subseteq V, |S| \leq \frac{n}{2}\right\}.$$

Ker je razširitev vozlišč po definiciji težko izmeriti, si pri njeni oceni pomagamo z naslednjimi mejami, glej [213].

Izrek 10.36 $\frac{\lambda_2(L)}{\frac{\Delta}{2}+\lambda_2(L)} \leq c_V = \mathcal{O}(\sqrt{\lambda_2(L)})$.

Graf z razširitvijo vsaj α imenujemo α - *povečevalec.*

Zgled 10.37 Za zvezdo s slike 10.14 dobimo $c_V \geq \frac{35.7313}{\frac{62}{2}+35.7313} = 0.53545$.

Navedli in dokazali bomo šibkejšo verzijo zgornjega izreka za regularne grafe, glej [224].

Izrek 10.38 *$d-$regularen graf je $\frac{\lambda_2(L)}{2d}-$povečevalec.*

Dokaz. Naj bo $G = (V, E)$ $d-$regularen graf na vozliščih $V = \{1, \ldots, n\}$. Naj bo S podmnožica množice vozlišč V kardinalnosti $s \leq \frac{n}{2}$. Definirajmo vektor $x \in \{s-n, s\}^n$ kot

$$x_i := \begin{cases} s-n, & i \in S \\ s, & i \in V \setminus S. \end{cases}$$

Po definiciji Laplaceove matrike L, je

$$x^T L x = x^T(D-A)x = d\sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{\{i,j\}\in E} x_j = \sum_{\{i,j\}\in E} (x_i - x_j)^2 = n^2|E(S, V\setminus S)|,$$

kjer je $E(S, V \setminus S)$ množica povezav z natanko enim krajiščem v S. Ker je $\sum_i x_i = s(s-n) + (n-s)s = 0$, vidimo, da je x pravokoten na 1_n, lastni vektor samih enic matrike L, ki pripada lastni vrednosti 0. Ker že vemo, da je $\lambda_2(L) = \min_{x\perp 1_n} \frac{x^T L x}{x^T x}$, lahko zaključimo, da je $n^2|E(S, V\setminus S)| = x^T L x \geq \lambda_2(L) x^T x = \lambda_2(L) sn(n-s)$, in posledično $|N(S)| \geq \frac{|E(S,V\setminus S)|}{d} \geq \frac{\lambda_2(L)s(n-s)}{dn} \geq \frac{\lambda_2(L)|S|}{2d}$.

□

Zgled 10.39 Za kubični poliedrski graf na sliki 10.15, ki je $3-$regularen, dobimo $\frac{\lambda_2(L)}{2d} = \frac{0.213}{2\cdot 3} = 0.0355-$povečevalec.

Routing number

Imejmo množico različnih kroglic, ki jih na začetku postavimo na različna vozlišča povezanega grafa $G = (V, E)$, kjer je $|V| = n$. Dana je permutacija π vozlišč V. Naš cilj je, vsako kroglico, ki je na začetku na vozlišču $i \in V$, prestaviti na vozlišče $\pi(i)$. To naredimo tako, da izberemo množico neincidenčnih povezav $E_0 \subseteq E$ in zamenjamo kroglici na krajiščih vsake povezave iz E_0. To ponavljamo, dokler niso vse kroglice na pravih mestih. Minimalno število korakov, ki jih potrebujemo, da dosežemo cilj, označimo z $rt(G, \pi)$. *Routing number* $rt(G)$ grafa G pa je enak

$$rt(G) := \max_{\pi\in S_n} rt(G, \pi).$$

Zgoraj opisani postopek lahko uporabimo na primer pri ocenjevanju učinkovitosti paralelnih arhitektur. Predstavljamo si, da so kroglice podatki, ki morajo biti preneseni med procesorji. Predpostavimo, da je čas potreben za obdelavo podatkov zanemarljiv. Routing number predstavlja zgornjo mejo časa izvršitve vsake stopnje tega paralelnega algoritma. Pove nam torej, koliko je določena struktura primerna za paralelne sisteme. Navedimo le naslednji izrek.

Izrek 10.40 *Naj bo G povezan $d-$regularen graf. Potem je $\lambda_{n-1} < d$ in*

$$rt(G) = \mathcal{O}\left(\frac{d^2}{(d-\lambda_{n-1})^2}\ln^2 n\right).$$

Kromatično število

Barvanje grafa je dodelitev barv vozliščem tako, da sta sosednji vozlišči pobarvani z različnima barvama. O k-barvanju govorimo, kadar lahko graf pobarvamo s k različnimi barvami. Kromatično število grafa je najmanjše število barv, potrebnih za barvanje. Označimo ga s $\chi(G)$. Računanje kromatičnega števila je $\mathcal{NP}-$težak problem, glej [225]. Pri ocenjevanju kromatičnega števila si pomagamo z lastnimi vrednostmi matrike sosednosti, ki nam podajo zgornjo in spodnjo mejo, glej [222].

Izrek 10.41 *Naj bo G graf. Potem je $\chi(G) \leq 1 + \lambda_n$.*

Dokaz. Naj bo H podgraf grafa G. Naj bo H brez izoliranih točk in tak, da je $\chi(H) = \chi(G)$ ter da za vsako povezavo e v H velja $\chi(H-e) < \chi(H)$. Preprosto lahko pokažemo, da tak graf vedno obstaja in da je $\chi(H-e) = \chi(H) - 1$. Naj bo $\delta(H)$ minimalna stopnja vozlišča podgrafa H. Velja, da je $\chi(H) \leq \delta(H)+1$, kar dokažemo s protislovjem. Predpostavimo, da je $\chi(H) > \delta(H)+1$ in naj bo $i \in V(H)$ vozlišče z $d_H(i) = \delta(H)$. Naj bo $j \in N(i)$ in $e = \{i,j\}$. Definiramo k kot $k := \chi(H-e) = \chi(H) - 1 > \delta(H)$. Ker je $d_H(i) = \delta(H)$, imamo za vozlišče i prosto barvo in lahko konstruiramo k-barvanje grafa H iz k-barvanja grafa $H-e$. Tako smo prišli do protislovja. Iz leme 10.24 in izreka o prepletanju dobimo, da je $\chi(G) - 1 = \chi(H) - 1 \leq \delta(H) \leq \lambda_n(H) \leq \lambda_n(G)$.

□

Poglejmo si še naslednja izreka o spodnji meji kromatičnega števila, ki jih najdemo v [217] in [218] ter [208].

Izrek 10.42 *Naj bo G graf. Potem je $1 - \frac{\lambda_n}{\lambda_1} \leq \chi(G)$.*

Izrek 10.43 *Naj bo G graf. Potem je $\frac{n}{n-\lambda_n} \leq \chi(G)$.*

Zgled 10.44 Zgornji meji kromatičnega števila za graf zvezde s slike 10.14 in kubični poliedrski graf s slike 10.15 sta enaki

$$\begin{aligned} \chi(Z) &\leq 1 + 50.4085 = 51.4085 \\ \chi(KPG) &\leq 1 + 3 = 4. \end{aligned}$$

Spodnji meji pa sta enaki

$$\begin{aligned} 1 - \frac{50.4085}{-9.9212} = 6.08089 &\leq \chi(Z) \\ 1 - \frac{3}{-2.7448} = 2.09298 &\leq \chi(KPG) \end{aligned}$$

oziroma

$$\begin{aligned} \frac{100}{100 - 50.4085} = 2.01647 &\leq \chi(Z) \\ \frac{66}{66-3} = 1.04762 &\leq \chi(KPG). \end{aligned}$$

Neodvisno število

Neodvisna množica je množica vozlišč, v kateri nobeni dve vozlišči nista sosednji. Pravimo ji tudi *stabilna množica*. V grafu G je *neodvisno število* $\alpha(G)$ maksimalna kardinalnost med vsemi kardinalnostmi neodvisnih množic grafa G. V [221] lahko najdemo nekaj rezultatov, ki jih dobimo iz ugotovitev Hoffmana in Lovász, objavljenih v [223].

Izrek 10.45 *Naj bo G $d-$regularen graf. Potem velja*

$$\alpha(G) \leq n\left(1 - \frac{d}{\lambda_n(L)}\right).$$

Naj bo sedaj $G = (V, E)$ graf z n vozlišči in stopnjami vozlišč $d_1 \leq d_2 \ldots \leq d_n$. Definirajmo $\overline{d}_s := \frac{1}{s}\sum_{i\in\{1,\ldots,s\}} d_i$ za vsak $s \in \{1, \ldots, n\}$. Potem je zaporedje $\overline{d}_1, \overline{d}_2, \ldots, \overline{d}_n$ nepadajoče in za $d-$regularen graf velja $\overline{d}_s = d$ za vsak $s \in \{1, \ldots, n\}$.

Izrek 10.46 *Naj bo s_0 najmanjše tako celo število, da je $\overline{d}_{s_0} > \frac{\lambda_n(L)(n-s_0)}{n}$. Potem je $\alpha(G) \leq s_0 - 1$ in zato $\alpha(G) \leq n\left(1 - \frac{\overline{d}_{s_0-1}}{\lambda_n(L)}\right)$.*

Dokaz. Pokazali bomo, da vedno, kadar imamo stabilno množico moči $s > 1$ v G, je $\frac{\lambda_n(L)(n-s)}{n} \geq \overline{d}_s$. Naj bo torej $S \subseteq V$ stabilna množica moči $s := |S| > 1$. Enakost velja, ko je $s = n$, saj je potem $\overline{d}_i = 0$ za vsak $i \in \{1, \ldots, n\}$. Naj bo sedaj $s < n$. Definirajmo $x \in \mathbb{R}^n$ kot

$$x_i := \begin{cases} 0, & \text{če } i \in S \\ 1, & \text{sicer.} \end{cases}$$

Vektor x ni konstanten (ni večkratnik vektorja samih enic 1_n). Po trditvi 10.15 dobimo

$$\lambda_n(L) \geq n\frac{\sum_{\{i,j\}\in E}(x_i - x_j)^2}{\sum_{\{i,j\}\in\binom{V}{2}}(x_i - x_j)^2} = n\frac{|E(S, V \setminus S)|}{s(n-s)}.$$

Ker med vozlišči v množici S ni povezav, imamo $|E(S, V \setminus S)| \geq s\overline{d}_s$ in zato je $\lambda_n(L) \geq n\frac{s\overline{d}_s}{s(n-s)} = n\frac{\overline{d}_s}{n-s}$, torej je res $\frac{\lambda_n(L)(n-s)}{n} \geq \overline{d}_s$. Naj bo sedaj s_0 najmanjše celo število, za katero to ne velja. Potem ne obstaja stabilna množica kardinalnosti s_0. Ker je $(\overline{d}_s)_{s=1,\ldots,n}$ nepadajoče zaporedje, za vsak $s \geq s_0$ neenakost prav tako ne velja in zato ne more obstajati stabilna množica kardinalnosti večje od $s_0 - 1$.

□

Širina bisekcije

Kadar imamo podan graf na sodo mnogo vozliščih, pomislimo na razdelitev vozlišč v dva razreda enakih velikosti. *Širina bisekcije* grafa je minimalno število povezav med dvema enako velikima deloma grafa. Označimo jo z bw. Poglejmo si naslednjo mejo širine bisekcije.

Lema 10.47 *Naj bo $G = (V, E)$ graf na vozliščih $\{1, \ldots, n\}$, kjer je n pozitivno sodo število. Potem je*

$$bw(G) \geq \frac{n}{4}\lambda_2(L).$$

Dokaz. Naj bo S poljubna podmnožica V kardinalnosti $\frac{n}{2}$ in definirajmo

$$x_i := \begin{cases} 1, & \text{če } i \in S \\ -1, & \text{če } i \notin S \end{cases}$$

za vsak $i \in V$. Potem je $\sum_{i \in V} x_i = 0$. $x \perp 1_n$ in po posledici 10.14 in lemi 10.7 je $n\lambda_2(L) = x^T x \lambda_2(L) \leq x^T L x = \sum_{\{i,j\} \in E} (x_i - x_j)^2 = \sum_{\{i,j\} \in E(S, V \setminus S)} (x_i - x_j)^2 = 4 \cdot |E(S, V \setminus S)|$. Če za S izberemo razred z minimalno širino bisekcije, je s tem lema dokazana.

□

Spodnja meja širine bisekcije je dosežena natanko tedaj, ko so vsa vozlišča incidenčna $\frac{\lambda_2(L)}{2}$ prereznim povezavam, kar je res za na primer polne grafe, polne dvodelne grafe, hiperkocke in Petersenov graf. Za $\sqrt{n} \times \sqrt{n}$ kartezični produkt dveh poti pa je širina bisekcije enaka $\sqrt{n}$, medtem ko je $\lambda_2(L) = 2 - 2cos(\frac{\pi}{\sqrt{n}}) \approx \frac{\pi^2}{n}$ in zato $\frac{n}{4}\lambda_2(L) \approx \frac{\pi^2}{4}$. Tako je lahko razlika med optimalno širino bisekcije in spodnjo mejo zelo velika.

Širina bisekcije je tesno povezana z izoperimetričnim številom. Neposredno iz definicije obeh, $i(G)$ in $bw(G)$, dobimo, da je $i(G) \leq \frac{2bw(G)}{n}$. Spodnja meja za izoperimetrično število tako podaja tudi spodnjo mejo za širino bisekcije.

10.4.1 Dokaz izreka 10.33

V tem podpoglavju si bomo pogledali dokaz izreka 10.33. Spomnimo se, kaj nam izrek 10.33 pove.

Naj bo $G = (V, E)$ graf različen od K_1, K_2 ali K_3. Potem je $i(G) \leq \sqrt{\lambda_2(L)(2\Delta - \lambda_2(L))}$.

Dokaz. Naj bo $G = (V, E)$ graf na n vozliščih in z m povezavami, ki ni enak K_1, K_2 ali K_3. Naj bo Δ maksimalna stopnja vozlišča v grafu G in $\lambda_2(L)$ druga najmanjša lastna vrednost Laplaceove matrike, ki jo bomo kasneje označevali kar z λ. Dokazati želimo

$$i(G) \leq \sqrt{\lambda_2(L)(2\Delta - \lambda_2(L))}.$$

Če je $\lambda = 0$, je graf nepovezan, $i(G) = 0$ in smo končali. V primeru, ko je G poln graf na štirih ali večih vozliščih, preprosto uporabimo $\lambda = n$. Tako lahko predpostavimo, da G ni poln graf. V tem primeru, po izreku 10.32, velja $\lambda \leq \Delta$. Sedaj ločimo dva primera.

Če je $\delta < \lambda$, za izraz pod korenom velja

$$\lambda(2\Delta - \lambda) > \delta(2\Delta - \lambda) \geq \delta(2\Delta - \Delta) = \delta\Delta \geq \delta^2 \geq i(G)^2$$

in s tem je izrek dokazan.

Poglejmo še primer, kjer predpostavimo, da je $\lambda \leq \delta$. Naj bo $y \in \mathbb{R}^n$ lastni vektor za λ in definirajmo množico W kot $W := \{i \in V; y_i > 0\}$. Ker bi lahko y zamenjali z $-y$, vzamemo $|W| = \min\{|W|, |V \setminus W|\}$. Definiramo

$$g_i := \begin{cases} y_i, & \text{če } i \in W \\ 0, & \text{sicer.} \end{cases}$$

Z $E(W) \subseteq E$ označimo povezave med vozlišči v množici W. Če enačbo $Ly = \lambda y$ z leve pomnožimo z vektorjem y^T in upoštevamo, da je $L = (D - A)$, dobimo $y^T(D - A)y = \lambda y^T y$.

Leva stran te enačbe je enaka

$$\begin{aligned}
&\sum_{i\in W}\left(d(i)y_i \pm \sum_{j:\{i,j\}\in E} y_j\right) y_i = \sum_{i\in W}\sum_{j:\{i,j\}\in E}(y_i \pm y_j)y_i = \\
&= \sum_{\{i,j\}\in E(W)}((y_i \pm y_j)y_i + (y_j \pm y_i)y_j) + \sum_{\{i,j\}\in E(W,V\setminus W)}(y_i \pm y_j)y_i = \\
&= \sum_{\{i,j\}\in E(W)}(y_i \pm y_j)^2 + \sum_{\{i,j\}\in E(W,V\setminus W)}(y_i \pm y_j)y_i = \qquad (10.3)\\
&= \sum_{\{i,j\}\in E(W)}(y_i \pm y_j)^2 + \sum_{i\in W} d(i)y_i^2 \pm \sum_{\{i,j\}\in E(W,V\setminus W)} y_j y_i = \\
&= \sum_{\{i,j\}\in E}(g_i \pm g_j)^2 - \sum_{i\in W} d(i)g_i^2 + \sum_{i\in W} d(i)y_i^2 \pm \sum_{\{i,j\}\in E(W,V\setminus W)} y_j y_i = \\
&= \sum_{\{i,j\}\in E}(g_i \pm g_j)^2 \pm \sum_{\{i,j\}\in E(W,V\setminus W)} y_j y_i.
\end{aligned}$$

Seštevanje po vseh povezavah $\{i,j\} \in E$ ne bi smelo biti odvisno od tega, na katerem krajišču povezave leži i in na katerem j. Slednje v našem primeru zmeraj drži, saj je $(g_i \pm g_j)^2 = (g_j \pm g_i)^2$ za vsak $i, j \in V$. Iz lastnosti $Ly = \lambda y$ dobimo za vsak $i \in V$ enakost

$$\lambda y_i = d(i)y_i - \sum_{j:\{i,j\}\in E} y_j.$$

Ko uporabimo to enakost in v izrazu 10.3 vzamemo negativni predznak dobimo, da je

$$\begin{aligned}
\lambda y^T y &= \lambda \sum_{i\in W} y_i^2 = \\
&= \sum_{i\in W}\left(d(i)y_i - \sum_{j:\{i,j\}\in E} y_j\right) y_i = \\
&= \sum_{\{i,j\}\in E}(g_i - g_j)^2 - \sum_{\{i,j\}\in E(W,V\setminus W)} y_j y_i. \qquad (10.4)
\end{aligned}$$

Če v izrazu 10.3 vzamemo pozitivni predznak, dobimo

$$\begin{aligned}
(2\Delta - \lambda)\sum_{i\in W} y_i^2 &= 2\Delta \sum_{i\in W} y_i^2 - \sum_{i\in W} d(i)y_i^2 + \sum_{i\in W}\sum_{j:\{i,j\}\in E} y_j y_i \geq \\
&\geq \sum_{i\in W} d(i)y_i^2 + \sum_{i\in W}\sum_{\{i,j\}\in E} y_j y_i = \\
&= \sum_{\{i,j\}\in E}(g_i + g_j)^2 + \sum_{\{i,j\}\in E(W,V\setminus W)} y_j y_i.
\end{aligned}$$

Naj bo $\alpha := \sum_{\{i,j\}\in E(W,V\setminus W)} y_i y_j$. Zgornji izraz lahko pomnožimo z enačbo 10.4, saj sta

leva in desna stran nenegativni. Dobimo

$$\lambda(2\Delta-\lambda)\left(\sum_{i\in W} y_i^2\right)^2 \geq$$

$$\geq \sum_{\{i,j\}\in E}(g_i+g_j)^2 \sum_{\{i,j\}\in E}(g_i-g_j)^2 + \alpha\left(\sum_{\{i,j\}\in E}(g_i-g_j)^2 - \sum_{\{i,j\}\in E}(g_i+g_j)^2\right) - \alpha^2 =$$

$$= \sum_{\{i,j\}\in E}(g_i+g_j)^2 \sum_{\{i,j\}\in E}(g_i-g_j)^2 - \alpha\left(4\sum_{\{i,j\}\in E(W)} y_iy_j + \alpha\right).$$

V tem izrazu bi se radi znebili α. Opazimo, da iz definicije W sledi, da je $\alpha \geq 0$. Če upoštevamo, da je $(L-\lambda I)y = 0$, definicijo W in dejstvo, da obravnavamo primer, ko je $\lambda \leq \delta$, dobimo

$$\begin{aligned}
4\sum_{\{i,j\}\in E(W)} y_iy_j + \alpha &= 2\sum_{\{i,j\}\in E(W)} y_iy_j + 2\sum_{\{i,j\}\in E(W)} y_iy_j + \sum_{\{i,j\}\in E(W,V\setminus W)} y_iy_j = \\
&= 2\sum_{\{i,j\}\in E(W)} y_iy_j + \sum_{i\in W} y_i \sum_{j:\{i,j\}\in E} y_j = \\
&= 2\sum_{\{i,j\}\in E(W)} \underbrace{y_iy_j}_{\geq 0} + \sum_{i\in W} \underbrace{(d(i)-\lambda)}_{\geq 0} y_i^2 \geq \\
&\geq 0.
\end{aligned}$$

Od tod sledi, da je

$$\lambda(2\Delta-\lambda)\left(\sum_{i\in W} y_i^2\right)^2 \geq \sum_{\{i,j\}\in E}(g_i+g_j)^2 \sum_{\{i,j\}\in E}(g_i-g_j)^2.$$

Sedaj definirajmo vektorja $v, w \in \mathbb{R}^m$ z $v_{\{i,j\}} := g_i + g_j$ in $w_{\{i,j\}} := |g_i - g_j|$. Ko uporabimo Cauchy-Schwartzovo neenakost, dobimo

$$\begin{aligned}
\left(\sum_{\{i,j\}\in E}|g_i^2-g_j^2|\right)^2 &= \left(\sum_{\{i,j\}\in E}(g_i+g_j)|g_i-g_j|\right)^2 = \\
&= \langle v,w\rangle^2 \leq \\
&\leq \|v\|^2\|w\|^2 = \qquad (10.5)\\
&= \sum_{\{i,j\}\in E}(g_i+g_j)^2 \sum_{\{i,j\}\in E}(g_i-g_j)^2 \leq \\
&\leq \lambda(2\Delta-\lambda)\left(\sum_{i\in W} y_i^2\right)^2.
\end{aligned}$$

Poiščimo še spodnjo mejo izraza $\sum_{\{i,j\}\in E}|g_i^2 - g_j^2|$. Naj bodo $0 = t_0 < t_1 < \ldots < t_N$ različne vrednosti komponent g. Definiramo $V_k := \{i \in V; g_i \geq t_k\}$ za $k \in \{0,\ldots,N\}$. Naj bo $V_{N+1} := \emptyset$. Za $k \in \{0,\ldots,N+1\}$ velja $V_k \subseteq W$ in zato je $|V_k| \leq |W|$ ter $|V_k| = \min\{|V_k|, |V\setminus V_k|\}$. Pravtako je $V_N \subseteq V_{N-1} \subseteq \ldots \subseteq V_1 = W \subseteq V_0 = V$. $|V_k| - |V_{k+1}|$

je število komponent g, ki so enake t_k za vsak $k \in \{0, \ldots, N\}$. Vsoto $\sum_{\{i,j\}\in E} |g_i^2 - g_j^2|$ lahko na uporaben način zapišemo kot

$$\begin{aligned}
\sum_{\{i,j\}\in E} |g_i^2 - g_j^2| &= \sum_{k=1}^{N} \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_k}} (g_j^2 - g_i^2) = \\
&\underbrace{=}_{\text{glej spodaj}} \sum_{k=1}^{N} \sum_{\{i,j\}\in E(V_k, V\setminus V_k)} (t_k^2 - t_{k-1}^2) = \\
&= \sum_{k=1}^{N} |E(V_k, V \setminus V_k)|(t_k^2 - t_{k-1}^2) \geq \\
&\geq i(G) \sum_{k=1}^{N} |V_k|(t_k^2 - t_{k-1}^2) = i(G) \sum_{k=1}^{N} t_k^2(|V_k| - |V_{k+1}|).
\end{aligned}$$

Od tod sledi, da je $\sum_{\{i,j\}\in E} |g_i^2 - g_j^2| \geq i(G) \sum_{i\in V} g_i^2 = i(G) \sum_{i\in W} y_i^2$, kar skupaj s 10.5 dokaže izrek.

Preostane nam še dokaz naslednje enakosti

$$\sum_{k=1}^{N} \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_k}} (g_j^2 - g_i^2) = \sum_{k=1}^{N} \sum_{\{i,j\}\in E(V_k, V\setminus V_k)} (t_k^2 - t_{k-1}^2), \tag{10.6}$$

kar dokažemo z indukcijo na N. Primer, ko je $N = 1$, je očiten. Naj bo $N > 1$. Predpostavimo, da enačba že velja za vse take izraze, kjer prva vsota teče do $\leq N-1$; to je, ko imamo vektor $\tilde{g}$ grafa $\tilde{G} = (\tilde{V}, \tilde{E})$, ki ima N različnih vrednosti komponent $0 = \tilde{t}_0 < \tilde{t}_1 < \ldots < \tilde{t}_{N-1}$, in kjer so definirane podmnožice $\tilde{V}_{N-1} \subseteq \tilde{V}_{N-2} \subseteq \ldots \subseteq \tilde{V}_1 = \tilde{W} \subseteq \tilde{V}_0 = \tilde{V}$.

Zgornje bomo uporabili v naslednjem primeru. Definiramo $\tilde{G} := G - V_N$. Naj bo $\tilde{g}$ vektor, ki ga dobimo iz g, na množici vozlišč $\tilde{V}$. Vrednost $\tilde{t}_k = t_k$ za vsak $k \in \{0, \ldots, N-1\}$. Potemtakem definiramo množice $\tilde{V}_k$ kot $\tilde{V}_k = V_k \setminus V_N$ za vsak $k \in \{0, \ldots, N-1\}$. Za vsak $k \in \{0, \ldots, N-1\}$ velja $V_N \subseteq V_k$, tako da se množice $\tilde{V}_k$ razlikujejo od množic V_k natanko za vozlišča iz množice V_N. Dobimo

$$\begin{aligned}
\sum_{k=1}^{N} \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_k}} (g_j^2 - g_i^2) &= \sum_{k=1}^{N-1} \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_k}} (g_j^2 - g_i^2) + \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_N}} (g_j^2 - g_i^2) = \\
&= \sum_{k=1}^{N-1} \sum_{\substack{\{i,j\}\in E \\ \tilde{g}_i<\tilde{g}_j=t_k}} (\tilde{g}_j^2 - \tilde{g}_i^2) + \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_N}} (g_j^2 - g_i^2) = \qquad (10.7) \\
&= \sum_{k=1}^{N-1} \sum_{\{i,j\}\in E_{\tilde{G}}(\tilde{V}_k, \tilde{V}\setminus\tilde{V}_k)} (\tilde{t}_k^2 - \tilde{t}_{k-1}^2) + \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_N}} (g_j^2 - g_i^2) = \\
&= \underbrace{\sum_{k=1}^{N-1} \sum_{\{i,j\}\in E_{\tilde{G}}(\tilde{V}_k, \tilde{V}\setminus\tilde{V}_k)} (t_k^2 - t_{k-1}^2)}_{(*)} + \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_N}} (g_j^2 - g_i^2).
\end{aligned}$$

Opazimo, da so prerezne povezave $E_{\tilde{G}}(\tilde{V}_k, \tilde{V} \setminus \tilde{V}_k)$ sestavljene le iz povezav iz $\tilde{E}$. Če bi računali kar z grafom G, bi morali kasneje odstraniti nekaj povezav. Odstraniti bi morali povezave, ki imajo eno krajišče v V_N. Na ta načim bi za $(*)$ dobili

$$\begin{aligned}
&\sum_{k=1}^{N-1} \sum_{\{i,j\} \in E_{\tilde{G}}(\tilde{V}_k, \tilde{V} \setminus \tilde{V}_k)} (t_k^2 - t_{k-1}^2) = \\
&= \sum_{k=1}^{N-1} \sum_{\{i,j\} \in E(V_k, V \setminus V_k)} (t_k^2 - t_{k-1}^2) - \underbrace{\sum_{k=1}^{N-1} \sum_{\substack{\{i,j\} \in E(V_k, V \setminus V_k) \\ j \in V_N}} (t_k^2 - t_{k-1}^2)}_{(+)}.
\end{aligned}$$

Podrobneje si oglejmo izraz $(+)$. Za vsak $i \in V$ naj bo $k(i)$ najmanjši indeks, za katerega velja $i \in V \setminus V_{k(i)}$. Potem je $g_i = t_{k(i)-1}$ za vsak $i \in V$ in velja

$$\begin{aligned}
\sum_{k=1}^{N-1} \sum_{\substack{\{i,j\} \in E(V_k, V \setminus V_k) \\ j \in V_N}} (t_k^2 - t_{k-1}^2) &= \sum_{\substack{\{i,j\} \in E(V_N, V \setminus V_N) \\ j \in V_N}} \sum_{\substack{k \in \{0,\ldots,N-1\} \\ i \in V \setminus V_k}} (t_k^2 - t_{k-1}^2) = \\
&= \sum_{\substack{\{i,j\} \in E(V_N, V \setminus V_N) \\ j \in V_N}} \sum_{k=k(i)}^{N-1} (t_k^2 - t_{k-1}^2) = \\
&= \sum_{\substack{\{i,j\} \in E(V_N, V \setminus V_N) \\ j \in V_N}} (t_{N-1}^2 - t_{k(i)-1}^2) = \\
&= \sum_{\substack{\{i,j\} \in E(V_N, V \setminus V_N) \\ j \in V_N}} (t_{N-1}^2 - g_i^2).
\end{aligned}$$

Sedaj lahko nadaljujemo s 10.7. Upoštevajmo, da je

$$\{(i,j); \{i,j\} \in E(V_N, V \setminus V_N), j \in V_N\} = \{(i,j); \{i,j\} \in E, g_i < g_j = t_N\}.$$

Ko vse združimo, vidimo, da velja

$$
\begin{aligned}
&\sum_{k=1}^{N} \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_k}} (g_j^2 - g_i^2) = \\
&= \underbrace{\sum_{k=1}^{N-1} \sum_{\{i,j\}\in E_{\tilde{G}}(\tilde{V}_k,\tilde{V}\setminus\tilde{V}_k)} (t_k^2 - t_{k-1}^2)}_{(*)} + \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_N}} (g_j^2 - g_i^2) = \\
&= \sum_{k=1}^{N-1} \sum_{\{i,j\}\in E(V_k,V\setminus V_k)} (t_k^2 - t_{k-1}^2) - \underbrace{\sum_{k=1}^{N-1} \sum_{\substack{\{i,j\}\in E(V_k,V\setminus V_k) \\ i\in V_N}} (t_k^2 - t_{k-1}^2)}_{(+)} + \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_N}} (g_j^2 - g_i^2) = \\
&= \sum_{k=1}^{N-1} \sum_{\{i,j\}\in E(V_k,V\setminus V_k)} (t_k^2 - t_{k-1}^2) - \sum_{\substack{\{i,j\}\in E(V_N,V\setminus V_N) \\ j\in V_N}} (t_{N-1}^2 - g_i^2) + \sum_{\substack{\{i,j\}\in E \\ g_i<g_j=t_N}} (\underbrace{g_j^2}_{=t_N^2} - g_i^2) = \\
&= \sum_{k=1}^{N-1} \sum_{\{i,j\}\in E(V_k,V\setminus V_k)} (t_k^2 - t_{k-1}^2) + \sum_{\substack{\{i,j\}\in E(V_N,V\setminus V_N) \\ j\in V_N}} (t_N^2 - g_i^2 - t_{N-1}^2 + g_i^2) = \\
&= \sum_{k=1}^{N-1} \sum_{\{i,j\}\in E(V_k,V\setminus V_k)} (t_k^2 - t_{k-1}^2) + \sum_{\substack{\{i,j\}\in E(V_N,V\setminus V_N) \\ j\in V_N}} (t_N^2 - t_{N-1}^2) = \\
&= \sum_{k=1}^{N} \sum_{\{i,j\}\in E(V_k,V\setminus V_k)} (t_k^2 - t_{k-1}^2).
\end{aligned}
$$

S tem smo pokazali, da enakost 10.6 res velja in dokazali naš izrek.

□

Poglavje 11

Robustnost in odpornost

Tina Volčanšek, Tanja Grill, Petra Janžekovič

11.1 Uvod

Kompleksno omrežje je robustno, če obdrži svojo osnovno funkcionalnost tudi po odpovedi nekaterih njenih delov. Študija robustnosti v omrežjih je pomembna, saj temeljito poznavanje obnašanja nekaterih vrst omrežij lahko pomaga zaščititi pred napadi in napakami, ki podrejo sistem, na primer, komunikacijska omrežja kot so internet pred napadi ali za izkoriščanje slabosti omrežij.
Pogosto ločimo med naključnimi napakami in namernimi napadi na omrežja. Primeri za naključne in namerne odpovedi sestavnih delov v realnem svetu kompleksnih omrežij so, na primer, okoljski stres na omrežja, okvare usmerjevalnika na internetu ali namerni napadi na letalska omrežja. Spoznali bomo, da so nekatera omrežja kot na primer internet zelo robustna, ko gre za naključne izpade usmerjevalnikov, lahko pa močno trpijo zaradi ciljno usmerjenih napadov na dobro izbrane osrednje usmerjevalnike.
To poglavje se osredotoča na merila, ki merijo robustnost omrežja oziroma odpornost omrežja na ponavljajoče se napake.

Podali bomo pregled na različna merila in razpravljali o njihovi aplikaciji v praksi glede na njihovo uporabnost in računske zahtevnosti. Pogosto se raziskave o robustnosti osredotočajo na to, kako se ta merila spremenijo, z analizo in meritvijo učinkov, če omrežje podleže zaporedju sestavnih napak in odpove. Kadarkoli lahko, poskusimo povezati različna merila in razpravljati o njihovih prednostih in slabostih. V mnogih primerih uporabimo primere za ilustracijo definicij.

Struktura našega poglavja: razlikujemo med najslabšim primerom merila-povezanost in razdalja, povprečnim in verjetnostnim merilom. Razdelka 1 in 2 govorita o najslabši možni povezanosti in meritvah razdalj. Povprečno merilo za robustnost (razdelek 3) dovoljuje bolj globalen pogled na lastnosti robustnosti, medtem ko verjetnostno merilo (razdelek 4) upošteva okvare verjetnosti implicitno. Medtem, grobo rečeno, merila postanejo bolj in bolj smiselna, bolj kot se bližamo koncu tega poglavja, so pa tudi težja za izračunati.

11.2 Najslabši primer merila: povezanost

Ta del se ukvarja z merili, ki odgovarjajo na vprašanja v obliki "Katero je najmanjše število povezav ali vozlišč, ki jih je potrebno črtati iz omrežja tako, da je končno omrežje odklopljeno in ima lastnost P?"Ti so najslabši primeri meril, saj izbris poljubnega niza vozlišč ali povezav enake velikosti morda ne povzroči enakega učinka. Tako domnevamo, da odstranitev vozlišč ali povezav ni naključna, temveč natančno usmerjena za dosego največjega učinka.

11.2.1 Klasična povezanost

Klasična povezanost je osnova mnogih meril robustnosti. Omrežje imenujemo povezano, če obstaja pot med vsakim parom vozlišč v omrežju. V številnih aplikacijah je povezanost potreben pogoj za omrežje, da izpolni svoj namen. Torej, eno merilo robustnosti omrežja je število vozlišč ali povezav, ki jih je treba odstraniti, da postane omrežje nepovezano. Opazujemo točkovno povezanost in povezanost po povezavah v omrežju.
Povezanost nam torej služi kot merilo robustnosti omrežja.

Če omrežje izgubi svojo funkcionalnost takoj, ko ni več povezano, potem je povezanost res dobro merilo za robustnost omrežja. Če pa imamo primer, kjer z odklopom majhnega nabora vozlišč iz omrežja njegova uporabnost ni resno prizadeta, potem povezanost ni smiselno merilo. Poglejmo za primer internet. Namizni računalnik je na internet priključen preko ene povezave do ponudnika ali strežnika. Prekinitev te povezave izklopi internet, vendar ima le zanemarljiv vpliv na funkcionalnost celotnega interneta. Vendar povezanost omrežja po povezavah je enaka ena. Podobno, odklop majhnega usmerjevalnika bo odklopilo le peščico ljudi iz interneta, ampak dokazuje, da ima internet točkovno povezanost ena.

11.2.2 Kohezivnost

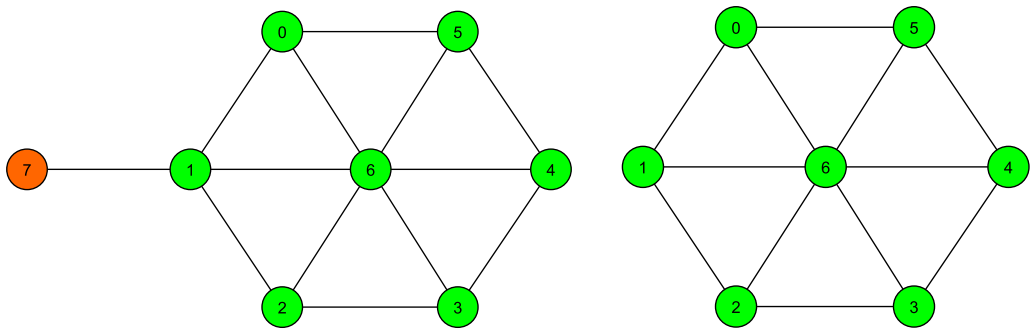
Kohezivnost je prvi predstavil Akiyama [183]. Definira za vsako vozlišče omrežja v kolikšni meri prispeva k povezanosti.

Definicija 11.1 Naj bo $\kappa(G)$ točkovna povezanost omrežja G. Omrežje $G - v$ dobimo tako, da iz omrežja G ostranimo vozlišče v. Za vsako vozlišče v v G je kohezivnost definirana kot:

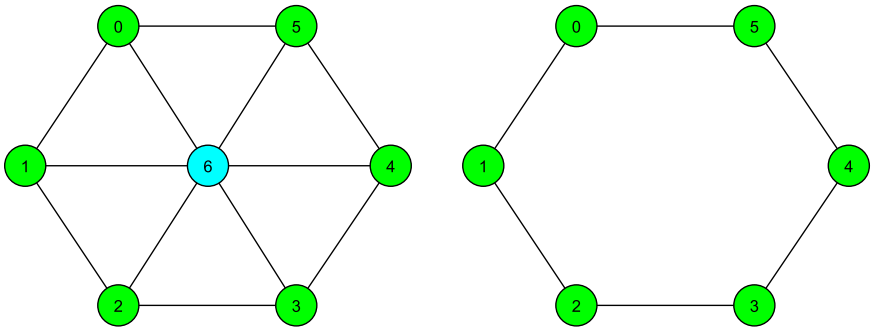
$$c(v) = \kappa(G) - \kappa(G - v).$$

Problem 11.2 *Pogljemo si najprej primer na sliki 1. Vozlišče 7 v tem omrežju ima kohezivnost -2, ker ima omrežje točkovno povezanost 1, če je vozlišče 7 v omrežju in točkovna povezanost 3, če to vozlišče odstranimo iz omrežja.*
Poglejmo še primer na sliki 2. Vozlišče 6 ima kohezivnost 1, ker če odstranimo to vozlišče iz omrežja se točkovna povezanost zmanjša iz 3 na 2.

Iz definicije sledi, da kohezivnost vozlišča ne more biti večja od 1. Intuitivno, vozlišče z negativno kohezivnostjo je vozlišče na robu omrežja medtem ko je vozlišče s kohezivnostjo 1 na sredini oziroma v samem centru omrežja. Da se pokazati, da ima lahko omrežje največ eno vozlišče z negativno kohezivnostjo in da soseska tega negativnega vozlišča vsebuje množico vozlišč velikosti $\kappa(G)$, katere odstranitev izklopi omrežje.



Slika 11.1: primer negativne kohezivnosti



Slika 11.2: primer pozitivne kohezivnosti

Poglejmo še enkrat sliko 1, kjer je vozlišče 7 edino vozlišče z negativno kohezivnostjo. Edini sosed tega vozlišča je vozlišče 1 in to je tudi edino vozlišče, ki razdeli omrežje na 2 dela, če ga izbrišemo.

Kohezivnost vozlišč se lahko izračuna z uporabo standardnih algoritmov za povezanost. Za izračun kohezivnosti vsakega vozlišča pa mora biti algoritem zagnan n-krat, ker je v omrežju n število vozlišč.

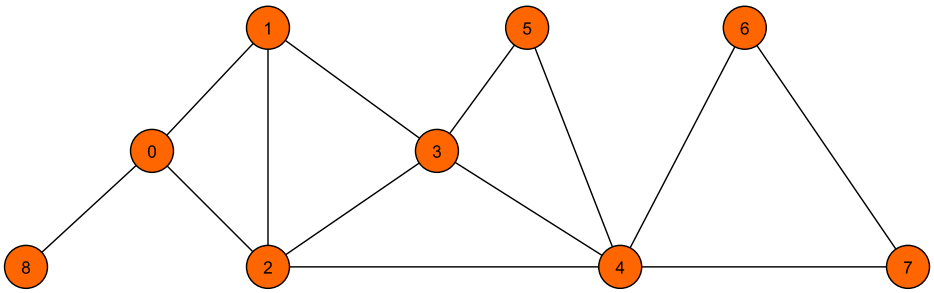
11.2.3 Minimalna m-stopnja

Merilo, ki smo ga do sedaj omenjali, izraža stališča glede povezovanja omrežja. M-stopnjo sta uvedla Boesch in Thomas [185] in se ukvarja s stanjem omrežja po izklopu omrežja.

Definicija 11.3 Minimalna m-stopnja $\xi(m)$ omrežja je najmanjše število povezav, ki jih moramo odstraniti iz omrežja, da omrežje razpade na dva dela G_1 in G_2, kjer G_1 vsebuje natanko m-vozlišč.

Naj bo $G = (V, E)$ omrežje z $V = n$. Za minimalno m-stopnjo veljajo naslednje lastnosti:
- $\xi(m) = \xi(n - m)$,
- $\xi(m) \geq m(\delta(G) - m + 1)$, kjer je $\delta(G)$ najmanjša stopnja kateregakoli vozlišča v G,
- Naj bo G regularno omrežje s stopnjo $r \leq \frac{n}{2}$,$n > 2$ in $m \geq l$. Potem je $r \geq \lfloor \frac{\xi(m)}{m} \rfloor + \lceil \frac{\xi(l)}{l} \rceil$.

Problem 11.4 *Poglejmo si primer minimalne $m-stopnje$ omrežja na sliki 3.*



Slika 11.3: primer omrežja za minimalno m-stopnjo

1-stopnja	2-stopnja	3-stopnja	4-stopnja	5-stopnja	6-stopnja	7-stopnja	8-stopnja
1	2	3	3	3	3	2	1

Tabela 11.1: Tabela prikazuje minimalno $m-stopnjo$ za omrežje na sliki 3

Ni znanega hitrejšega algoritma za izračun minimalne m-stopnje kot je preverjanje vseh sklopov vozlišč velikosti m in preverjanje, če so omrežja, ki jih povzroči ta niz in njegov komplement, povezana. Če je temu tako, preštejemo število povezav, ki povezujejo vozlišča v tem nizu z vozlišči zunaj tega niza. Minimum po vseh sklopih je m-stopnja. To se zgodi v času delovanja $O(\binom{n}{m}|E|)$.
Glavi problem tega merila je, da mora biti končen rezultat delitve omrežja G razpad omrežja na dva dela, G_1 in G_2. To ne izraža intuitiven koncept robustnosti.

Problem 11.5 *Omrežje na sliki 4 ima 3-stopnjo 3 torej moramo odstraniti najmanj 3 povezave, da omrežje razpade na dva dela, medtem ko je izbris dveh debelih povezav dovolj, da odstrani del s tremi vozlišči iz omrežja in tako dobimo 3 dele na katere je omrežje razpadlo.*

11.2.4 Žilavost (Toughness)

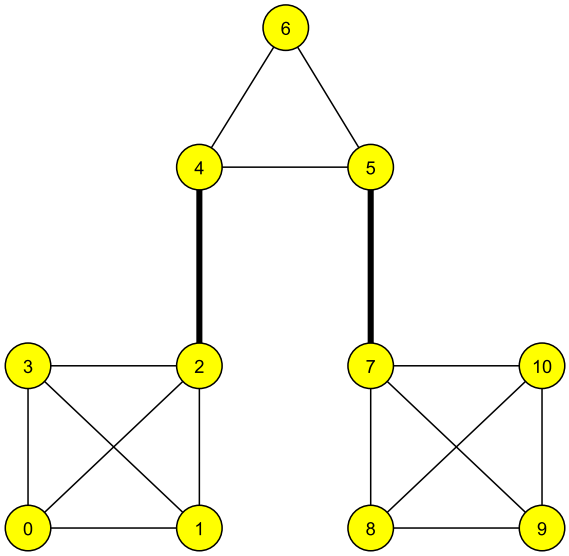
Žilavost omrežja je predstavil Chvátal [186]. Meri število notranje povezanih delov na katere se lahko omrežje razdeli po odstranitvi določenega števila vozlišč.

Definicija 11.6 Naj bo S podmnožica vozlišč G in naj bo $K(G-S)$ število notranje povezanih delov, na katere je G razdeljen po odstranitvi S. Žilavost omrežja G je definirana kot :

$$t(G) = \min_{S\subseteq V, K(G-S)>1} \{\frac{|S|}{K(G-S)}\}.$$

Žilavost povezav omrežja je opredeljena analogno.

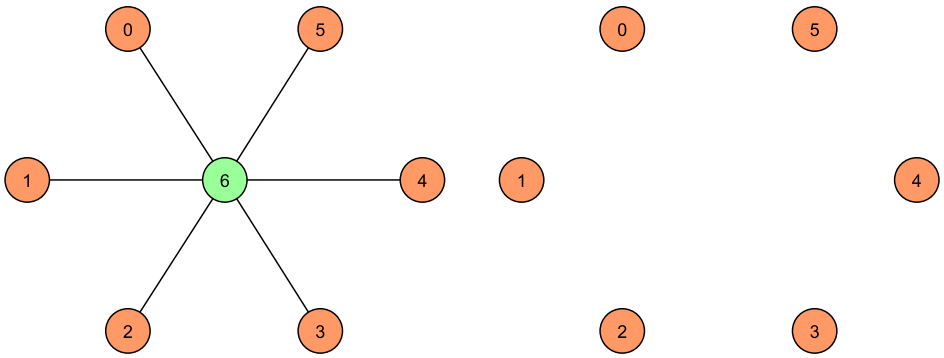
Žilavost omrežja je visoka, če odstranitev velikega števila vozlišč razdeli omrežje na le nekaj delov. Nasprotno pa, če lahko omrežje razdelimo na veliko delov z odstranitvijo majhnega števila vozlišč, potem je žilavost majhna. Omrežje z najmanjšo žilavostjo je



Slika 11.4: protiprimer

zvezda.

Problem 11.7 *Omrežje z najmanjšo žilavostjo je zvezda. Če odstranimo osrednjo vozlišče omrežja razdelimo omrežje na dele velikosti 1 in tako je žilavost zvezde z n vozlišči enaka* $\frac{1}{n-1}$. *Upoštevati moramo, da je osrednje vozlišče tudi edino vozlišče, katerega odstranitev razdeli omrežje.*



Slika 11.5: primer zvezda

Če je omrežje drevo, je žilavost enaka $\frac{1}{\Delta G}$ kjer je ΔG največja stopnja kateregakoli vozlišča [184]. Žilavost celotnega dvodelnega omrežja $K_{m,n}$ z $m \leq n$ in $n \geq 2$ je $\frac{m}{n}$.
Žilavost kroga je ena in iz tega sledi, da je žilavost Hamiltonovega grafa vsaj ena. Chvátal je pokazal tudi povezavo med neodvisnim številom omrežja in žilavostjo. Neodvisno število β_0 je velikost največje podmnožice S vozlišč z lastnostjo, da ni povezave v omrežju, ki

povezuje dve vozlišči v S. Žilavost omrežja G je spodaj omejena z $\frac{\kappa(G)}{\beta_0(G)}$ in zgoraj omejena z $\frac{n-\beta_0(G)}{\beta_0}$.

11.2.5 Pogojna povezanost

Pogojno povezanost je predstavil Harary [187] in je posplošitev minimalne m-stopnje. Merilo je parametrizirano z lastnostjo P, ki mora veljati za vse dele omrežja, ustvarjene z brisanjem vozlišč iz omrežja.

Definicija 11.8 P-povezanost $\kappa(G : P)$ omrežja G je najmanjše število vozlišč, ki jih moramo izbrisati iz omrežja tako, da ima nastalo omrežje G' sledeče lastnosti :
- G' ni povezan
- Vsak povezan del G' ima lastnost P.
Pogojna povezanost po povezavah je definirana analogno za izbris povezav.

Pogojna povezanost je lahko zelo uporabna v praksi saj lahko lastnost P izberemo glede na značilnosti naloge, ki jo mora omrežje izpolniti. Na primer P bi lahko definirali kot: Del ima največ k vozlišč. Pogojna povezanost se nato ujema z velikostjo najmanjše podmnožice vozlišč, ki jo moramo izbrisati, da omrežje razdelimo na dele, ki imajo največ k vozlišč vsak.
Klasična povezanost je poseben primer pogojne povezanosti, kjer $P = \emptyset$.

Če definiramo zaporedje lastnosti $S = (P_1, \ldots, P_k)$ glede na naše zahteve tako, da P_{i+1} implicira oziroma vsebuje P_i za $1 \leq i \leq k-1$, dobimo vektor pogojne povezanosti

$$(\kappa(G : P_1), \ldots, \kappa(G : P_k)).$$

Če so lastnosti definirane tako da modelirajo naraščajočo degradacijo omrežja torej da narašča število odstranjenih vozlišč ali povezav, ta vektor poda zgornje meje za uporabnost sistema glede na število odstranjenih vozlišč.

Podobno merilo je splošna povezanost, uvedel jo je Harary [188].
Če je G omrežje z lastnostjo P in Y podmnožica vozlišč ali povezav omrežja G, potem je $\kappa(G, Y : P)$ najmanjši del $X \subset Y$ vozlišč ali povezav v G katerih odstranitev se kaže v omrežju G', ki nima lastnosti P. Pogojna povezanost je poseben primer splošne povezanosti.

Glavna pomanjkljivost teh meril je, da ne obstaja učinkovit algoritem, ki bi to izračunal za splošno omrežje.

11.3 Najslabši primer merila: razdalja

Merila v tem poglavju govorijo o povečanju razdalj v mrežah, ki jih povzroča odstranjevanje vozlišč ali povezav. To so spet 'worst case statistics', saj podajo najmanjše število vozlišč ali povezav, ki jih je potrebno odstraniti, da povečamo razdaljo. Vsa merila, ki so predstavljena v tem poglavju, so definirana dokler mreža ne postane nepovezana zaradi odstranitve vozlišč ali povezav.

11.3.1 Vzdržljivost

Vzdržljivost mreže je minimalno število vozlišč, ki jih moramo odstraniti, da povečamo premer (najdaljša razdalja med parom vozlišč v mreži). Analogno lahko definiramo pojem *vzdržljivost na povezavah* za odstranjevanje povezav. Vzdržljivost so predstavili Boesch, Harary in Kabell [189]. Oni so tudi predstavili naslednje lastnosti vzdržljivosti mreže:

- Vzdržljivost mreže s premerom $2 \leq d \leq 4$ je enaka minimumu med vsemi pari nesosednjih vozlišč i in j z največjim številom nepovezanih oglišč i, j-poti dolžine ne več kot d.
- Vzdržljivost povezav (povezavna vzdržljivost) mreže s premerom $d \in \{2, 3\}$ je minimum po vseh parih vozlišč i, j z največjim številom nepovezanih povezav i, j-poti dolžine ne več kot d.

Veliko je teoretičnih rezultatov za vzdržljivost, ki v glavnem ugotavljajo povezave med povezanostjo in vzdržljivostjo. *Vektor vzdržljivosti* je razširitev pojma vzdržljivosti. i-ta komponenta $P(G) = (p_1, \ldots, p_n)$ je premer grafa G v najslabšem primeru, če odstranimo i-to vozlišče. Ta pojem je enak kot zaporedje premerov izbrisanih vozlišč, ki je predstavljeno v sledečem poglavju.

11.3.2 Naraščajoča razdalja in zaporedje premerov

Krishnamoorthy, Thulasiraman in Swamy so proučevali porast razdalje v mrežah, ki je nastala zaradi brisanja vozlišč in povezav [190]. Za mrežo G so predstavili štiri zaporedja A, B, D in T definirana kot:

Definicija 11.9 Naj bo $d(u, v) = d_G(u, v)$ razdalja med dvema vozliščema u in v v G. Naj bo l povezanost z vozlišči od G in naj bo m povezavna povezanost. Potem so zaporedja A, B, D in T definirana tako:

$$a_i = max_{|V_i|=i}\{d_{G-V_i}(u, v) - d(u, v) | u, v \in V - V_i\} \text{ za } 1 \leq i \leq l - 1$$

$$b_i = max_{|E_i|=i}\{d_{G-E_i}(u, v) - d(u, v)\} \text{ za } 1 \leq i \leq m - 1$$

$$d_i = max_{|V_i|=i}\{d(G - V_i)\} \text{ za } 1 \leq i \leq l - 1$$

$$t_i = max_{|E_i|=i}\{d(G - E_i)\} \text{ za } 1 \leq i \leq m - 1.$$

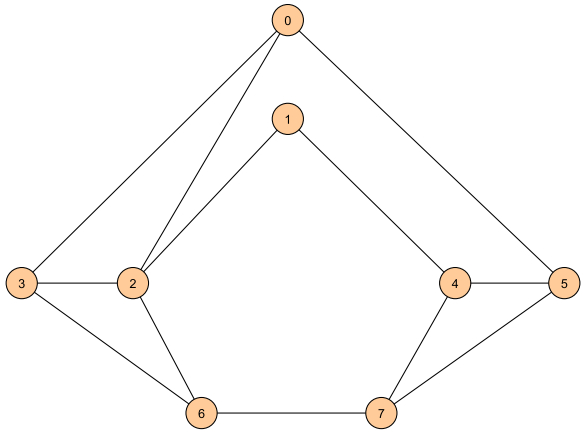
Zaporedje A se imenuje zaporedje naraščajoče razdalje z brisanjem vozlišč, B zaporedje naraščajoče razdalje s pomočjo brisanja povezav, D zaporedje premerov z brisanjem vozlišč in T zaporedje premerov z brisanjem povezav.

Število i v zaporedju A je največje povečanje razdalje med parom vozlišč, ki jo povzroči brisanje i vozlišč iz G. Zaporedje B vsebuje največjo povečanje razdalje zaradi brisanja povezav. Vnos i v zaporedju D je največji premer grafa zaradi brisanja i vozlišč, in zaporedje T je analogno zaporedje za brisanje povezav. Tabela 2 vsebuje štiri zaporedja za mrežo predstavljeno na sliki 6.

Lahko je opaziti, da so zaporedja A, B in T vedno monotono nepadajoča. Elementi zaporedja A so nenegativni in elementi zaporedja B so vsaj 1. Če je G popoln, potem zgledajo zaporedja tako:

A	(1, 2)
B	(3, 3)
D	(3, 4)
T	(4, 4)

Tabela 11.2: Zaporedja brisanja vozlišč in povezav za mrežo iz spodnje sheme.



Slika 11.6: Primer omrežja za zaporedja postopne rasti razdalje

- $A = (0, \ldots, 0)$
- $B = (1, \ldots, 1)$
- $D = (1, \ldots, 1)$
- $T = (2, \ldots, 2)$

Krishnamoorthy, Thulasiraman in Swamy so pokazali, da največjo razliko v razdalji med poljubnim parom vozlišč, ki jo povzroči brisanje i vozlišč ali povezav, lahko vedno najdemo med sosedi izbrisanih objektov. Ta ugotovitev opazno pospeši izračun zaporedij in tudi poenostavi definiciji zaporedij A in B. Ti dve zaporedji lahko definiramo tudi sledeče (opazimo, da je $N(V_i)$ množica sosednjih vozlišč vozliščem iz množice V_i in $N(E_i)$ je množica vozlišč, ki se stikajo s povezavami iz E_i):

$$a_i = \max_{|V_i|=i} \{d_{G-V_i}(u, v) - d(u, v) | u, v \in N(V_i)\} \text{ za } 1 \leq i \leq l - 1$$

$$b_i = \max_{|E_i|=i} \{d_{G-E_i}(u, v) - d(u, v) | u, v \in N(E_i)\} \text{ za } 1 \leq i \leq l - 1$$

Zaporedja brisanja vozlišč in povezav so mere za najslabši primer povečanje razdalje, ki jo povzroči pomanjkanje vozlišč ali povezav in ne povejo nič o stanju grafa po tem, ko se povezava prekine. Ta merila so primerna samo za aplikacije, kjer je razdalja ključnega pomena in prekinitev povzroči neuporabnost celotnega omrežja. Kljub izboljšavam omenjenim zgoraj, je izračun zaporedij možen samo za grafe z nizko povezanostjo.

11.4 Povprečno merilo robustnosti

Metoda v tem poglavju ugotavlja povprečno število vozlišč ali povezav, ki lahko manjkajo, da bi imelo omrežje še vedno določeno lastnost ali strukturo povprečja lokalnih lastnosti, v smislu, da bi še vedno pokrivalo podobo globalnega omrežja.

11.4.1 Povprečna povezanost

Povprečna povezanost, katero je predstavil Tainiter [191], [192], govori o verjetnosti, da je omrežje izključeno po izbrisu naključnega števila povezav v omrežju.

Definicija 11.10 Naj bo $G = (V, E)$ povezano omrežje z n vozlišči in m povezavami. Naj bo $S(G)$ vrsta vseh $m!$ ureditev povezav in $G_0 = (V, \emptyset)$. Za vsako ureditev $s \in S(G)$ definiramo število $\xi(s)$, in sicer: Povezave omrežja G vstavimo v G_0, v zaporedju kot ga narekuje s. $\xi(s)$ definiramo kot indeks povezave, ki spremeni omrežje iz nepovezanega v povezano. Potem je povprečna povezanost omrežja G definirana kot

$$\mathcal{M}(G) = m - \frac{1}{m!} \sum_{s \in S(G)} \xi(s)$$

Lastnosti povprečne povezanosti:

- Če je G = (V,E'), kjer je $E' \subseteq E$, povezano podmrežje omrežja $G = (V, E)$, potem je $\mathcal{M}(G') \leq \mathcal{M}(G)$.
- Naj bo G omrežje z n vozlišči in m povezavami. Konstruiramo novo omrežje G', kjer dodamo eno novo vozlišče in h povezav, katere povežejo novo vozlišče z vozlišči v grafu G. Naj bo $\mathcal{M}(G, k)$ število zaporedja povezav za G z $\xi(s) = k$. Potem velja naslednja neenakost:

$$\mathcal{M}(G') - \mathcal{M}(G) \geq \frac{\mathcal{M}(G) - 1}{m + 1} - \frac{1}{h + 1} \sum_{k=n-1}^{m} \mathcal{M}(G, k) \frac{(h + m - k + 1)!}{(m - k)!(m + h)!}$$

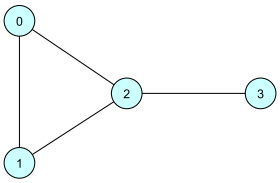
- Veljajo meje:

$$\lambda(G) - 1 \leq m - n + 1,$$

kjer je $\lambda(G)$ povezanost omrežja G. Primer, ko sta obe meji močni je za cikel, kjer velja $\lambda(G) = 2$ in $\mathcal{M}(G) = 1$.

Problem 11.11 *Povprečna povezanost omrežja je $\mathcal{M}(G) = \frac{3}{4}$. Za vsako zaporedje povezav, kjer je povezava $(2, 3)$ zadnja, je $\xi(s) = 3$. Za vsa druga zaporedja imamo $\xi(s) = 4$. Ker imamo 6 zaporedij kjer je povezava $(2, 3)$ zadnja in 24 vseh povezav, je torej povprečna povezanost omrežja enaka $\frac{3}{4}$. Opazimo, da $\mathcal{M}(G)$ ni enaka povprečnemu številu povezav, ki jih moramo zbrisati, da omrežje G ne bo povezano. Če pogledamo vsa zaporedja brisanja povezav in izračunamo povprečni indeks, kjer omrežje postane nepovezano, dobimo vrednost $\frac{7}{4}$ za zgornje omrežje.*

Če je razlika med povprečno in klasično povezanostjo velika, potem se v omrežju pojavi "ozko grlo"povezljivosti. Povezljivost omrežja lahko okrepimo tako, da vstavimo nekaj povezav da premostimo ta zastanek.



Slika 11.7: Omrežje s povprečno povezanostjo $\frac{3}{4}$.

Problem 11.12 *Imamo omrežje z enim visečim vozliščem, ki je povezana s preostalim omrežjem samo z eno povezavo. Z vsako povezavo, ki jo dodamo temu vozlišču, lahko povečamo povezanost omrežja za ena.*

Slaba stran merila povprečne povezanosti je ta, da ne obstaja učinkovit algoritem za ocenjevanje oziroma za računanje tega.

11.4.2 Povprečna razdalja povezave in razdrobitev

V 90ih letih so Albert, Jeong in Barabasi [193] naredili simulacijo naključnega manjkanja vozlišča in namenskega napada na vozlišče visoke stopnje v naključnih in brezlestvičnih (scale-free networks) omrežjih. Merili so učinke na dva parametra v omrežju, in sicer na povprečno razdaljo povezave in na razdrobitev.
Povprečna razdalja povezave $\bar{d}$ je povprečna dolžina najkrajše poti med povezanimi pari vozlišč v omrežju.
Razdrobitev pa meri razpad omrežja, v smislu, kako velike so njegove povezane komponente.

Definicija 11.13 (razdrobitev) Naj bo G omrežje s k povezanimi komponentami $S_1, ..., S_k$. Razdrobitev $frag(G) = (frag_1(G), frag_2(G))$ je definirana z dvemi parametri:
RELATIVNA VELIKOST največje komponente

$$frag_1 = \frac{\max_{i=1,...,k}|S_k|}{\sum_{i=1}^{k}|S_k|}$$

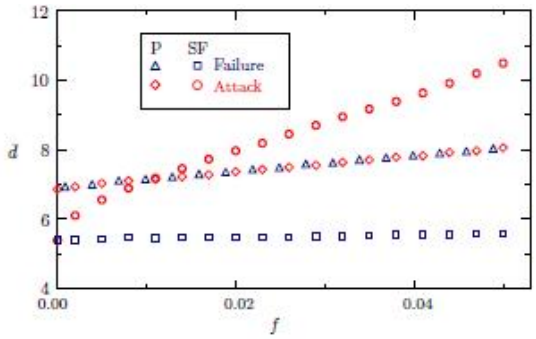
POVPREČNA VELIKOST izolirane komponente

$$frag_2 = \frac{\sum_{i=1}^{k}|S_k| - \max_{i=1,...,k}|S_k|}{k-1},$$

kjer $|S_k|$ pomeni število vozlišč v k-ti komponenti.

Slika 8 prikazuje učinek razdora vozlišč in napadov na povprečno razdaljo povezave $\bar{d}$ naključnih omrežij, katerih stopnja porazdelitve sledi Poissonovi in potenčni porazdelitvi. Poissonova omrežja enakovredno prenašajo naključne in namenske napade. Vsako vozlišče ima enako vlogo in če izbrišemo eno izmed njih, to v povprečju malenkostno vpliva na povprečno razdaljo povezave ali pa sploh ne. Nasprotno, pa so brezlestvična omrežja zelo robustna do razdorov, v smislu povprečne razdalje povezave. Verjetnost, da je izbrisano vozlišče visoke stopnje, je majhna in ker so ta vozlišča pogoj za majhno povprečno razdaljo v brezlestvičnih omrežjih, se razdalja skoraj nič ne poveča, če izbrišemo vozlišča

naključno. Če pa so ta vozlišča ciljna za napad, potem se povprečna razdalja povezave hitro poveča. Simulacije na grafih majhnih delov internetnih usmerjevalnikih in na WWW grafih prikazujejo podobno obnašanje kot naključno brezlestvično omrežje.



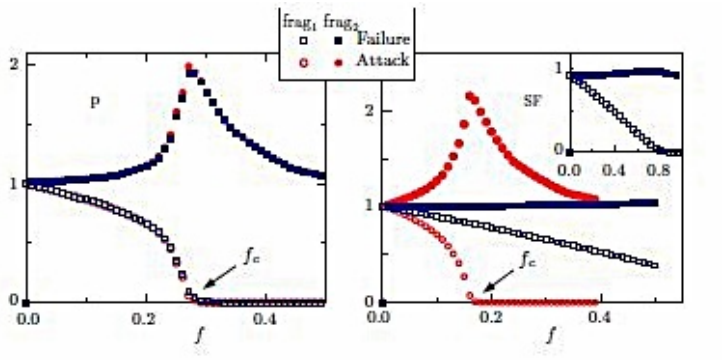
Slika 11.8: Spremembe povprečne razdalje povezav $\bar{d}$ naključno generiranih omrežij ($|V| = 10.000, |E| = 20.000$) s Poissonovo (P) in brezlestvično (SF) porazdelitvijo, po naključnem izbrisu $f|V|$ vozlišč.

Samo povečanje povprečne razdalje povezav $\bar{d}$ ne pove veliko o povezljivosti omrežja, ko nastopi razdrobitev. Mogoče je ustvariti omrežja z majhnim $\bar{d}$, ki vsebuje veliko nepovezanih komponent (npr., veliko nepovezanih trikotnikov, kjer je $\bar{d} = 1$). Zato je Albert meril proces razdrobitve pod napadi in razdori.

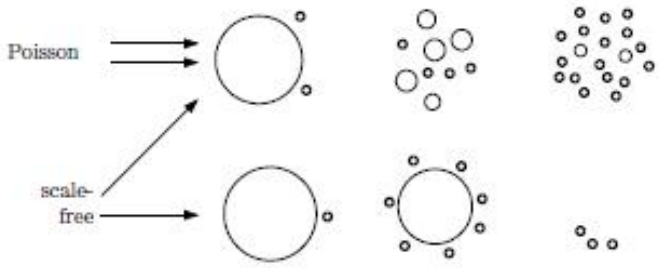
Slika 9 prikazuje rezultate študije razdrobitve. Poissonovo omrežje se obnaša kot, da ima prag pri $f > f_c \approx 0.28$, ko postane $frag_1$, relativna velikost največje komponente, skoraj enača nič. Skupaj z obnašanjem $frag_2$, povprečna velikost izključenih komponent, dosežeta vrh pri 2. To kaže na zlom, kot je prikazano tudi na sliki 10. Odstranitev nekaj vozlišč izključi le posamezna vozlišča. Komponente postanejo večje ko f doseže filtrirni prag f_c. Nato sistem razpade. Kot na sliki 2, so rezultati enaki za naključne in namenske razdore v omrežjih z Poissonovo porazdelitvijo.
Postopek zgleda drugačen pri brezlestvičnih omrežjih (podatki za usmerjevalnik in WWW grafih zgledajo podobni kot pri naključno generiranih brezlestvičnih omrežjih). Za naključni izbris vozlišč ne moremo opaziti filtrirnega praga: Tako obnašanje omrežja se imenuje *elegantna degradacija (graceful degradation)*. V primeru napadov, opazimo enak zlom kot pri Poissonovem omrežju, s filtrirnim pragom $f_c \approx 0.18$, kater nastopi prej.

Torej, brezlestvična omrežja so tolerantna do naključnih razdrobitev, vendar pa so občutljiva na napade usmerjene v točno določeno vozlišče. Ker ima internet brezlestvično strukturo, so študije pokazale občutljivost tega omrežja in ta vrsta omrežja se tako imenuje 'Ahilova peta interneta'.

Broder je študiral strukturo omrežja bolj temeljito in prišel do zakjučka, da ima ta obliko pentlje (bow tie structure). Njegovi rezultati eksperimenta na omrežju W so odkrili, da je "world wide web"robusten za napade. Izbris vseh vozlišč $\{v \in V(W)|d^-(v) \geq 5\}$ ne zmanjša velikosti največje komponente, še vedno vsebuje približno 30% vozlišč. To



Slika 11.9: Spremembe pri razdrobitvi $frag = (frag_1, frag_2)$ naključnega omrežja (P za Poissonovo porazdelitev, SF za brezlestvično porazdelitev), ko naključno odstranimo $f|V|$ vozlišč. Slikica v zgornjem desnem kotu prikazuje scenarij za niz izbrisov v brezlestvičnih omrežjih.



Slika 11.10: Zlom omrežij s Poissonovo in brezlestvično porazdelitvijo.

protislovje z simulacijami opisanimi na začetku je očitno, saj je

$$\frac{|\{v \in V(W)|d^-(v) \geq 5\}|}{V(W)}$$

še vedno pod filtrirnim pragom za robustnost in tako je le drug način za ogled istih podatkov, čeprav se sliši "izbris vseh vozlišč z visoko stopnjo"drastično, je to še vedno osnova.
Številne študije na to temo vključujejo povprečno razdaljo povezave in razdrobitev kot merilo, da bi dokazali lastnosti robustnosti ustreznega omrežja.

Problem 11.14 *Jeong je preučeval kako omrežje proteinov vzajemno delujejo v zaporedju vseh proteinov, ter pokazal da je robustno do naključnih mutacij proteinov, vendar občutljivo do uničenja proteinov visoke stopnje [194]. Uporaba povprečne razdalje povezave in razdrobitev za preučevanje širjenja epidemičnega omrežja je pokazala, da je potrebno najprej poskrbeti za središčne točke, ko uporabljamo strategijo cepljenja [195].*
Holme [196] je preučeval malo bolj komplekne napade na omrežja. Poleg napada na vozlišča je preučil tudi izbris povezav, ter izbral vmesnost osrednje lege, kot alternativna izbira merila za izbris. Poleg tega, pa je raziskal kako ta preračunana merila, po vsakem izbrisu, spremenijo rezultat. Empirično je pokazal, da so napadi, ki temeljijo na preračunanih vrednostih, bolj učinkoviti.

Bolj teoretično študijo pa je naredil Cohen, ter neodvisno tudi Callaway. In sicer, analizirala sta proces razdrobitve na brezlestvičnih omrežjih. Medtem, ko je Cohen [197] uporabil teorijo filtriranja, je Callaway [198] uporabil generirne funkcije(generating functions) in dobil bolj splošne rezultate za poljubno stopnjo porazdelitev. eoretične analize so potrdile rezultate empiričnih študij in potrdile filtrirni prag, ki je določen z $\frac{|\{v \in V(W) | d^-(v) \geq 5\}|}{V(W)}$.

11.4.3 Odpornost na uravnotežen rez

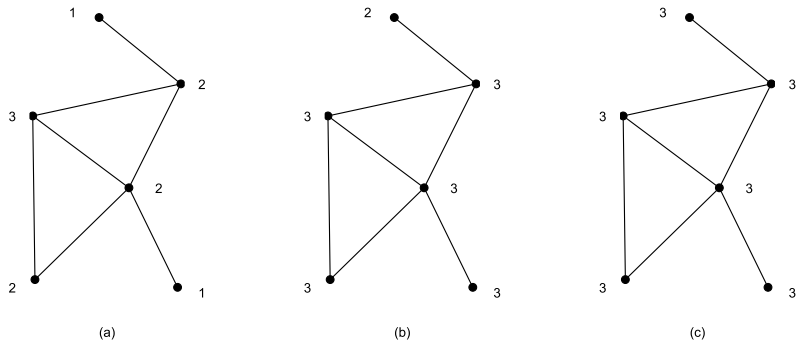
Poleg drugih meril, Tangmunarunkit uporablja novo merilo za robustnost, da poveže napake v poskusni študiji [199]. Cilj poskusa je ocena generatorja, ki simulira topologijo interneta. Poleg raztezanja in izkrivljanja omrežja, se meri tudi podobnost med generiranimi in realnimi omrežji z uravnoteženimi rezi (balanced-cut). Torej pravimo, da je omrežje odporno na okvare, če je povprečna velikost uravnoteženega reza velika, znotraj h-sosednosti vsakega vozlišča.

Definicija 11.15 (Odpornost na uravnotežen rez) Naj bo $G = (V, E)$ omrežje z n vozlišči in kapaciteta vsake povezave v G je enaka 1. Minimalni uravnotežen rez omrežja G je kapaciteta minimalnega reza, tako da dve dobljeni vozlišči vsebujeta približno isto število vozlišč, to je $\lfloor \frac{n}{2} \rfloor$ in $\lceil \frac{n}{2} \rceil$. Odpornost na uravnotežen rez $R(N(v,h))$ je povprečna velikost minimalnega uravnoteženega reza znotraj h-sosednosti $Neigh_h(v)$ okoli vsakega vozlišča v, to je

$R(N(v,h)) = \frac{1}{n}(\sum_{v \in V}$ minimalni uravnotežen rez v $Neigh_h(v))$.

- h-sosednost vozlišča v vsebuje vsa vozlišča z razdaljo manjšo ali enako h od v.
- Odpornost na uravnotežen rez je funkcija števila vozlišč $N(v,h)$ v h-sosednosti točke v in izključimo dejstvo da ima razširjeno omrežje več vozlišč in sosednosti v istem radiju. Očitno je $R(h) = 1$ za poti in drevesa. Odpornost za naključne grafe v Erdos-Renyi modelu s povprečno stopnjo k je proporcionalno kn, medtem ko je proporcionalno n-ju za končne grafe.
- Za navadne grafe je odpornost na uravnotežen rez enaka $\sqrt{n}$.

Kot ilustrativen primer poglejmo sliko 11.



Slika 11.11: Uravnotežen rez za vozlišča (a) 1 - sosednost (b) 2 - sosednost (c) 3 - sosednost

Oceniti minimalni uravnotežen rez je $\mathcal{NP}$ - težko [200] in tako je slaba stran tega merila računska zahtevnost, kar ga naredi nepraktičnega za velika omrežja. Obstajajo metode, ki dajejo dobre vrednosti tako, da je odpornost na uravnotežen rez vsaj ocenjena. Karypis and Kumar [201] sta predstavila več ravni delitve metode s časovno zahtevnostjo $\mathcal{O}(m)$, kjer je m število povezav v omrežju.

11.4.4 Učinkovit premer (Effective diameter)

Palmer je predstavil učinkovito ekscentričnost in učinkovit premer kot merili odpornosti razdora vozlišč in povezav [202]. Ti merili temeljita na hop-plot in ju definiramo:

Definicija 11.16 (Učinkovita ekscentričnost, učinkovit premer) Učinkovita ekscentričnost (effective eccentricity) $\varepsilon_{eff}(v,r)$, $0 \leq r \leq 1$, vozlišča v je najmanjši h tako, da število vozlišč $N(v,h)$ znotraj h-sosednosti vozlišča v je najmanj r-kratnik vseh vozlišč, to je
$\varepsilon_{eff}(v,r) = \min\{h \in \mathbb{N} | N(v,h) \geq rn\}$.

Učinkovit premer omrežja je najmanjši h, pri katerem je število parov znotraj h-sosednosti vsaj r-kratnik vseh dosegljivih vozlišč :

$$diam_{eff}(r) = \min\{h \in \mathbb{N} | P(h) \geq rP(\infty)\}$$

kjer je P število parov v določeni sosednosti (hop-plot), to je

$$P(h) := |\{(u,v) \in V^2 | d(u,v) \leq h\}| = \sum_{v \in V} N(v,h)$$

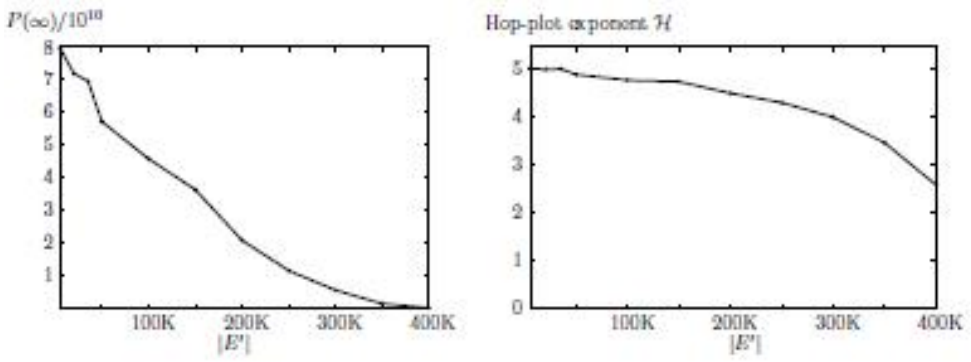
Narejen je bil poskus na 285.000 usmerjevalnikih, da bi ugotovili kako in pod katerimi pogoji se učinkovit premer spreminja. Lahko se izbriše povezave ali točke omrežja. Nov izračun oziroma ocena učinkovitega premera po vsakem izbrisu, pri vrednosti $r = 0.9$ in z uporapo funkcije sosednosti, je enaka 400.

Problem 11.17 *Spodnji sliki prikazujeta posledice razdora povezav in usmerjevalnika na internetnem grafu. Torej lahko potrdimo, da je internet zelo robusten do naključnih izbrisov, vendar zelo občutljiv do izbrisa točk visoke stopnje. Tudi izbris točk z nizko učinkovito ekscentričnost postopoma zmanjšuje povezljivost.*

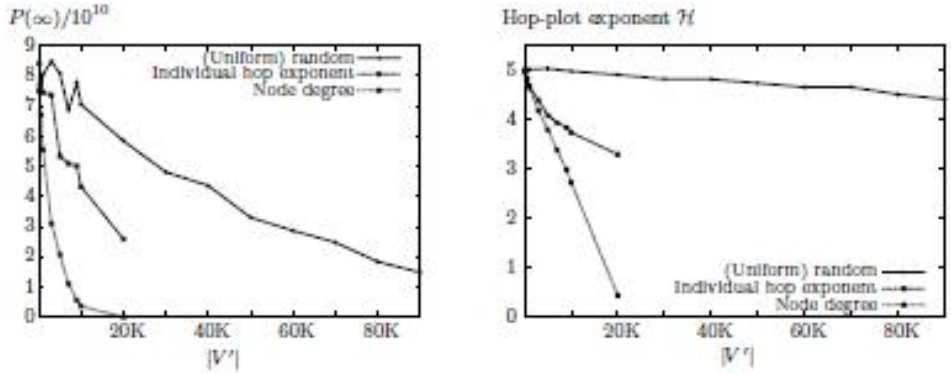
11.5 Verjetnostno merilo robustnosti

To poglavje opisuje merila za robustnost, ki eksplicitno upošteva verjetnostne napake komponent omrežja in so zato primernejša za opis napak na neciljanih komponentah. Tukaj predstavimo dva različna pristopa, da določimo verjetnost prekinitve omrežja s pomočjo danih verjetnostnih napak: *zanesljivostni polinom* in *verjetnostna odpornost*.

Tukaj se nismo osredotočili na čisto teoretični pristop kot je naprimer simbolični pristop k robustnosti, ki ga obravnava Flajolet in drugi [203], kjer avtorji definirajo merilo robustnosti z določitvijo pričakovanega števila povezavno nepovezanih poti, da pridemo iz začetnega vozlišča s do željenega vozlišča t v grafu.



Slika 11.12: Učinek na izbris povezav na omrežju z 285.000 usmerjevalniki. Množica E' označuje izbrisane povezave.



Slika 11.13: Učinek na izbris vozlišč (razdor usmerjevalnika) na omrežju z 285.000 usmerjevalniki. Množica V' označuje izbrisane povezave.

11.5.1 Zanesljivostni polinom

Zanesljivostni polinom sta že omenila Boorstyn in Frank leta 1977[204].

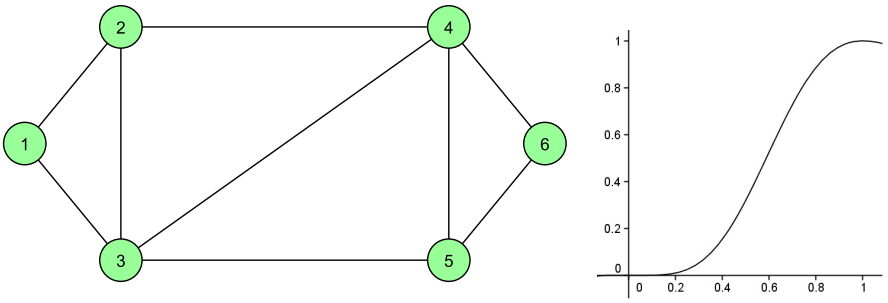
Definicija 11.18 Naj bo G povezano omrežje z n vozlišči in m povezavami. Predpostavimo, da povezave odpovejo neodvisno druga od druge z verjetnostjo $1 - p$, kjer je $0 \leq p \leq 1$. Zanesljivostni polinom $R(G, p)$ je verjetnost, da je omrežje G povezano.

Očitne lastnosti zanesljivostnega polinoma $R(G, p)$ so:

1. $R(G, 0) = 0$, $R(G, 1) = 1$.
2. iz $p_1 < p_2$ sledi, da je $R(G, p_1) < R(G, p_2)$.
3. Naj bo G povezan graf in G_{-e} graf, ki ga dobimo, če grafu G odstranimo povezavo e. Naj bo G_e graf, ki ga dobimo s skrčitvijo povezave e. Potem velja naslednja enakost:
$$R(G, p) = (1 - p)R(G_{-e}, p) + pR(G_e, p).$$
4. Če je G drevo z m povezavami, potem je $R(G, p) = p^m$.

V svoji doktorski tezi je Rosenthal[206] pokazal, da je določiti verjetnostno napako NP-problem, če je verjetnost, da je mreža povezana vsaj neka znana vrednost q. Enako velja, če imamo podano verjetnostno napako za vozlišča in povezave. Pönitz in Tittmann[205] sta pokazala, da je problem možno rešiti v času $\mathcal{O}((2n+m)B(k))$ za grafe s širino poti k, kjer je $B(k)$ Bellovo število. Belovo število določa število particij množice s k elementi na neprazne podmnožice. Sledi, da je problem rešljiv v polinomskem času za grafe s povezanimi širinami poti. Spodnja shema prikazuje graf s širino poti enako 2, skupaj z narisanim grafom njegovega zanesljivostnega polinoma. Polinom ima sledečo formulo:

$$R(G,p) = 55p^5 - 155p^6 + 169p^7 - 84p^8 + 16p^9$$



Slika 11.14: Graf in pripadajoči izris zanesljivostnega polinoma

Ni znanega algoritma, ki bi znal izračunati zanesljivostni polinom za poljubni graf v polinomski časovni zahtevnosti.

11.5.2 Verjetnostna odpornost

V nasprotju z determinističnimi merami predstavljenimi v poglavju o najslabšem primeru merila povezanosti sta Najjar in Gaudiot[207] obravnavala verjetnostno različičo povezanosti. Avtorja vzameta razred regularnih omrežij in preučujeta verjetnost prekinitev s pomočjo naključnih napak na vozliščih. *Verjetnost prekinitve* omrežja G definiramo kot

$$P(G,i) = Pr[G \text{ prekinitev natanko po i-ti napaki}]$$

Motivirani s strani arhitekture ogromne skale računalniških omrežij, so avtorji preučevali družino $\mathcal{F}$ k-regularnih grafov, ki vsebujejo tudi npr. toruse in hiperkocke. Izkaže se, da za omrežja v $\mathcal{F}$ verjetnost prekinitve $P(G,i)$ lahko aproksimiramo z naslednjim izrazom:

$$P_1(G,i) = Pr[G \text{ prekinitev po natanko i-ti napaki}$$

$$\text{in ena komponenta vsebuje natanko eno vozlišče}],$$

to pomeni, verjetnost prekinitve lahko ocenimo z verjetnostjo prekinitve zgolj enega vozlišča iz omrežja. Za omrežja v družini $\mathcal{F}$ lahko $P_1(G,i)$ in zato tudi približek $P(G,i)$ dobimo analitično.
Funkcija $P(G,i)$ ima krivuljo v obliki zvonca, katerega višina se povečuje z n, tj. številom

vozlišč v omrežju, medtem ko je x koordinata maksimuma odvisna od k, tj. stopnje vozlišč (glej spodnjo sliko). Večja kot je povezanost regularnega omrežja glede na k, več napak je potrebnih, da se pojavi prekinitev. Avtorji potrdijo svoje teoretične hipoteze z izvajanjem poizkusov Monte-Carlo na velikem številu grafov iz $\mathcal{F}$.
Pojem verjetnosti prekinitve nam omogoči verjetnostno različico povezanosti: *verjetnostna odpornost*. Intuitivno, odporno omrežje bi moralo vzdržati veliko število napak na vozliščih, preden bi prišlo do prekinitve.

Definicija 11.19 (Verjetnostna odpornost). Naj bo G omrežje z n vozlišči. *Verjetnostna odpornost* $res_{prob}(G,p)$ je največje število vozliščnih napak tako, da je G še vedno povezano z verjetnostjo $1-p$, to je,

$$res_{prob}(G,p) = \max\{I | \sum_{i=1}^{I} P(G,i) \leq p\}.$$

Relativna verjetnostna odpornost se nanaša na $res_{prob}(G,p)$ glede na velikost G:

$$\overline{res}_{prob}(G,p) = \frac{res_{prob}(G,p)}{n}.$$

Očitno je to verjetnostno merilo povezano s klasično povezanostjo in velja enakost $res_{prob}(G,0) = \kappa(G) - 1$. Analiza $P(G,i)$ za regularne grafe pokaže, da verjetnostna odpornost $res_{prob}(G,p)$ narašča z velikostjo G. Po drugi strani pa relativna verjetnostna odpornost $r\bar{e}s_prob(G,p)$ pada z velikostjo, če stopnja omrežja ostane konstantna. Tako relativna odpornost narašča za hiperkocke in pada za toruse z naraščajočo velikostjo omrežja.
Za bolj kompleksne družine mrež kot je $\mathcal{F}$, je precej težko izračunati verjetnostno odpornost. Celo v tem primeru, lahko $P(G,i)$ kvečjemu ocenimo. Kljub temu zgleda verjetnostna različica povezanosti primerno orodje za opis degradacije pod slučajnimi napakami komponent. Zaradi analitične kompleksnosti bo najverjetneje v uporabi zgolj pri empiričnih vrednotenjih.

Poglavje 12

Risanje velikih grafov s pomocjo linearne algebre

Gasper Azman

V tem poglavju se bomo ukvarjali z implementacijo algoritma, primernega za risanje velikih grafov. Velik graf bomo definirali kot 10000 točk ali več. Pri tako velikih grafih klasične metode za risanje, ki so večinoma osnovane na silah med točkami, odpovejo, saj zahtevajo vsaj $O(m + n)$ operacij na iteracijo, in konvergirajo počasi, saj navadno za konvergenco rabimo $O(n)$ iteracij. Poleg sta tako rezultat kot hitrost odvisna od dobre izbire začetne pozicije, kar otežuje analizo. Večji od teh dveh problemov je rezultat, saj se iterativne metode prerade ujamejo v lokalne minimume, kar pomeni, da o globalni sliki ne moremo povedati ničesar. V tem primeru so linearne metode odlične, saj se ukvarjajo predvsem z zvesto globalno sliko, ki jo lahko z uporabo nekaj iteracij kake iterativne metode popravimo do tega, da lepo predstavi tudi lokalne detajle. Algoritma, ki ju bomo predstavili, uporabljata linearno algebro, njuna časovna zahtevnost pa je boljša kot kvadratna.

Za lažje razumevanje bomo predstavili tudi osnovni algoritem MDS, ki je poskrbel za idejo, a je zaradi potratnosti izračuna razdaljne matrike neuporaben.

Glavni vir za to nalogo je bilo doktorsko delo Christiana Picha: *Applications of Multidimensional Scaling to Graph Drawing*, univ. v Konstanzu, 2009.

12.1 Definicije

Razporeditev je seznam točk v $\mathbb{R}^3$ ali $\mathbb{R}^2$, ki pripadajo posameznim točkam grafa. Cilj te naloge je poiskati tako razporeditev.

BFS ali iskanje v širino, je algoritem, s katerim bomo rešili problem poti med dvema točkama v $O(n)$ času. Seveda moramo pri tem predpostaviti, da so vse poti dolge 1.

APSP je ime za problem določitve razdalj med vsemi pari točk v grafu (*angl.* all pairs shortest paths). V tej nalogi bomo za rešitev problema uporabili kar iterirani BFS,

kar nam ta problem reši v $O(n^2)$ za neutežene grafe.

max_eigenpairs(D, k) bo metoda, ki poišče k lastnih parov matrike D, začenši z največjimi po absolutni vrednosti. V ta namen uporabimo potenčno metodo s hotelingovimi redukcijami.

Oznaka $D^{(2)}$ bo označevala matriko, katere elementi so kvadrati elementov matrike D. Vsi vektorji, ki jih obravnavamo, so stolpci.

12.2 Klasični MDS

Klasični MDS je eksaktna metoda za vložitev grafa v R^{n-1}. Je večinoma akademske vrednosti, saj potrebujemo najprej izračunati celotno razdaljno matriko (rešiti problem najkrajših poti med vsemi pari ali APSP), kar nam vzame $O(n^2)$ časa, nato pa še izračunati lastne vrednosti matrike, kar je spet $O(n^3)$, ali pa $O(n^2)$, če se omejimo le na nekaj največjih lastnih vrednosti.

Naj bo D matrika razdalj med vozlišči (neusmerjenega) grafa G. Označimo $d_{ij} = d_G(v_i, v_j)$.

$$D = (d_{ij})_{v_i, v_j \in V(G)}$$

Opazimo: D je simetrična, z ničlami na diagonali, in vsi njeni elementi so nenegativni.

Iščemo tako razporeditev $X = [x_1, \ldots, x_n] \in \mathbb{R}^{3\times d}$, da se bodo razdalje med točkami v $\mathbb{R}^d$ ujemale z razdaljami v grafu, torej

$$\|x_i - x_j\| = d_{ij} \ .$$

Ker je pogoj translacijsko invarianten, lahko zahtevamo še, da je celotna razporeditev centrirana na izhodišče, torej da velja

$$\sum_{i=1}^{n} x_i = 0 \ .$$

Sedaj lahko $\langle x_i, x_j \rangle$ izrazimo kot

$$\langle x_i, x_j \rangle = -\frac{1}{2}(d_{ij}^2 - \|x_i\|^2 - \|x_j\|^2),$$

saj

$$d_{ij} = \|x_i - x_j\|^2 = \langle x_i - x_j, x_i - x_j \rangle = \|x_i\|^2 - 2\langle x_i, x_j \rangle + \|x_j\|^2.$$

Če seštejemo najprej po vrsticah ter uporabimo dejstvo, da je razporeditev centrirana, dobimo za vsak j:

$$\begin{aligned}
\sum_{i=1}^{n} d_{ij} &= \sum_{i=1}^{n} (\|x_i\|^2 - 2\langle x_i, x_j \rangle + \|x_j\|^2) \\
&= \sum_{i=1}^{n} (\|x_i\|^2 + \|x_j\|^2) - \left\langle \left(\sum_{i=1}^{n} x_i \right), x_j \right\rangle \\
&= \sum_{i=1}^{n} (\|x_i\|^2 + \|x_j\|^2) \\
&= \sum_{i=1}^{n} \|x_i\|^2 + n\|x_j\|^2
\end{aligned}$$

in podobno za vsak i. Dobimo:

$$\sum_{i=1}^{n} d_{ij} = \sum_{i=1}^{n} \|x_i\|^2 + n\|x_j\|^2$$
$$\sum_{i=j}^{n} d_{ij} = n\|x_i\|^2 + \sum_{i=j}^{n} \|x_j\|^2$$

Če vzamemo enačbo za j in seštejemo po vseh i:

$$\sum_{i=1,j=1}^{n} d_{ij}^2 = 2n \sum_{i=1}^{n} \|x_i\|^2$$

Zanimivo je, da lahko to dejstvo skupaj s prejšnjo enačbo za $\langle x_i, x_j \rangle$ uporabimo, da pokažemo

$$b_{ij} = \langle x_i, x_j \rangle = -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2 - \sum_{i=1}^{n} d_{ij}^2 + \frac{1}{n^2}\sum_{i=1,j=1}^{n} d_{ij}^2 \right),$$

s čimer smo pokazali, da se da skalarne produkte med vektorji izraziti iz same razdaljne matrike. Označimo $B = (b_{ij})_{i,j=1}^{n} \in \mathbb{R}^{n\times n}$.

Spomnimo se, da je X iskana razporeditev točk v prostoru. Pokazali smo, da velja $X^T X = B$.

Ker je B simetrična, so njene lastne vrednosti in vektorji realni. Dekomponirajmo B v

$$B = U\Lambda U^T = \sum_{i=1}^{n} \lambda_i u_i u_i^T,$$

kjer je $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ in $U = [u_1, \ldots, u_n]$ matrika normiranih lastnih vektorjev u_i. Nadalje zahtevamo, da velja za vsak $i : \lambda_i \geq \lambda_{i+1}$.

Ker smo privzeli, da so koordinate centrirane, vedno dobimo vsaj eno ničelno lastno vrednost. To je očitno tudi zaradi dejstva, da lahko graf na n točkah vedno eksaktno vložimo v R^{n-1}.

Recimo, da smo dobili d neničelnih lastnih vrednosti. Tedaj lahko graf eksaktno vložimo v $\mathbb{R}^d$.

Originalne koordinate nato dobimo kot

$$X^T = [\sqrt{\lambda_1}u_1, \ldots, \sqrt{\lambda_d}u_d].$$

V praksi nam tako numerične napake kot tudi napake v podatkih ponavadi povzročijo, da dobimo relativno malo ničelnih lastnih vrednosti, čeprav je bil graf na začetku npr. ravninski. Pri rekonstrukciji vsak lastni par (λ_i, u_i) določa eno dimenzijo, katere interpretacija je odvisna od λ_i: če je λ_i velika in pozitivna, ta dimenzija veliko prispeva k dani matriki, in jo moramo upoštevati; majhne lastne vrednosti interpretiramo kot napake, in jih lahko ignoriramo, velike negativne lastne vrednosti pa pomenijo, da smo dobili podatke, ki v osnovi niso bili del $\mathbb{R}^n$ - recimo, ne zadoščajo trikotniški neenakosti. Ker so grafi metrični prostori, za razdaljne matrike grafov takih problemov ne bomo imeli.

Algorithm 21 Klasični MDS

Vhod: Razdaljna matrika $D \in \mathbb{R}^{n \times n}$
Izhod: Razporeditev $X \in \mathbb{R}^{n \times 3}$
  Izračunaj B.
  Dekomponiraj B v $U\Lambda U^T$.
  for $i = 1 \to d$ **do**
    $x_i = \sqrt{\max(\lambda_i, 0)} u_i$
  end for
  return $X = [x_1 x_2 \cdots x_n]$

12.3 Pivot MDS

Pivot MDS je način aproksimacije klasičnega multidimenzionalnega skaliranja, ki porabi le $O(k^2 n)$ operacij, da izračuna sliko, kjer je k število pivotov. Večje je število pivotov, boljša je slika. V praksi pogosto za dobro sliko zadošča že $p = 50$, zato lahko za pivot MDS rečemo, da je linearen.

Naj bo $P \subseteq V(G)$, $|P| = k$. Če $p \in P, v \in V$, je d_{pv} razdalja med p in v v grafu. Naj bo $D \in R^{n \times k}$ matrika razdalj med k pivoti in vsemi vozlišči grafa. Za njen izračun potrebujemo $O(kn)$ operacij (BFS za vsak pivot). Iskali bomo razporeditev, ki kar najbolj ustreza pogojem

$$d_{vp} \approx \|x_v - x_p\|.$$

Ponovimo izpeljavo iz prejšnjega razdelka, in dodatno predpostavimo, da so pivoti dobro porazdeljeni po grafu, tako da je tudi njihov baricenter v ničli:

$$\sum_{p \in P} x_p = 0.$$

Po enakem računu kot prej (a tokrat le za pivote) dobimo

$$\langle x_v, x_p \rangle \approx -\frac{1}{2}(d_{vp}^2 - \langle x_v, x_v \rangle - \langle x_p, x_p \rangle).$$

Spet ponovimo izpeljavo od prej, pri čemer upoštevamo, da imamo le k pivotov, da dobimo

$$\langle x_v, x_p \rangle = -\frac{1}{2}\left(d_{vp}^2 - \frac{1}{k}\sum_{p \in P} d_{vp}^2 - \frac{1}{n} d_{vp}^2 + \frac{1}{nk} \sum_{v \in V, p \in P} d_{vp}^2\right).$$

Tako lahko spet izrazimo delno matriko skalarnih produktov med vektorji. Recimo ji C.

$$C = (\langle x_v, x_p \rangle)_{v \in V, p \in P}$$

Zanimajo nas singularne vrednosti in vektorji te matrike. Izračunamo jih kot lastne vrednosti CC^T, ki pa je $n \times n$, in je zato nočemo izračunati v celoti.

Tu nam na pomoč priskoči asociativnost množenja z matriko in algoritem, ki smo ga uporabili za izračun lastnih vrednosti in vektorjev.

Izračunati želimo

$$\lim_{i \to \infty} (CC^T)^i y,$$

kar storimo tako, da iterativno računamo

$$y_{i+1} = \frac{CC^T y_i}{\|CC^T\|}.$$

Zaradi asociativnosti lahko to preuredimo v

$$\lim_{i\to\infty}(CC^TCC^T\dots CC^T)y = \lim_{i\to\infty} C(C^TC\dots C^TC)C^Ty = C\lim_{i\to\infty}(C^TC)^iC^Ty.$$

Pri tem je C^TC $k \times k$ matrika, kar pomeni, da je majhna, in množenje z njo je hitro.

Algorithm 22 Pivot MDS

Vhod: Razdaljna matrika $D \in \mathbb{R}^{n\times k}$
Izhod: Razporeditev $X \in \mathbb{R}^{n\times 3}$
Izračunaj C.
$H = C^TC$
Izračunaj d lastnih parov H z ortogonalno iteracijo kot v klasičnem MDS. Naj bodo to $y_i \in R^k$.
for $i = 1 \to d$ **do**
$x_i = Cy_i$
end for
return $X = [x_1x_2x_3]$

12.3.1 Izbira pivotov

Ničesar še nismo povedali o tem, kako izbiramo pivote tako, da dosežemo kar najboljšo razporeditev pri nizkem številu pivotov.

Spomnimo se, da moramo pivote izbrati tako, da bo njihov baricenter karseda enak baricentru vseh točk v grafu. Bližje bosta oba centra, bolj natančna bo slika, saj tako izračunamo pozicije vseh ostalih točk v grafu.

Pogojem algoritma z visoko verjetnostjo zadostimo že kar tako, da pivote izberemo naključno, a izkaže se, da jih lahko izbiramo še bolje.

Prvi pivot p_1 izberimo naključno. Izberimo naslednji pivot tako, da bo kar se da da oddaljen od prvega. Izračunamo problem najkrajših poti iz p_1, (BFS ali Dijkstra), nato pa izberemo element, pri katerem je dosežen maksimum. Izbrano vozlišče označimo s p_2. Naslednjega izberemo tako, da izračunamo problem najkrajših poti za p_2 in izberemo tisto vozlišče, ki ima največji minimum oddaljenosti od p_1 in p_2, naslednjega tako, da ima največji minimum oddaljenosti do vseh prejšnjih treh itd.

Iskaže se, da je ta način problematičen za grafe, na katere so obešene dolge poti, saj so le-te potem prekomerno reprezentirane. V izogib temu je najbolje, da vsak k-ti pivot izberemo naključno, za nek primeren k, npr. 10 ali 15.

12.4 Implementacija

Izdelal sem tudi implementacijo tega algoritma in vseh ostalih, potrebnih za njegovo delovanje. Program je napisan v jeziku C++, za risanje uporablja grafično knjižnjico za 3d grafiko OpenSceneGraph *http://openscenegraph.org*) in Boost Graph Library (*www.boost.org/-libs/graph/*) za implementacijo Floyd-Warshallovega algoritma. Uporablja tudi knjižnjico

Algorithm 23 Izbira pivotov po načinu minmax.

Vhod: G graf, k dolžina želenega seznama pivotov
Izhod: P seznam pivotov, D razdaljna matrika od pivotov do vseh vozlišč v $V(G)$.

```
D ∈ R^{n×k}.
p_1 ∈ P = naključno vozlišče iz V(G).
for i = 1 → k do
    d_i = spp(G, p_i)
    p_{i+1} = argmax_{j=1→n} argmin_{l=1→i} d_{ij}
end for
return p = [p_1 p_2 ··· p_k], D
```

Boost Ublas (*www.boost.org/libs/numeric*) za osnovne operacije v linearni algebri. Iskanje lastnih vrednosti sem implementiral sam, kakor tudi vse naprednejše funkcije iz linearne algebre.

Koda je dostopna na zahtevo pod MIT licenco.

Poglavje 13

Hevristike za Identifikacijo Grafov

Leon Lampret

13.1 Naključni Grafi

Grafovski model je množica grafov (običajno končna), skupaj z neko verjetnostno porazdelitvijo. **Porazdelitev stopenj** grafa G je funkcija $\mathbb{N}_0 \to [0,1]$, ki slika $k \mapsto$ (št. vozlišč grafa G stopnje k) / (št. vozlišč grafa G) = (delež vozlišč stopnje k). Obravnavali bomo naslednje tri modele naključnih grafov: ER-, BA-, in WS-model.

Model naključnega grafa tipa **Erdős-Rényi**, oz. **ER-model**, označen z $\mathcal{G}(n,p)$ za $n \in \mathbb{N}$ in $p \in [0,1]$, je graf ki ga dobimo iz praznega grafa $\overline{K_n}$, tako da med vsakim parom vozlišč z verjetnostjo p dodamo povezavo. Avtorja sta ga uvedla leta 1959 [35]. Graf z n vozlišči in m povezavami ima torej verjetnost $p^m(1-p)^{\binom{n}{2}-m}$. Pričakovano število povezav je $\binom{n}{2}p$. Pričakovana povprečna stopnja vozlišč je $\frac{2 \cdot \text{št.povezav}}{\text{št.vozlišč}} = \frac{2\binom{n}{2}p}{n} = (n-1)p$. Alternativno, $\mathcal{G}(n,p)$ lahko konstruiramo induktivno (kot t.i. *grafovski proces*), tako da začnemo z enim vozliščem v_1, nato pa v vsakem koraku dodamo novo vozlišče v_i, in za vsak $j<i$ z verjetnostjo p dodamo povezavo $\{v_i, v_j\}$ v obstoječi graf, ko i teče od 2 do n.

Soroden grafovski model, označen z $\mathcal{G}(n,m)$ za $n,m \in \mathbb{N}$ in $m \leq \binom{n}{2}$, dobimo tako da tvorimo množico vseh grafov z n vozlišči in m povezavami, ter jo opremimo z enakomerno verjetnostno porazdelitvijo (ki je enaka $1/\binom{\binom{n}{2}}{m}$). Opazimo, da je model $\mathcal{G}(n,\frac{1}{2})$ enak množici vseh grafov na n vozliščih (teh je $2^{\binom{n}{2}}$) z enakomerno porazdelitvijo (ki je enaka $p^m(1-p)^{\binom{n}{2}-m} = \frac{1}{2}^m(1-\frac{1}{2})^{\binom{n}{2}-m} = \frac{1}{2}^{\binom{n}{2}-m+m} = 1/2^{\binom{n}{2}}$).

V študiju naključnih grafov se pogosto obravnava situacijo ko n teče proti neskončnosti, p oz. m pa je fiksen. Lahko pa je p oz. m tudi funkcija v odvisnosti od n. Naprimer, da se pokazati, da je **skoraj vsak** graf $G \in \mathcal{G}(n, \frac{2\ln n}{n})$ povezan, kar po definiciji pomeni, da velja $\Pr\big(\text{graf } G \in \mathcal{G}(n, \frac{2\ln n}{n}) \text{ je povezan}\big) \xrightarrow{n\to\infty} 1$. V tej terminologiji iz opazke, da je pričakovano število povezav grafa $G \in \mathcal{G}(n,p)$ enako $\binom{n}{2}p$, po zakonu velikih števil iz teorije verjetnosti sledi, da če velja $\binom{n}{2}p(n) \xrightarrow{n\to\infty} \infty$, potem ima skoraj vsak graf $G \in \mathcal{G}(n,p(n))$ ravno $\binom{n}{2}p(n)$ povezav. Od tod sklepamo, da če velja $n^2p(n) \xrightarrow{n\to\infty} \infty$ (npr. ko je p konstanta), potem modela $\mathcal{G}(n,p)$ in $\mathcal{G}(n,m)$ za $m = \binom{n}{2}p$ postajata enaka, ko $n \to \infty$.

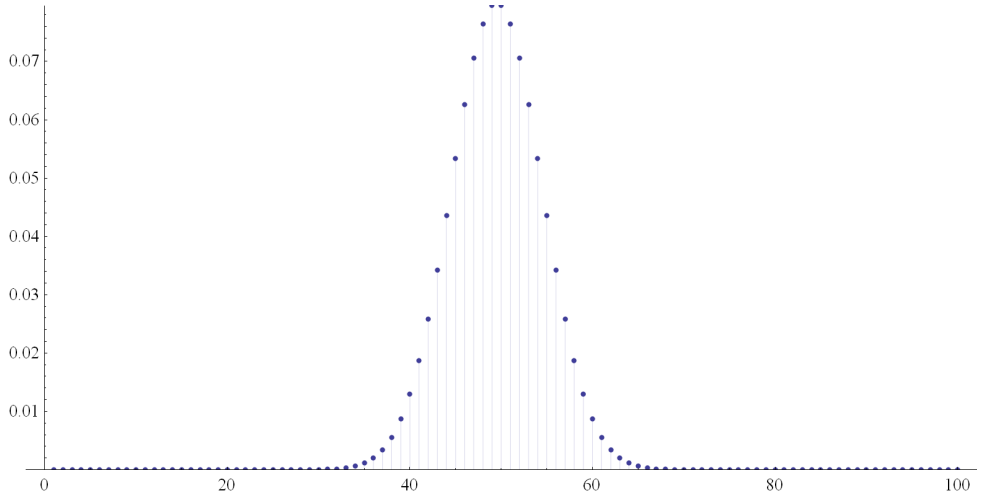
Verjetnost da ima poljubno izbrano vozlišče stopnjo k je $\binom{n-1}{k}p^k(1-p)^{(n-1)-k}$, kar je binomska porazdelitev. To za $np = \text{konst.}$ in $n \to \infty$ konvergira k $\frac{(np)^k e^{-np}}{k!}$, kar je Poissonova porazdelitev. V članku [34] iz leta 1960 sta Erdős in Rényi med drugim dokazala naslednje lastnosti modela $\mathcal{G}(n,p)$.

- Za $np < 1$ graf $G \in \mathcal{G}(n,p)$ skoraj gotovo nima komponent za povezanost, večjih od $O(\log n)$.
- Za $np = 1$ graf $G \in \mathcal{G}(n,p)$ skoraj gotovo vsebuje največjo komponento za povezanost, ki je velikostnega reda $n^{2/3}$.
- Za $np \xrightarrow{n\to\infty} \text{konst.} > 1$ graf $G \in \mathcal{G}(n,p)$ skoraj gotovo vsebuje natanko eno *ogromno* komponento (tj. tako ki vsebuje nek konstanten delež vseh vozlišč v grafu), nobena druga komponenta pa ne vsebuje več kot $O(\log n)$ vozlišč.
- Za $p < \frac{(1-\varepsilon)\ln n}{n}$ graf $G \in \mathcal{G}(n,p)$ skoraj gotovo vsebuje izolirana vozlišča in zato ni povezan.
- Za $p > \frac{(1-\varepsilon)\ln n}{n}$ je graf $G \in \mathcal{G}(n,p)$ skoraj gotovo povezan.

Pričakovan delež vozlišč stopnje k v naključno izbranem grafu $G \in \mathcal{G}(n,p)$ je enaka $\Pr(\text{vozlišče ima stopnjo } k) =$

$$\binom{n-1}{k} p^k (1-p)^{(n-1)-k}.$$

Porazdelitev stopenj je torej binomska:



Obe lastnosti modela $\mathcal{G}(n,p)$, namreč da povezave grafa $G \in \mathcal{G}(n,p)$ nastopijo z enakomerno verjetnostjo in neodvisno druga od druge, sta posledica preveč preproste konstrukcije in sta neprimerni za modeliranje mnogih naključnih grafov iz vsakdanjega življenja.

Model naključnega grafa tipa **Barabási-Albert**, oz. **BA-model**, poznan tudi kot '**power law**' model, prihaja iz vsakdanjih situacij. Motivacija je naslednja. Večina omrežij iz vsakdanjega življenja (npr. ljudje + poznanstva; www strani + linki; molekule + reakcije; članki + citati; itd) ima lastnost, da je njihova porazdelitev stopenj potenčna (tj. 'power law´ oz. 'scale free´). Razlog za to je 'preferrential attachment' (npr. ko človek pride v novo družbo bo najverjetneje spoznal ljudi ki imajo veliko poznanstev; nova internetna stran bo najverjetneje imela povezave do priljubljenih strani

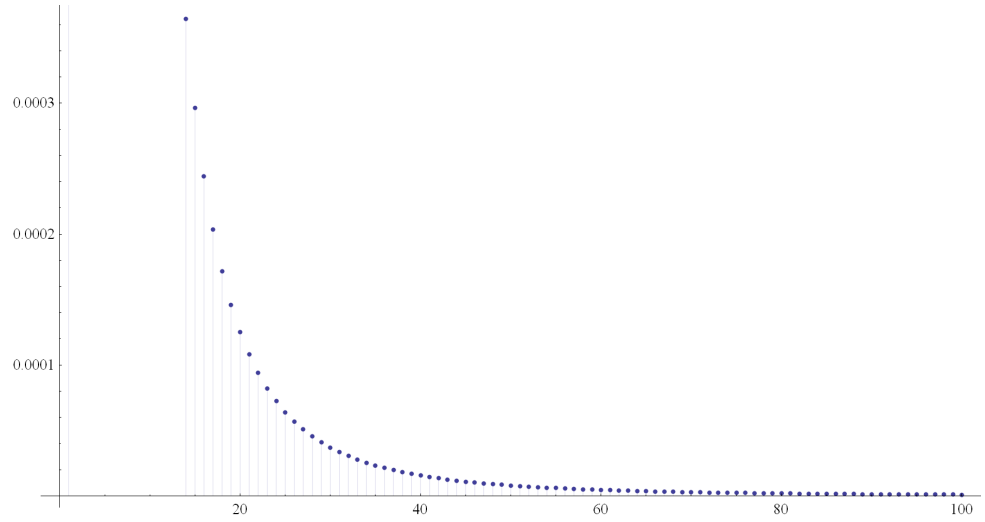
kot so google in wikipedia; molekula ki smo jo dodali v spojino bo najverjetneje reagirala z visoko reaktivnimi molekulami; nov članek se bo najverjetneje skliceval na članke z veliko citati; itd). Zato je naš cilj imeti grafovski model, katerega porazdelitev stopenj vozlišč je potenčna ('power law´), tj. oblike $ck^{-\delta}$ za $c, \delta > 0$.

Tak model konstuiramo na naslednji induktiven način: na začetku imamo graf z vsaj dvemi vozlišči in brez izoliranih točk (sicer dobimo na koncu nepovezan graf); v vsakem koraku dodamo novo vozlišče v_n, in ga povežemo z vsakim v_i ($i = 1, \ldots, n–1$) z verjetnostjo

$$\frac{d_i}{\sum_{j=1}^{n-1} d_j},$$

kjer d_i označuje stopnjo vozlišča v_i v trenutnem grafu. To pomeni, da tista vozlišča ki že imajo veliko sosedov, imajo dosti večjo verjetnost da dobijo spet novega soseda, vozlišča z majhno stopnjo pa zelo verjetno ostanejo še naprej 'osamljena'. Takemu načinu dodajanja novih vozlišč pravimo '**preferrential attachment**', njegovemu efektu pa 'the rich get richer' pojav. Avtorja sta ga uvedla leta 1999 [30], čeprav so ideje bile prisotne že leta 1925 (Yule, [44]), 1955 (Simon, [40]), 1976 (Price, [33]).

Da se pokazati, da je pričakovana porazdelitev stopenj vozlišč tega grafovskega modela enaka k^{-3}, torej potenčna:



Povprečna razdalja (dolžina najkrajše poti) med pari vozlišč je $\frac{\ln n}{lnlnn}$.

Model naključnega grafa tipa **Watts-Strogatz**, oz. **WS-model**, poznan tudi kot '**small world**' model, ki sta ga avtorja uvedla leta 1998, ima motivacijo v naslednjem primeru. Eksperimentalni poskus je pokazal, da za dva naključno izbrana prebivalca x in y v ZDA z veliko verjetnostjo obstajajo štirje prebivalci, tako da vseh šest skupaj tvori verigo poznanstev, ki se žacne v x in konca v y. Z drugimi besedami, povprečna razdalja v grafu ljudi in poznanstev je približno 6, torej majhna, kljub temu pa je celotno število vozlišč zelo veliko in povprečna stopnja majhna. Podobne lastnosti ima graf vseh letališc na svetu in direktnih letov, itd. Cilj je torej imeti grafovski model, katerega povprečna razdalja med pari vozlišč je 'majhna', kljub temu pa je tudi povprečna stopnja vozlišč 'majhna'. Natančneje, za grafovski model na n vozliščih mora povprečna razdalja med vozlišči biti proporcionalna $\ln n$.

Tak model konstuiramo na naslednji induktiven način:

- Input: n (št. vozlišč), $\overline{d}$ (povprečna stopnja, sodo naravno št.), $\beta \in [0,1]$, tako da je $n \gg \overline{d} \gg \ln n \gg 1$.
- (deterministično korak) Konstruiramo 'regular ring lattice´, tj. graf katerega vozlišča so $v_0, \ldots, vn-1$, $v_0, \ldots, v_{n-1}$, in med v_i ter v_j je povezava natanko tedaj ko je $0 < |i-j| < \frac{\overline{d}}{2}$.
- (stohastičen korak) Iterativno za vsako obstoječe vozlišče v_i $(i = 1, \ldots, n)$ vsako povezavo $\{v_i, v_j\}$ $(i < j)$ z verjetnostjo β naključno prevežemo, tj. naključno z enakomerno verjetnostjo izberemo tak $k \in \{1, \ldots, i-1, i+1, \ldots, n\}$, da $\{v_i, v_k\}$ ni povezava v trenutnem grafu, ter povezavo $\{v_i, v_j\}$ nadomestimo z $\{v_i, v_k\}$.
- Output: enostaven graf z n vozlišči, $\frac{n\overline{d}}{2}$ povezavami, in povprečno stopnjo d.

13.2 Nove Grafovske Statistike

Za dani graf bi radi izračunali določene lastnosti (tj. statistike grafa), ki bi čim več povedale o njegovi strukturi. Znano je (glej [30, str. 44, Izrek 1.8]), da je do izomorfizma natančno vsak graf enolično določen z njegovih spektrom in lastnimi vektorji. Zato se zdi smiselno, da za statistiko izberemo le določene podatke iz spektra in lastnih vektorjev, namreč tiste ki čim več povedo o grafu.

Ker je matrika sosednosti A grafa G realna simetrična, se jo da diagonalizirati (z realnimi lastnimi vrednostmi). Če so $w_1, \ldots, w_n$ normirani lastni vektorji matrike A, in če so $w_{i,1}, \ldots, w_{i,n}$ komponente vektorja w_i, potem definiramo **inverse participation ratio** [37] kot

$$I_i = I_i(G) := \sum_{j=1}^{n} w_{i,j}^4.$$

Spomnimo se, da je $w \in \mathbb{R}^n \setminus \{0\}$ lastni vektor matrike A natanko tedaj ko obstaja $\lambda \in \mathbb{R}$, da je $Aw = \lambda w$, natanko tedaj ko za preslikavo $w\colon V \to \mathbb{R}, v_i \mapsto w_i$ (ki ji pravimo 'utežitev vozlišč´) in za za vsak $i = 1, \ldots, n$ velja $\lambda w(v_i) = \sum_{v_j \in N(v_i)} w(v_j)$, kjer $N(v_i)$ označuje okolico od v_i, tj. sosednja vozlišča v grafu. Zadnja ekvivalenca sledi iz definicije matrike sosednosti in dejstva, da je i-ta komponenta od Aw enaka $\sum_{j=1}^{n} A_{i,j} w_j = \sum_{v_j \in N(v_i)} w_j$. Torej je lastni vektor natanko neničelna utežitev vozlišč z lastnostjo $\lambda w(v_i) = \sum_{v_j \in N(v_i)} w(v_j)$ $(*)$ za nek $\lambda \in \mathbb{R}$ in vsak $v_j \in V$. Ker je vsak w_i normiran, je $\sum_{j=1}^{n} w_{i,j}^2 = 1$, in zato velja $\sum_{j=1}^{n} w_{i,j}^2 \leq 1$. Spomnimo se, da je za $a_1, \ldots, a_n \in \mathbb{R}$ in $p \in \mathbb{R} \setminus \{0\}$ posplošena srednja vrednost definirana kot $M_p(a_1, \ldots, a_n) := \left(\frac{1}{n} \sum_{i=1}^{n} a_i^p\right)^{\frac{1}{p}}$, in da zanjo velja implikacija $p < q \Rightarrow M_p(a_1, \ldots, a_n) \leq M_q(a_1, \ldots, a_n)$. Za $p = 2$ in $q = 4$ dobimo $\left(\frac{\sum_{j=1}^{n} a_j^2}{n}\right)^{\frac{1}{2}} \leq \left(\frac{\sum_{j=1}^{n} a_j^4}{n}\right)^{\frac{1}{4}}$, torej $\left(\frac{\sum_{j=1}^{n} a_j^2}{n}\right)^2 \leq \frac{\sum_{j=1}^{n} a_j^4}{n}$, in če to uporabimo na $a_j = w_{i,j}$, dobimo $\frac{1}{n^2} \leq \frac{\sum_{j=1}^{n} w_{i,j}^4}{n}$, tj. $\frac{1}{n} \leq \sum_{j=1}^{n} w_{i,j}^4$. Dokazali smo, da je

$$I_i(G) \in [\tfrac{1}{n}, 1].$$

Obe skrajni vrednosti sta lahko doseženi: vektor $w_i = (\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}})$ je utež na vozliščih polnega grafa K_n z lastnostjo $(*)$ za $\lambda = 1$, in nam da $I_i = \frac{1}{n}$; vektor $w_i = (1, 0, \ldots, 0)$ je utež

na vozliščih praznega grafa $\overline{K_n}$ z lastnostjo $(*)$ za $\lambda=0$), in nam da $I_i=1$. Vidimo torej, da za poljuben graf G število $I_i(G)$ meri koliko so uteži normiranega lastnega vektorja w_i lokalizirane (ali so vrednosti enakomerno razporejene po vozliščih, ali pa zgoščene v enem vozlišču).

Če lastne vrednosti matrike A uredimo po velikosti kot $\lambda_1 \leq \ldots \leq \lambda_n$, potem definiramo **odmik največje lastne vrednosti** kot

$$R(G) := \frac{\lambda_n - \lambda_{n-1}}{\lambda_{n-1} - \lambda_1}.$$

Ta količina torej meri koliko je največja lastna vrednost odmaknjena od preostanka spektra, relativno glede na širino preostanka spektra. V [37] so avtorji opazili korelacijo med $R(G)$ in kromatičnim številom $\chi(G)$. Po izreku [31] velja $1-\frac{\lambda_n}{\lambda_1} \leq \chi(G)$, in zato je

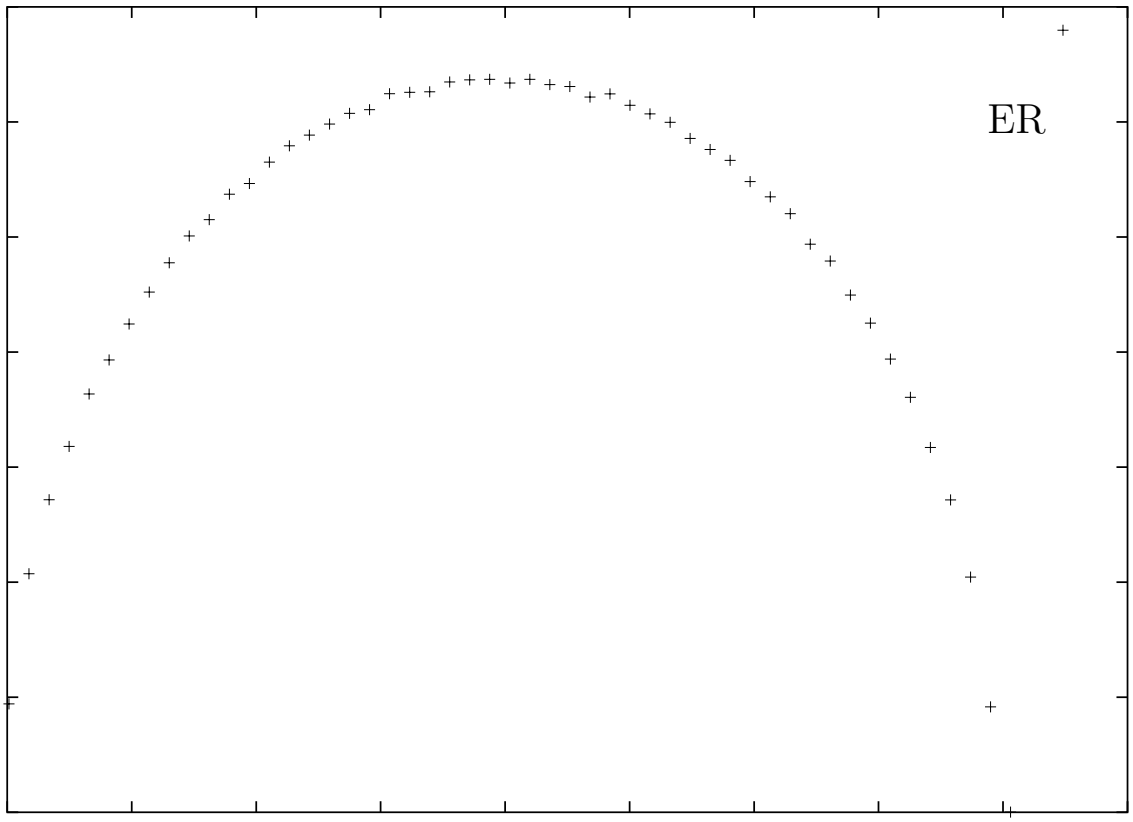
$$\frac{\chi(G)}{2} - 1 \geq \frac{-\lambda_n}{2\lambda_1} - \frac{1}{2} = \frac{-\lambda_n - \lambda_1}{2\lambda_1} =$$
$$\frac{-\lambda_n + \lambda_{n-1} - (\lambda_{n-1} + \lambda_1)}{\lambda_1 - \lambda_{n-1} + (\lambda_{n-1} + \lambda_1)} = \frac{\lambda_n - \lambda_{n-1} + (\lambda_{n-1} + \lambda_1)}{\lambda_{n-1} - \lambda_1 - (\lambda_{n-1} + \lambda_1)}.$$

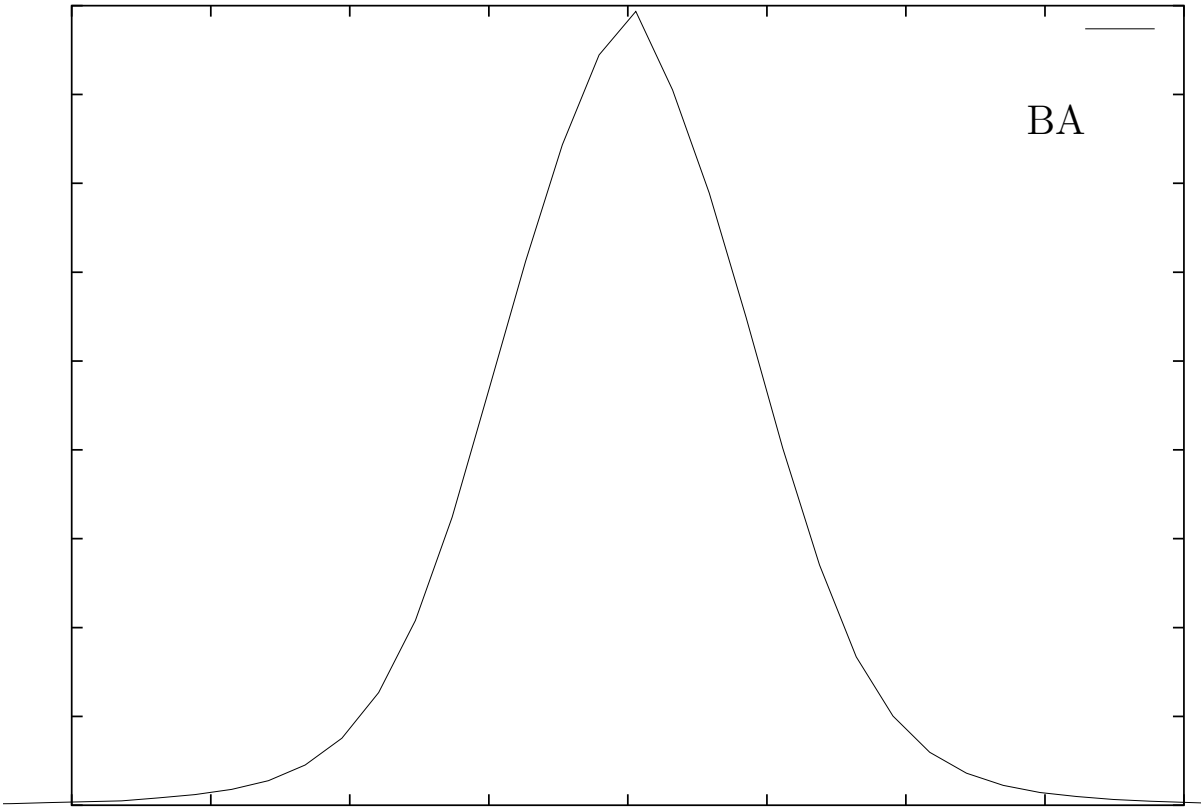
Če je torej $\lambda_{n-1} + \lambda_1 = 0$, je $\chi(G) \geq 2R(G)+2$. Od tod sklepamo, da za majhne $\lambda_{n-1} + \lambda_1$ (tj. ko je preostanek spektra simetričen glede na 0) ta ocena še vedno približno velja.

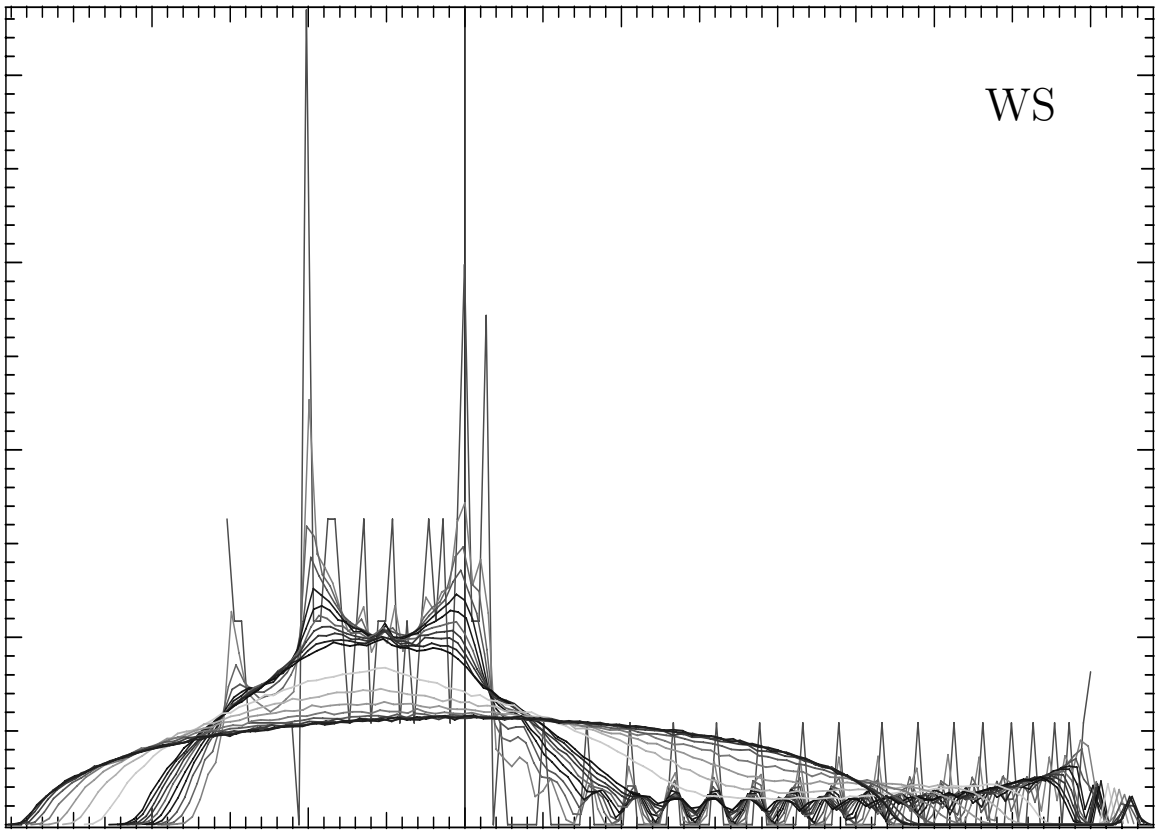
13.3 Spektralne Lastnosti Naključnih Grafov

Za dani graf bi radi ugotovili njegove kvalitativne lastnosti. Natančneje, radi bi določili katera izmed treh skupin naključnih grafov (ER-, BA-, in WS-model) je najbolj primerna za opis lastnosti grafa. En način prepoznavanja grafa je, da izračunamo njegov spekter, narišemo histogram lastnih vrednosti (graf, katerega x-os zaznamuje velikost lastnih vrednosti, torej cela realna os, y-os pa zaznamuje pogostost pojavitve lastne vrednosti, torej naravna števila). V ta namen moramo prvo preučiti spektre treh modelov naključnih grafov.

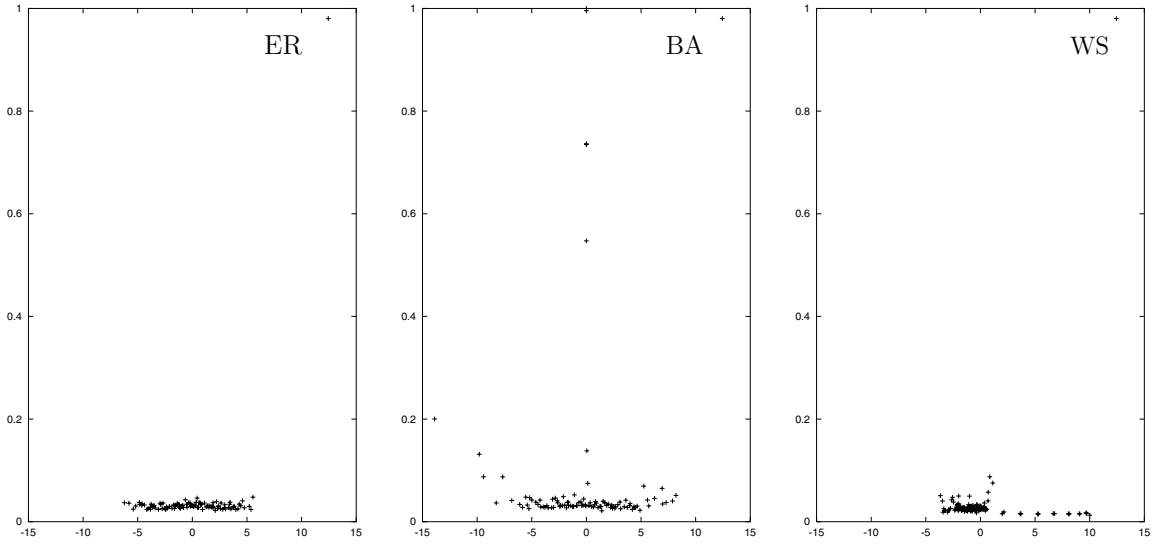
Napisan je bil kratek program v C-ju, ki je iz vsake izmed treh družin generiral 100 grafov na 2000 vozliščih. Za vsakega izmed njih je izračunal spekter ter unijo teh spektrov podal v obliki histograma - glej spodnje tri slike. Pri ER-modelu je oblika histograma polkrožna. To je značilna lastnost ER-grafov in je posledica polkrožnostnega zakona iz teorije stohastičnih matrik. Prvotno je ta zakon odkril Wigner [42, 43], kasneje pa ga je izostrila kopica raziskovalcev [29, 38, 39]. Histogram spektra BA-modela ima značilno trikotno obliko, z velikim deležem relativno majhnih lastnih vrednosti. Histogram spektra WS-modela pa je tipično bistveno bolj divji, z večimi vrhovi in asimetrično obliko. Tu je potrebno povedati da zadnji histogram prikazuje več WS-modelov, namreč pri različnih vrednostih β (verjetnost prevezovanja povezav v konstrukciji). Eksperimentalno se izkaže, da je oblika histograma vsakega izmed treh modelov neodvisna od velikosti modela in parametrov; to so torej kvalitativne invariante.


ER


BA



Na koncu omenimo še kako izgledajo histogrami od ‘inverse participation ratios´ vseh treh modelov na 100 vozliščih (x-os so vrednosti lastnih vrednosti, y-os pa je $I_i(G)$):



Vidimo, da se $I_i(G)$ res giblje med $\frac{1}{100}$ in 1. Grafe, ki imajo lastnosti ER-tipa lahko hitro prepoznamo: vrednosti $I_i(G)$ so natlačene na enem majhnem območju, relativno blizu $\frac{1}{n}$. Tipična lastnost BA-modela so visoki $I_i(G)$. Za WS-model pa je značilna zelo asimetrična oblika.

Poglavje 14

Pajek

Jaka Kranjc, Jernej Zupančič

14.1 Uvod

Pri svojem delu se večkrat srečamo z obsežnimi omrežji, kjer gre število točk in povezav v tisoče ali celo v milijone. Primeri takšnih omrežij so: socialna omrežja, rodovniki, omrežja dobljena iz slovarjev in drugih besedil, računalniška omrežja, omrežja delov spleta, omrežja soavtorstev, omrežja sklicevanj, transportna ter cestna omrežja, vodovodna in električna omrežja, diagrami poteka programskih sistemov, organske molekule (npr. DNK), omrežja razširjanja bolezni, omrežja lastništva delnic ...

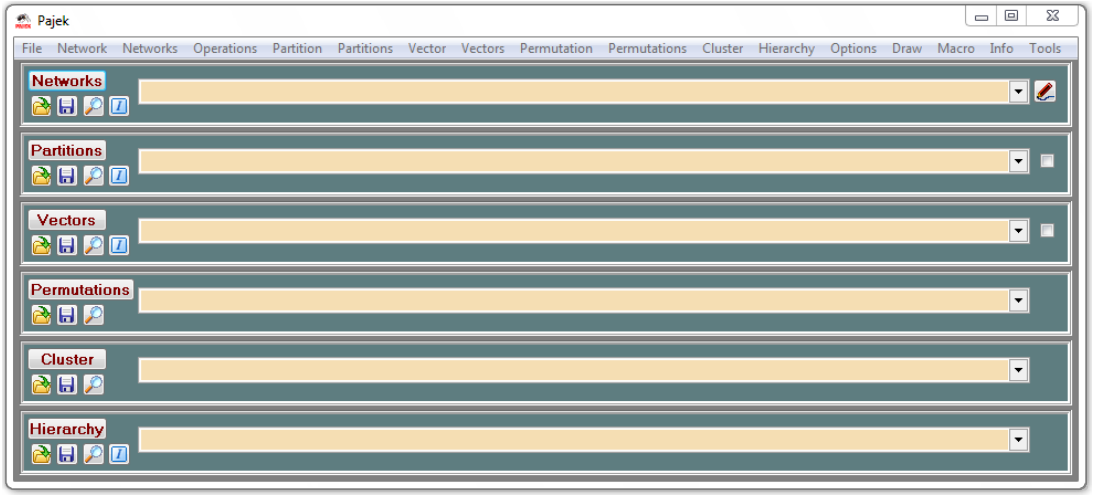
Ker obvladovanje velikih omrežij predstavlja tako časovno kot tudi prostorsko zahteven problem, sta se Vladimir Batagelj s Fakultete za matematiko in fiziko in Andrej Mrvar s Fakultete za družbene vede leta 1996 odločila za razvoj programskega paketa za Windows sisteme namenjenega analizi in prikazu velikih omrežij. Programski paket, poimenovan Pajek, vsebuje zelo učinkovite (do superlinerane) algoritme in je trenutno prosto dostopen na spletni strani `http://pajek.imfm.si/`.

Glavni cilji pri zasnovi programa Pajek so:

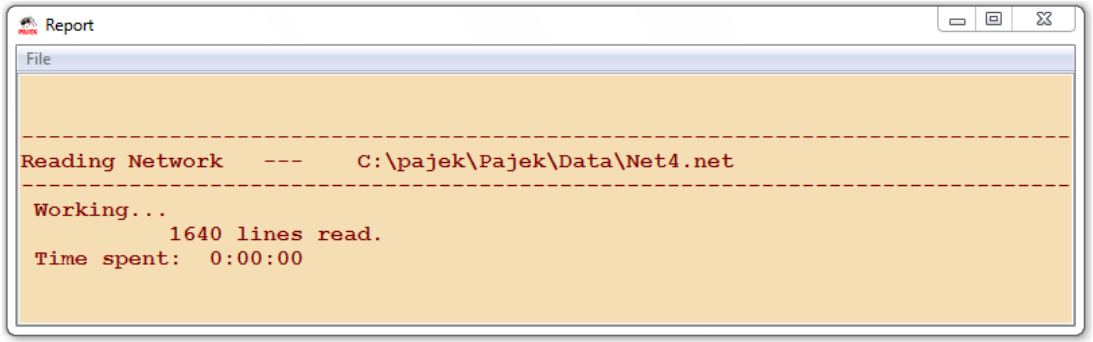
- podpreti abstrakcijo z (rekurzivno) razčlenitvijo velikega omrežja na več manjših omrežij, ki jih nato lahko analiziramo z uporabo običajnih metod,
- ponuditi uporabniku močna orodja za grafični prikaz omrežij in
- vgraditi večje število učinkovitih algoritmov za analizo obsežnih omrežij.

Ob programu Pajek se je v zadnjem času začel razvijati tudi program Pajek-XXL, ki na istem omrežju porabi tudi do trikrat manj pomnilnika kot Pajek, hkrati pa se pomnilniško zathevnejše operacije izvajajo hitreje.

Pajek nam torej omogoča iskanje gruč (npr. komponente, okolice „pomembnejših“ vozlišč, jedra itd.), izločanje vozlišč, pripadajočih isti gruči, in prikaz le teh. Po možnosti z deli okolice ali z relacijami do preostalih gruč. Poleg klasičnih omrežij so podprta tudi časovno spremenljiva omrežja, hipergrafi in dvovrstna omrežja.



Slika 14.1: Glavno okno programa Pajek



Slika 14.2: Okno poročil

Poleg glavnega okna programa Pajek (slika 14.1) je izredno pomembno okno poročil Report (slika 14.2), ki se ga odpre, če že ni odprt, z *File* → *Show Report Window*, kamor se izpisujejo vsi podatki o podatkovnih strukturah in operacijah.

14.2 Podatkovne strukture

Program Pajek podpira šest podatkovnih struktur. Te so

1. omrežje (Network) – sestoji iz točk in povezav in predstavlja osrednjo podatkovno strukturo,
2. razbitje (Partition) – vsakemu vozlišču priredi razred,
3. vektor (Vector) – vsakemu vozlišču priredi neko (realno) številsko vrednost,
4. permutacija (Permutation) – prerazporeditve vozlišč,
5. gruča (Cluster) – podmnožica množice vozlišč in
6. hierarhija (Hierarchy) – razvrstitev po položaju, funkcijah, pomembnosti.

Ko je podatkovna struktura naložena v program, se z dvoklikom nanjo prikaže njena vsebina. Omenimo še, da so operacije na podatkovnih strukturah v meniju glavnega okna programa Pajek razporejene glede na podatkovno strukturo, ki jo uporabljajo za vhoden podatek.

14.3 Omrežje

Omrežje se v program Pajek lahko vnese na vsaj tri načine:

- z uvozom datoteke formata
  - Pajek networks (`.net`),
  - Pajek matrices (`.mat`),
  - Vega graphs (`.vgr`),
  - GEDCOM files (`.ged`),
  - UCINET DL files (`.dat`),
  - Ball and Stick files (`.bs`),
  - Mac Molecules (`.mac`) in
  - MDL MOL (`.mol`),
- z grafičnim ustvarjanjem v samem programu ali
- z generiranjem naključnega omrežja v samem programu.

V nadaljevanju bomo spoznali enega od formatov, ki sta bila razvita skupaj s programom Pajek.

```
1  *Vertices 5
2  1 "x"
3  2 "z"
4  3 "b"
5  4 "r"
6  5
7  *Arcslist
8  1 2 4
9  2 3
10 3 1 4
11 4 5
12 *Edgeslist
13 1 5
```

Primer 14.1: Datoteka za omrežje 14.3

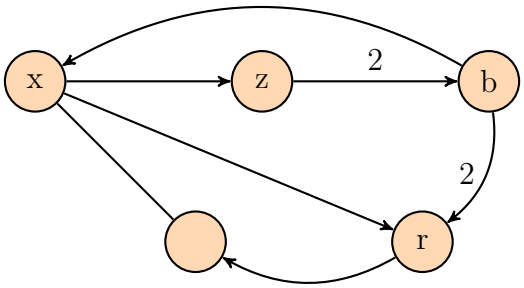
14.3.1 Zapis omrežja v formatu Pajek networks

Ker je format Pajek networks zapisan kot besedilna datoteka s privzeto končnico `.net`, jo je enostavno urejati z enakimi programi, kot se ureja LaTeX kodo (npr. Notepad++, Emacs, Vim ipd.).

V prvi vrstici datoteke Pajek networks je za ukazom `*Vertices` navedeno skupno število vseh vozlišč, ki so nato opisana. Zaporedni številki vozlišča nato sledi njena oznaka, ki jo navedemo v narekovajih. Če je oznaka vozlišča izpuščena, mu Pajek dodeli oznako sestavljeno iz črke `v` in zaporedenega števila vozlišča npr. `v1`, `v2` ...

Po navedbi vozlišč so v področju `*Arcslist` zapisane usmerjene povezave, kjer sta v vsaki vrstici navedena vsaj dve števili vozlišč. Prvo število v vrstici označuje začetno vozlišče povezave in vsa naslednja konce. Povezave, kjer usmerjenost ni pomembna, so zapisane po istem kopitu v `*Edgeslist`. Na koncu datoteke ne sme biti praznih vrstic.

Primer 14.1 prikazuje, kako se najenostavnejše zapiše omrežje na sliki 14.3 v format Pajek networks.



Slika 14.3: Omrežje na petih vozliščih

Ker zapis v primeru 14.1 ne omogoča podajanja vrednosti na povezavah, se lahko namesto `*Arcslist` uporabi `*Arcs`, kot je v primeru 14.2, kjer je v vsaki vrstici navedena samo po ena usmerjena povezava.

```
*Vertices 5
1 "x"
2 "z"
3 "b"
4 "r"
5
*Arcs
1 2 1
1 4 1
2 3 2
3 1 1
3 4 2
4 5 1
*Edges
1 5 1
```

Primer 14.2: Datoteka z `*Arcs` za omrežje 14.3

```
*Vertices 5
1 "x"
2 "z"
3 "b"
4 "r"
5
*Matrix
0 1 0 1 1
0 0 2 0 0
1 0 0 2 0
0 0 0 0 1
1 0 0 0 0
```

Primer 14.3: Datoteka z `*Matrix` za omrežje 14.3

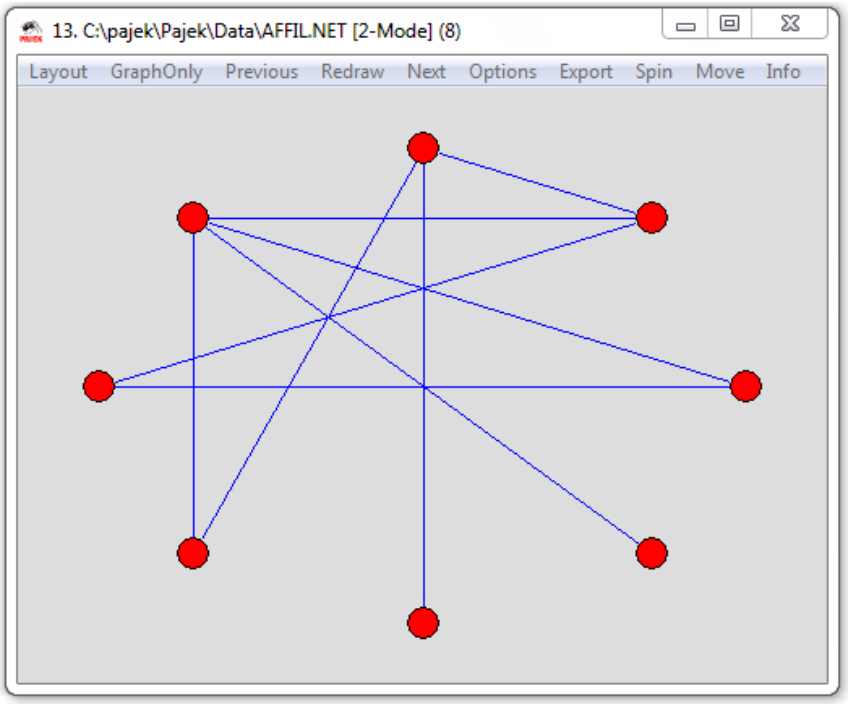
Tokrat prvo število označuje začetno vozlišče, drugo končno vozlišče in tretje utež (tudi vrednost) povezave. Če se utež izpusti, ji Pajek dodeli privzeto vrednost 1. Neusmerjene povezave so tokrat zapisane pod ukazom `*Edges` in za njih veljajo enaki dogovori kot pri `*Arcs`. Omrežje pa se lahko poda tudi z matriko sosednosti `*Matrix` kot v primeru 14.3.

Ker Pajek pri matriki sosednosti za podano neusmerjeno povezavo ustvari dve nasprotno usmerjeni povezavi, se vsaki takšni dve lahko združita v neusmerjeno tako, da se v meniju izbere *Network* → *Create New Network* → *Transform* → *Arcs to Edges* → *Bidirected Only*.

14.3.2 Grafični prikaz omrežja

Predenj se lotimo grafičnega ustvarjanja omrežja, si je potrebno pogledati, kako se omrežje izriše.

Omrežje se izriše z izbiro *Draw* → *Network* ali s pritiskom tipk `Ctrl+G`. Slednje v novem oknu (slika 14.4) odpre predstavitev omrežja z risbo.



Slika 14.4: Okno grafičnega prikaza omrežja

Na risalnem področju je mogoče premikati vozlišča in s sekundarnim klikom in vlečenjem izbrati področje za povečavo. Do začetnega oziroma osnovnega pogleda se vrne z uporabo izbire *Redraw*. Pritisk na tipke `x`, `y` in `z` zavrti omrežje okoli osnovnih treh osi, uporaba tipk `X`, `Y` in `Z` pa zavrti omrežje v drugo smer. Poljubno os se izbere z *Spin* → *Normal*, okoli katere se nato vrti s `s` in `S`.

Uporabne bližnjice:

- `Ctrl+L` prikaže oznake točk,
- `Ctrl+N` prikaže zaporedna števila vozlišč,
- `Ctrl+D` skrije oznake vozlišč,
- `Ctrl+B` prikaže zaporedna števila povezav,
- `Ctrl+V` prikaže vrednosti povezav.

Meni *Info* nam postreže s podatki o lastnosti risbe, ki jih tudi izpisuje v okno poročil 14.2. Te so

- najbližje narisani vozlišči (vozlišči se obarvata rumeno),
- najmanjši kot med narisanimi povezavami (vozlišča, pripadajoča povezavam, ki tvorijo kot, se obarvajo zeleno),
- najdaljša narisana povezava (krajiščni vozlišči se obarvata modro),
- najkrajša narisana povezava (krajiščni vozlišči se obarvata rdeče),

- narisana križanja povezav (krajiščna vozlišča vseh sekajočih se povezav se obarvajo svetlo vijolično) in
- najbližja točka do neke druge povezave (vozlišče se obarva belo).

Prikazi omrežij

Ker se pravo risbo omrežja dobi šele s slikovno predstavitvijo, je v programu Pajek implementiranih več algoritmov za samodejen izris omrežja.

Med zanimivejše algoritme sodijo t. i. energijska risanja, ki temeljijo na fizikalnih lastnostih, kjer so povezave predstavljene z vzmetmi in se išče minimalna skupna energija. V program Pajek sta vgrajena dva taka algoritma:

- *Layout* → *Energy* → *Fruchterman Reingold*, ki omogoča risanje v ravnini ali v prostoru z izbiro odbojnega faktorja, in
- *Layout* → *Energy* → *Kamada-Kawai*, ki je nekoliko počasnejši, a vsebuje ukaz *Separate Components*, ki je primeren za risanje nepovezanih omrežij. Primer risbe po metodi *Free* je na sliki 14.5.

Poleg energijskih risanj Pajek omogoča tudi risanje s pomočjo lastnih vrednosti. Za prostorsko predstavo se risbo lahko zavrti s *Spin* → *Spin Around*.

Izvoz grafa

Dobljene ravninske ali prostorske risbe omrežja je mogoče izvoziti v formatu Encapsulated PostScript (`.eps`) in ga nato vključiti v LaTeX, kar se doseže z izbiro *Export* → *2D* → *EPS/PS*. Sliki 14.6 in 14.9 sta risbi ustvarjeni s programom Pajek.

Pajek poleg formata `.eps` podpira še Scalable Vector Graphics (`.svg`), Joint Photographic Experts Group (`.jpg`), bitne slike (`.bmp`), Extensible 3D Graphics (`.x3d`), Virtual Reality Modeling Language (`.vrml`) in MDL MOL (`.mol`).

14.3.3 Grafično (interaktivno) ustvarjanje omrežja

Ideja je, da se v nepovezano omrežje z nekaj vozlišči dodaja vozlišča in (usmerjene) povezave.

Nepovezano omrežje z N točkami (brez povezav) se ustvari z *Network* → *Create New Network* → *Empty Network*, kjer se odgovori z N (število željenih vozlišč). Podobno se lahko doseže z *Network* → *Create Random Network* → *Total No. of Arcs*, kjer se na prvo vprašanje po številu vozlišč odgovori z N, na drugo vprašanje po številu povezav pa z 0, kar pomeni, da bo Pajek ustvaril naključno omrežje brez povezav. Dodajanje novih vozlišč se doseže z *Network* → *Create New Network* → *Transform* → *Add* → *Vertices*.

Nato se prestavimo v, zdaj že znani, grafični način, kjer z miško po potrebi prenesemo točke na željena mesta. Sekundarni miškin klik na izbrano vozlišče x odpre novo pogovorno okno s seznamom povezav, katerih krajišče je izbrano vozlišče x.

Dvoklik na `New line` in vpis negativno predznačenega zaporednega števila vozlišča doda povezavo iz izbranega vozlišča x; vpis pozitivno predznačenega števila pa pomeni, da je izbrano vozlišče x končno. Če se predznak izpusti, potem usmerjenost nove povezave ni določena.

Vrednost na povezavi se določi s sekundarnim klikom na željeno označeno povezavo na seznamu povezav. Povezavo se odstrani z dvoklikom na željeno vrstico – povezavo na seznamu povezav.

14.3.4 Generiranje naključnega omrežja

Poleg zgoraj omenjenih dveh načinom Pajek podpira tudi generiranje različnih modelov naključnih omrežij, kot so slučajni ER grafi, brezlestvična omrežja in omrežja z lastnostjo malega sveta.

Do ukaza za generiranje dostopamo z *Network* → *Create Random Network*, kjer se nato izbere željen model naključnega omrežja. Pajek nas interaktivno vodi pri podajanju parametrov, ki so odvisni od izbranega modela. Trenutna različica Pajek 3.03 podpira izgradnjo naslednjih naključnih omrežij.

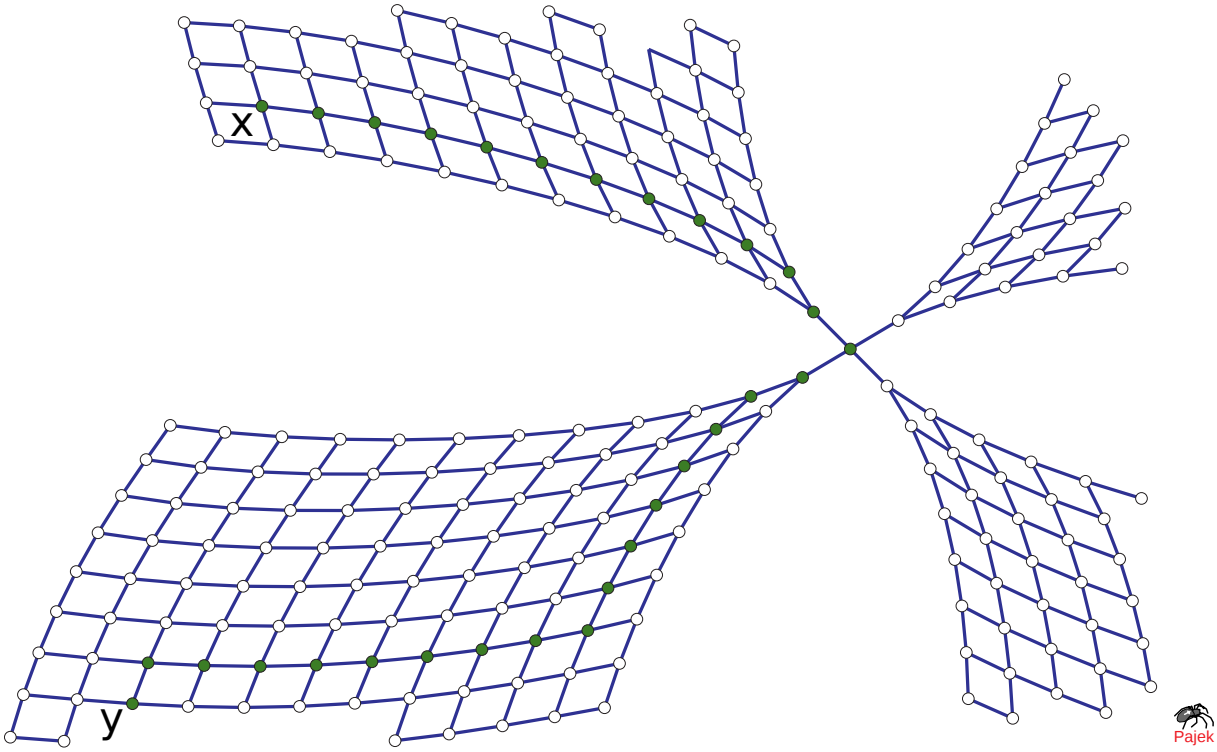
- *Total No. of Arcs* - Generira naključno usmerjeno omrežje z izbranim številom vozlišč in povezav.
- *Vertices Output Degree* - Generira naključno omrežje z izbranim številom vozlišč, kjer ima vsako vozlišče naključno izhodno stopnjo iz vnaprej določenega intervala.
- *Bernoulli/Poisson* - Generira usmerjen, neusmerjen, acikličen ali dvodelen graf po Bernoulli/Poisson modelu, kjer vsako povezavo med vozliščema ustvarimo z verjetnostjo p. Ta parameter je za velika omrežja ponavadi zelo majhno število, zato Pajek uporablja uporabniku bolj prijazno povprečno stopnjo $\overline{d}$, ki pa je v povezavi s p, saj $\overline{d} = \frac{1}{n}\sum_{v\in V}\deg(v) = \frac{2m}{n}$ in $m = pM$, kjer je n število vozlišč, M maksimalno število povezav v družini grafov in m pričakovano število povezav v željenem grafu.
- *Scale Free* - Generira brezlestvično usmerjeno, neusmerjeno ali aciklično omrežje. Postopek temelji na izboljšanem modelu za generiranje brezlestvičnih omrežij, predstavljenem v [45]. Omrežju v vsakem koraku dodamo novo vozlišče in k novih povezav. Krajišča povezav so naključno izbrana z verjetnostjo

  $$\Pr(v) = \alpha\frac{\text{indeg}(v)}{|E|} + \beta\frac{\text{outdeg}(v)}{|E|} + \gamma\frac{1}{|V|},$$

  kjer je $\alpha + \beta + \gamma = 1$, E je množica trenutnih povezav v omrežju, V pa množica trenutnih vozlišč. Pajek nas povpraša po parametrih α in β, γ pa določi iz enačbe.
- *Small World* - Generira omrežje z lastnostjo malega sveta, opisano v [46], po modelu Wattsa in Strogatza. V dobljenem omrežju med poljubnima vozliščema pridemo v le nekaj korakih (lastnost majhnega sveta) in število ciklov dolžine tri je zelo veliko (velik koeficient gručavosti).
- *Extended Model* - Generira naključno omrežje po razširjenem modelu, ki sta ga predstavila Albert in Barabasi v [47].

14.3.5 Operacije

Operacije na enem omrežju se nahajajo v meniju *Network*, medtem ko se operacije na več omrežjih nahajajo v meniju *Networks*. Če operacije poleg omrežja potrebujejo še podatkovno strukturo drugega tipa, so dosegljive pod *Operations*. Podroben opis vseh operacij se nahaja v uradni dokumentaciji [48].



Slika 14.5: Omrežje, kjer so vozlišča na najkrajši poti od x do y obarvana zeleno

Najkrajše poti

Definicija 14.1 Zaporedje vozlišč omrežja $(v_1, v_2, \ldots, v_k)$ se imenuje pot, če iz vsakega vozlišča v_i obstaja neposredna usmerjena povezava v naslednje vozlišče v_{i+1}, pri čemer se neusmerjeno povezavo gleda kot dve usmerjeni. Razen robnih se vozlišča ne smejo ponavljati. Pri tem je $k - 1$ (število povezav na poti) dolžina poti.

Med dvema vozliščema lahko obstaja več poti. Najzanimivejša med njimi je najkrajša pot glede na število povezav ali, če imamo na povezavah podane nenegativne vrednosti, pot z najmanjšo vsoto vrednosti na njenih povezavah oziroma najcenejša pot.

Metoda za določitev najkrajše poti glede na dolžino ali najcenejše poti glede na vrednost povezav med dvema vozliščema se najde pod *Network* → *Create New Network* → *SubNetwork with Paths* → *One Shortest Path between Two Vertices*, kamor se vneseta oznaki ali zaporedni številki vozlišč. Na vprašanje „Forget values on lines?“ Se odgovori z Yes, če se gleda dolžino, oziroma z No, če se gleda vrednosti povezav. Na drugo vprašanje „Identify vertices in source network?“ se odgovori z No, če se želi nov podgraf, oziroma z Yes, če se želi razbitje omrežja. Slednje se lahko grafično predstavi z *Draw* → *Network + First Partition* oziroma z `Ctrl+P`, ki označi točke na poti, kar je kot primer prikazano na sliki 14.5.

Vse najkrajše poti med danima vozliščema se poišče z *Network* → *Create New Network* → *SubNetwork with Paths* → *All Shortest Paths between Two Vertices*, kjer so vprašanja takšna kot zgoraj. Če se pri iskanju poti ne želi upoštevati usmerjenosti povezav, se lahko vse usmerjene povezave z *Network* → *Create New Network* → *Transform* → *Arcs to Edges* → *All* spremeni v neusmerjene.

Merjenje dolžin

Metoda za izračun premera (diametra) omrežja oziroma najdaljše dolžine najkrajše poti med vozlišči se nahaja pod *Network* → *Create New Network* → *SubNetwork with Paths* → *Info on Diameter*. Poleg rezultata, ki se izpiše v okno Report, Pajek označi tudi vozlišči, ki sta „najbolj narazen“.

Za majhna omrežja (algoritem je časovno zahteven) se lahko poišče tudi vse najkrajše poti glede na dolžino med vsemi pari vozlišč in število najkrajših poti, ki povezujejo vozlišča. Uporabi se *Network* → *Create New Network* → *SubNetwork with Paths* → *Geodesics Matrices**.

Porazdelitev dolžin najkrajših poti, povprečno razdaljo in vozlišči, ki sta med sabo najbolj oddaljeni, dobimo z *Network* → *Create Vector* → *Distribution of Distances**

14.4 Razbitje

Pri analizi velikih omrežij se pogosto zgodi, da ne želimo opazovati vsakega vozlišča posebej, ampak si hočemo ustvariti neko globalno sliko. Pri tem želimo podobna vozlišča združiti v razrede, kar nam omogoča preučevanje odnosov med razredi ali pa relacij v samo enem razredu. Vsi razredi skupaj sestavljajo razbitje.

Pri delu z razbitjem v programu Pajek se vsakemu vozlišču priredi celo število, kjer to število ponazarja razred, kamor vozlišče pripada. Razbitje si lahko definiramo sami ali pa nam pri tem pomaga Pajek z nekaterimi, že vgrajeni, operacijami, ki so dostopne pod *Network* → *Create Partition*.

14.4.1 Operacije

Komponente omrežja

Definicija 14.2 Množica vozlišč je krepko povezana komponenta, če iz vsakega vozlišča te množice obstaja usmerjena pot v vsako drugo vozlišče te množice. Če smer poti ni pomembna, se takšna množica vozlišč imenuje šibko povezana komponenta.

V primeru neusmerjenega omrežja so krepke komponente enake šibkim, zato jih poimenujemo samo komponente. V programu Pajek se krepke komponente izračuna z ukazom *Network* → *Create Partition* → *Components* → *Strong*, medtem ko se šibke dobi preko *Network* → *Create Partition* → *Components* → *Weak*. Rezultat operacij je razbitje, ki vozliščem, pripadajočim isti komponenti, priredi isto število.

Jedra

Definicija 14.3 Množica vozlišč H_k je k-jedro, če je vsako vozlišče množice povezano z vsaj k vozlišči te množice.

Poseben primer jeder so klike.

Definicija 14.4 Množica vozlišč C_j je j-klika (na j točkah), če je vsako vozlišče množice povezano z vsemi vozlišči te množice.

Jedra lahko računamo glede na povezave, ki vstopajo v vozlišča (Input), povezave, ki izstopajo iz vozlišč (Output) ali vse povezave (All). V primeru neusmerjenega omrežja so vhodna in izhodna jedra enaka.

Lastnosti jeder:

- jedra so gnezdena, t. j. $i < j \Rightarrow H_j \subseteq H_i$,
- jedra niso nujno povezani podgrafi,
- jedra lahko posplošimo na omrežja z vrednostmi na povezavah.

V Pajku se jedra izračuna z ukazom *Network → Create Partition → k-Core*, kjer se nato izbere vrsto jedra: *Input*, *Output* ali *All*. Rezultat operacij je razbitje, ki vozliščem, pripadajočim istemu jedru, priredi isto število.

p-Klike

Definicija 14.5 Množica vozlišč H je p-Klika, če je vsako vozlišče množice povezano z vsaj p deležem vozlišč te množice.

V konkreten zgledu, če je $p = 0.2$ in $|H| = 100$, je množica vozlišč H p-Klika, če je vsako vozlišče povezano z vsaj 20 drugimi vozlišči iz te množice. Pajek ima za računanje p-Klik vgrajen ukaz *Network → Create Partition → p-Cliques*, kjer se pri izbiri opcije *Strong* upošteva usmeritev povezav, pri *Weak* pa ne.

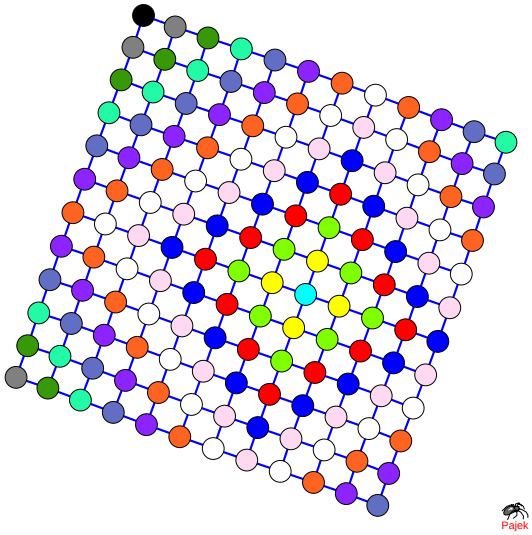
k-Sosedi

V programu Pajek se lahko izračuna najkrajše oddaljenosti vseh vozlišč od izbranega vozlišča x. To omogoča operacija pod *Network → Create Partition → k-Neighbours → Output*, kamor se vnese izbrano vozlišče in pri vprašanju „Distance“ odgovori z `0`, da se pregleda vse oddaljenosti. Rezultat te operacije je razbitje, ki vsakem vozlišču za razred priredi njegovo oddaljenost od izbranega vozlišča. Podobno lahko Pajek izračuna oddaljenosti vseh vozlišč do izbranega vozlišča z *Network → Create Partition → k-Neighbours → Input* in brez upoštevanja usmerjenosti povezav *Network → Create Partition → k-Neighbours → All*. Dobljeno razbitje se prikaže z *Draw → Network + First Partition*. Primer takšnega prikaza, kjer je pripadnost vozlišč k razredom določena z barvo, je na sliki 14.6.

S pomočjo razbitja se lahko iz omrežja izreže podomrežje. Operacija *Operations → Network + Partition → Extract SubNetwork* vpraša po razredih, katere se želi ohraniti. Če nas konkretno zanimajo k-sosedi vozlišča x, to so vozlišča do katerih iz vozišča x obstaja najkrajša pot dolžine k, to so vsa vozlišča, ki so oddaljena za natanako k-povezav, se vpiše k.

Različnost in podobnost

Pri analizi velikih omrežij je vedno problem preglednost. Težko je ugotoviti ali sta si vozlišči podobni in koliko ter ali sta si različni in koliko. Pajek podpira izračun različnih mer različnosti (*Dissimilarity*), ki pa so računsko zahtevnejše in na velikih omrežjih ali pa manj zmogljivih računalnikih potrebujejo več časa. Različnost se računa le na izbrani množici oz. gruči vozlišč (*Cluster*); če želimo matriko različnosti izračunati na celem omrežju, izberemo *Cluster → Create Complete Cluster*, ki nam ustvari gručo vseh vozlišč omrežja.



Slika 14.6: Primer omrežja, kjer so enakoobarvana vozlišča enako oddaljena od izbranega svetlomodrega vozlišča

Nato izračunamo različnost v meniju *Operations → Network + Cluster → Dissimilarity* → Network Based*; izberemo eno od mer različnosti. Z N_v označimo množico vhodnih, izhodnih ali vseh sosedov vozlišča v, $+$ pomeni simetrično razliko množic, $\cup$ unijo množic, $\setminus$ razliko množic, Δ in Δ_2 pa pomenita največjo ter drugo največjo stopnjo v omrežju. Mere različnosti so

$$d_1(u,v) = \frac{|N_u + N_v|}{\Delta + \Delta_2},$$
$$d_2(u,v) = \frac{|N_u + N_v|}{|N_u \cup N_v|},$$
$$d_3(u,v) = \frac{|N_u + N_v|}{|N_u| + |N_v|},$$
$$d_4(u,v) = \frac{\max(|N_u \setminus N_v|, |N_v \setminus N_u|)}{\max(|N_u|, |N_v|)},$$
$$d_5(u,v) = \sqrt{\sum_{s=1, s\neq u,v}^{n} ((q_{us} - q_{vs})^2 + (q_{su} - q_{sv})^2) + p \cdot ((q_{uu} - q_{vv})^2 + (q_{uv} - q_{vu})^2)},$$
$$d_6(u,v) = \sum_{s=1, s\neq u,v}^{n} (|q_{us} - q_{vs}| + |q_{su} - q_{sv}|) + p \cdot (|q_{uu} - q_{vv}| + |q_{uv} - q_{vu}|).$$

Meri d_5 in d_6 sta izračunani glede na neko matriko $Q = [q_{uv}]$ nad vozlišči, npr. matrika sosednosti ali matrika razdalj. Parameter p je običajno nastavljen na vrednost 1 ali 2. Ko velja $N_u = N_v = \emptyset$, nastavimo vse različnosti od d_1 do d_4 na 1. Tako izračunamo različnost med vsakim parom vozlišč, razen če vklopimo opcijo *Among all linked Vertices only*, ki nam izračuna samo različnost med sosedi. Kot izhod dobimo novo omrežje, kjer je utež povezave med vozliščema kar njuna različnost, hierarhijo vozlišč, manj kot sta si dve vozlišči različni, prej sta povezani v hierarhiji, permutacijo vozlišč, ki je potrebna za konstrukcijo hierharije ter še dendrogram hierarhije, ki ga shranimo v *.eps* formatu. Na

sliki 14.7 je prikazan dendrogram omrežja `Imports_manufacturers.net`, ki smo ga dobili s pomočjo d_1 mere različnosti, ki pravzaprav izračuna število različnih sosedov dveh točk in vse skupaj normira.

Ostale operacije

- Operacija *Network* → *Create Partition* → *Valued Core* naredi razbitje glede na posplošena jedra in se izvaja na uteženih omrežjih. Namesto števila sosedov se gleda vsoto po povezavah do sosedov iz istega razreda. Ker so te vrednosti ponavadi realne, določimo intervale, ki ustrezajo posameznim razredom.

- Razbitje omrežja na „otoke“, kjer so vozlišča med sabo bolj povezana kot z okolico glede na uteži po povezavah ali po številu povezav, dobimo z ukazom *Network* → *Create Partition* → *Islands*.

- Ukaz *Network* → *Create Partition* → *Bow-Tie* vrne razbitije po modelu interneta (slika 14.8), ki pravi, da je vsako vozlišče enega od šestih tipov: 1 - LSCC (v največji povezani komponenti), 2 - IN (lahko doseže vozlišča iz LSCC), 3 - OUT (vozlišča iz LSCC lahko dosežejo ta vozlišča), 4 - TUBES (vozlišča, ki niso iz LSCC, vendar povezujejo IN ter OUT), 5 - TENDRILS (vozlišča, ki niso v nobeni komponenti, vendar lahko dosežejo OUT ali pa so dosegljiva iz IN), 0 - OTHER (posebna in nepovezana vozlišča).

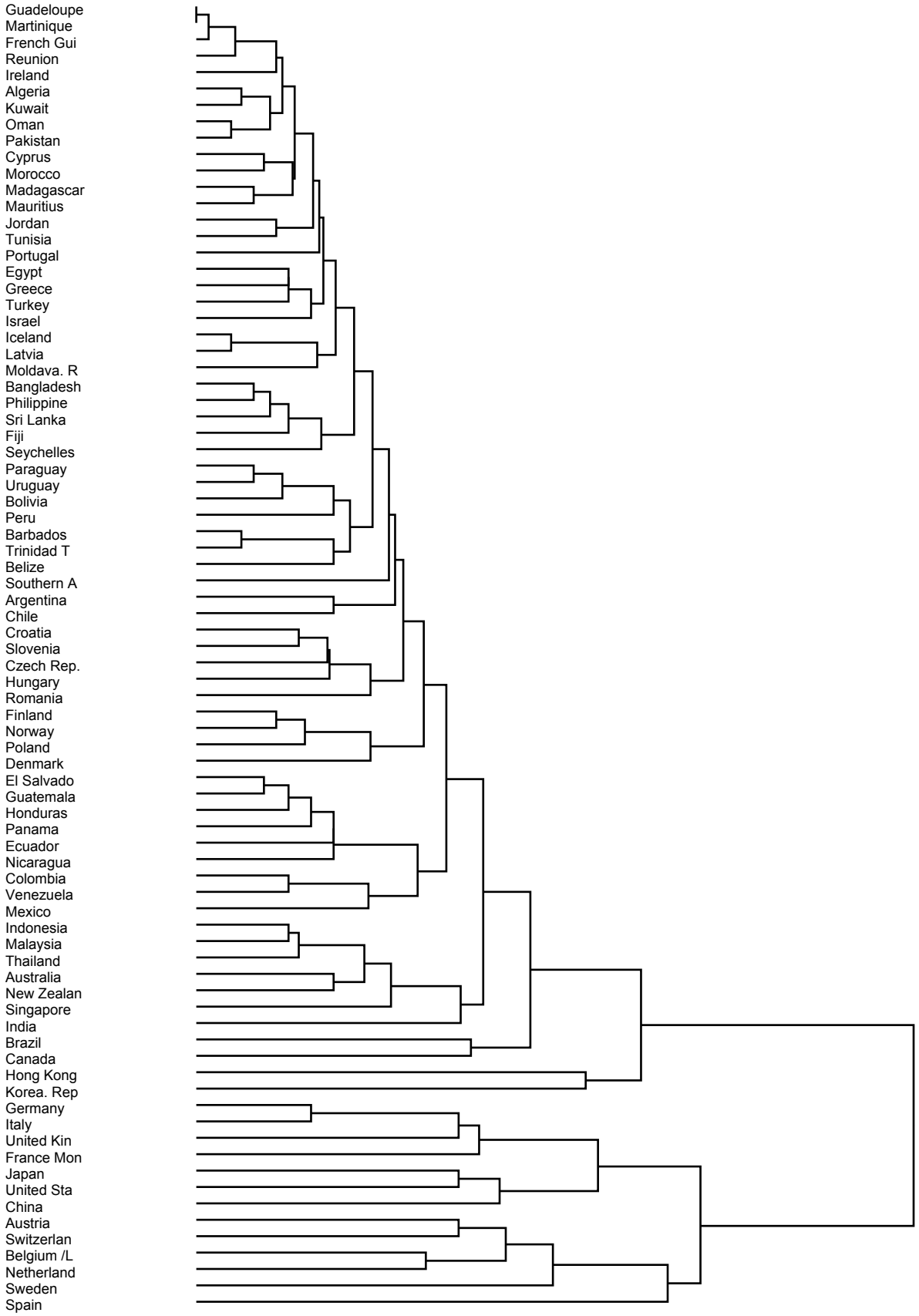
- Skupnosti po metodi Louvain se poišče z ukazom *Network* → *Create Partition* → *Louvain Communities*. Pri tem se omrežje razdeli na razrede tako, da se maksimizira moduliranost.

  Definicija 14.6 Moduliranost je razlika med deležem povezav, ki pripadajo gručam in pričakovanim deležem povezav v gručah v grafu z naključno razporeditvijo povezav na istem številu vozlišč. Naj bo neusmerjen graf razdeljen na g gruč. Definirajmo simetrično $g \times g$ matriko E, katere elementi predstavljajo delež povezav med posameznimi gručami. Moduliranost Q lahko definiramo kot (niso vse definicije enake)
  $$Q = \mathrm{Tr}E - ||E^2||.$$

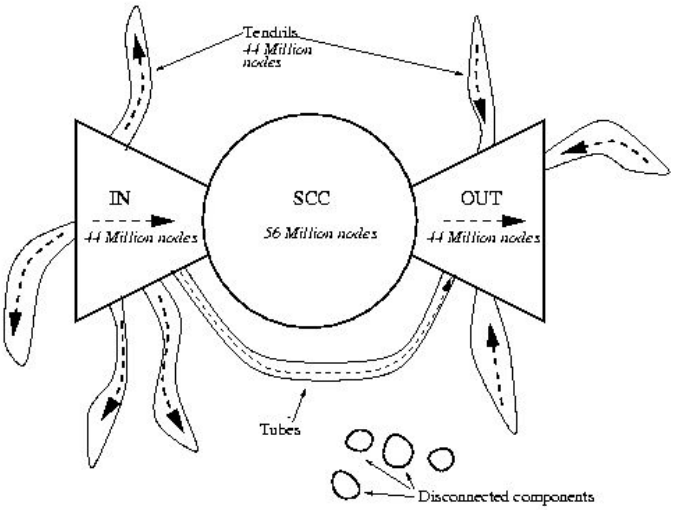
- Razbitje glede na oznake vozlišč, kot smo jih definirali v omrežju, npr. glede na imena molekul ali atomov v omrežju kemijskih reakcij, dobimo s pomočjo *Network* → *Create Partition* → *Vertex Labels*. Če pa želimo razbitje glede na oblike vozlišč, kot smo jih definirali v omrežju, npr. elipse, kvadrati, diamanti ..., uporabimo *Network* → *Create Partition* → *Vertex Shapes*

14.5 Vektor

Vektorji so še ena od pomembnih struktur v programu Pajek. Podobno kot pri razbitjih vsakemu vozlišču priredimo neko število, le da je tokrat to število lahko tudi realno. Vektorje naložimo v polje *Vectors* ali pa jih izračunamo z ukazom *Network* → *Create Vector*.



Slika 14.7: Dendrogram podobnosti



Slika 14.8: Bowtie struktura interneta

14.5.1 Operacije

Stopnje vozlišč

Definicija 14.7 Vhodna stopnja točke (input degree) je število povezav, ki vstopajo v vozlišče. Izhodna stopnja (output degree) je število povezav, ki izstopajo iz vozlišča. (Skupna) stopnja vozlišča (all degree) je število povezav, ki imajo krajišče v vozlišču.

V programu Pajek se stopnje vozlišč izračuna z *Network* → *Create Vector* → *Centrality* → *Degree*, kjer se nato izbere *Input*, *Output* ali *All* v odvisnosti, katere zgoraj definirane stopnje nas zanimajo. Stopnje vozlišč se nato shranijo v vektor.

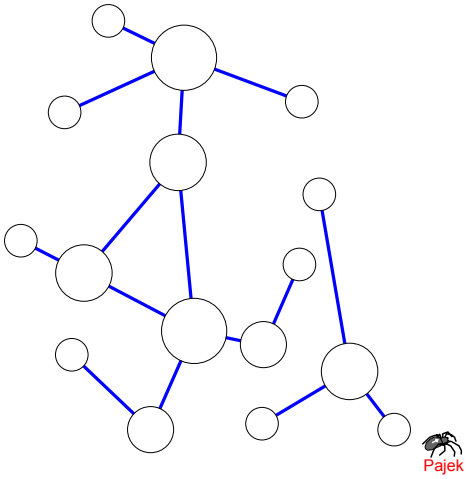
Za grafično ponazoritev lahko vektor narišemo skupaj z risbo omrežja (*Draw* → *Network + First Vector*), kjer bodo velikosti vozlišč sorazmerne z vrednostjo v vektorju. Primer takšne risbe je prikazan na sliki 14.9.

Če želimo pri grafični ponazoritvi stopnjam vozlišč prirediti barve, to najlažje naredimo tako, da vektor najprej spremenimo v razbitje, kar se stori z *Vector* → *Make Partition* → *by Intervals* → *First Threshold and Step*, kjer za `First Threshold` vpišemo najnižjo stopnjo vozlišč in za `Step` 1. Nato se dobljeno razbitje nariše z *Draw* → *Network + First Partiton*. Če poleg barv želimo tudi velikosti vozlišč, izberemo *Draw* → *Network + First Partiton + First Vector*.

Indeksi centralnosti

Za izračun indeksa centralnosti za vsako vozlišče posebej ali centralni indeks celotnega omrežja so na voljo naslednje operacije.

- *Network* → *Create Vector* → *Centrality* → *Weighted Degree*, ki vrne uteženo sosedsko centralnost – vsakemu vozlišču priredi vsoto uteži na povezavah, ki imajo to vozlišče za rob.
- *Network* → *Create Vector* → *Centrality* → *Closeness*, ki vrne bližinsko centralnost – meri oddaljenost vozlišča do vseh drugih vozlišč v omrežju.



Slika 14.9: Omrežje, kjer so velikosti vozlišč sorazmerne stopnji

- *Network* → *Create Vector* → *Centrality* → *Betweenes*, ki vrne centralnost posrednika na najkrajši poti – pove, kolikšen delež najkrajših poti v grafu vsebuje dano vozlišče.
- *Network* → *Create Vector* → *Centrality* → *Hubs-Authorities*, ki poišče pomembnejša vozlišča v omrežju. Vozlišče je dober „Hub“, če kaže na veliko dobrih „Authorites“ in je dober „Authority“, če nanj kaže veliko „Hub“-ov.
- *Network* → *Create Vector* → *Centrality* → *Proximity Prestige*, ki izračuna količnik med deležem vozlišč, povezanih z vozliščem, za katerega računamo indeks, kar lahko poimenujemo domena vpliva, in povprečno razdaljo med izbranim vozliščem in vsemi drugimi iz njegove domene vpliva.
- *Network* → *Create Vector* → *Centrality* → *Line Values*, ki vsakemu vozlišču priredi najmanjšo ali največjo utež povezave, ki ima to vozlišče za rob.
- *Network* → *Create Vector* → *Centrality* → *Centers*, ki poišče centre omrežja z uporabo algoritma kraje (*robbery algorithm*). V začetku vsako vozlišče dobi neko vrednost, relativno z njegovo stopnjo. Ko algoritem najde šibko vozlišče, ga sosedje okradejo in si prilastijo njegovo vrednost glede na to, koliko so sami močni.

Koeficient gručavosti

Koeficient gručavosti se izračuna z *Network* → *Create Vector* → *Clustering Coefficients*. Naj $\deg(v)$ označuje stopnjo vozlišča v, $|E(N_1(v))|$ število povezav v 1-soseski vozlišča v (vozlišča, ki so od v oddaljena največ 1), Δ maksimalno stopnjo vozlišča v omrežju in $|E(N_2(v))|$ število povezav v 2-soseski vozlišča v (vozlišča, ki so od v oddaljena največ 2).

- CC_1 - koeficient, izračunan na podlagi 1-soseske:

$$CC_1(v) = \frac{2|E(N_1(v))|}{\deg(v) \cdot (\deg(v) - 1)}, \quad CC_1'(v) = \frac{\deg(v)}{\Delta} CC_1(v).$$

- CC_2 - koeficient, izračunan na podlagi 2-soseske:

$$CC_2(v) = \frac{|E(N_1(v))|}{|E(N_2(v))|}, \quad CC_2'(v) = \frac{\deg(v)}{\Delta} CC_2(v).$$

Če je $\deg(v) \leq 1$ dobijo vsi koeficienti gručavosti vozlišča v manjkajočo vrednost (9999998). Koeficient gručavosti je podan ravno z obratno vrednostjo, kot bi si želeli, vendar lahko inverzne vrednosti v vektorju hitro izračunamo z *Vector* → *Transform* → *Invert*.

14.6 Primeri uporabe

Poglejmo si še dva primera uporabe programa Pajek v praksi.

14.6.1 Delo s razbitji

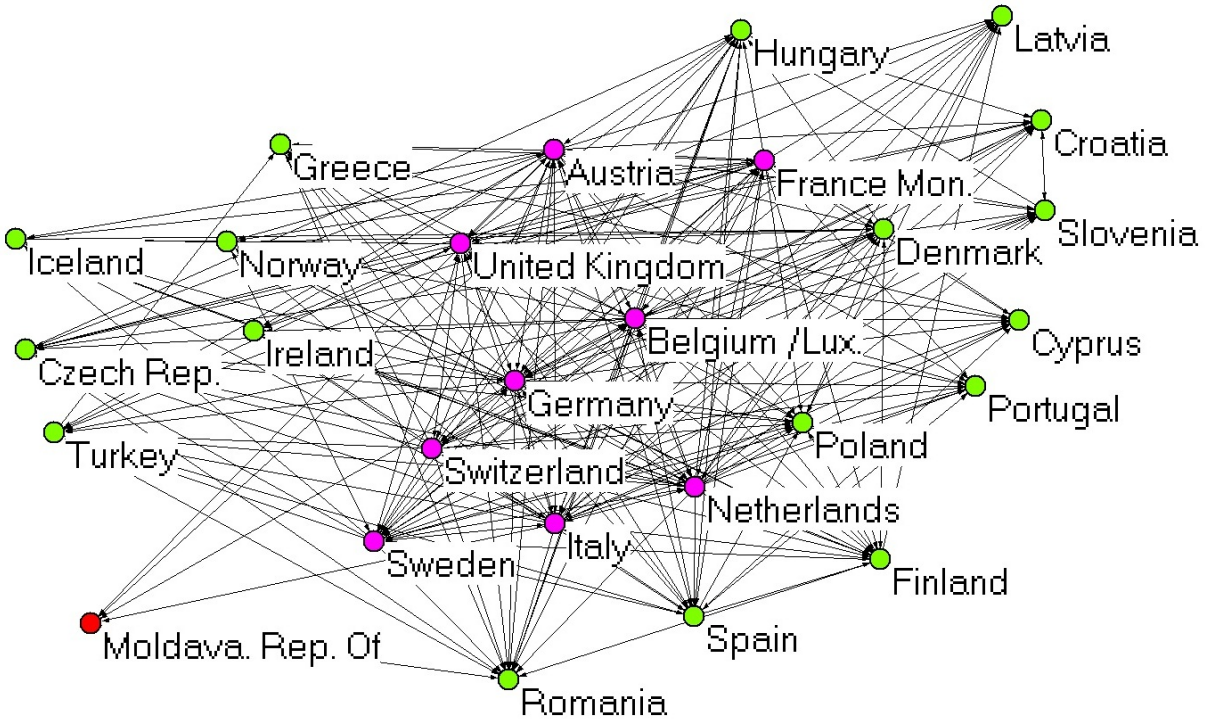
V Pajek naložimo omrežje iz datoteke `Imports_manufacturers.net`, ki pripada drugemu poglavju knjige *Exploratory Social Network Analysis with Pajek* [50] in ki je prosto dostopna na uradni spletni strani [51]. Ker nas zanima trgovanje evropskih držav, bi nam prav prišlo razbitje držav po celinah. Na srečo so avtorji že ustvarili razbitje `Continent.clu`, ki jo je potrebno naložiti v program Pajek. Ker nas zanima številka razreda, ki ponazarja Evropo, v razbitju poiščemo znano evropsko državo npr. Austria in pogledamo vrednost razbitja, tukaj je to 3.

Z *Operations* → *Network + Partition* → *Exctract SubNetwork* izberemo razrede, ki jih želimo izrezati iz omrežja. Konkretno nas zanima razred 3 – Evropa, torej v okence vpišemo 3. Dobili smo omrežje trgovanja med evropskimi državami.

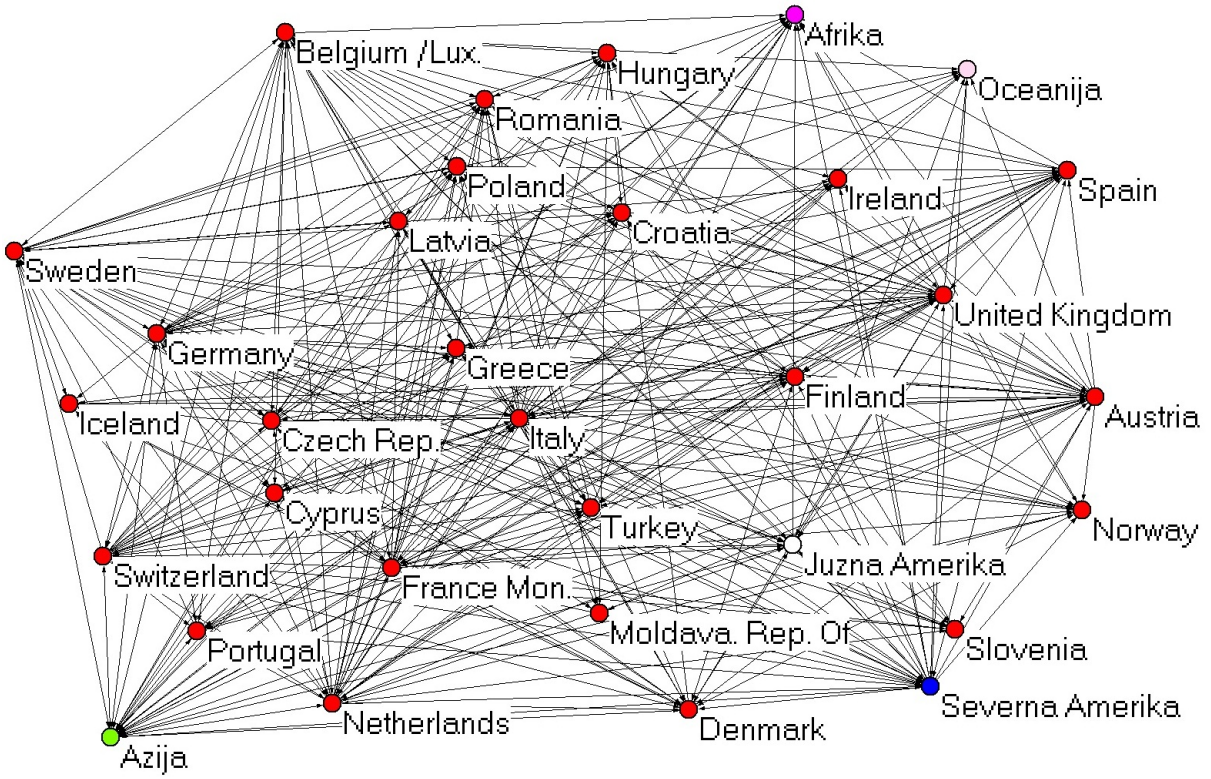
Recimo, da nas zanima še, kako so razvite te države. Razbitje po razvitosti držav imamo shranjeno v datoteki `World_system.clu`, vendar to razbitje velja za celo omrežje in ne samo za Evropo, zato bi razbitje radi skrčili. V prvo polje razbitij naložimo datoteko `World_system.clu`, v drugo polje pa `Continent.clu`. Z ukazom *Partitions* → *Extract SubPartition (Second from First)* in izbiro razredov drugega razbitja, ki nas zanimajo, v našem primeru samo 3 – Evropa, dobimo novo razbitje evropskih držav po razvitosti. Z ukazom *Draw* → *Network + First Partition* dobimo sliko 14.10.

Včasih se zgodi, da samo z izločanjem podomrežja izgubimo pomembne podatke. V našem primeru tako ne vemo, kako evropske države trgujejo z državami na preostalih celinah. Ponovno naložimo omrežje `Imports_manufacturers.net` in razbitje `Continent.clu` ter uporabimo ukaz *Operations* → *Network + Partition* → *Shrink Network*. Pajek nas najprej vpraša po najmanjšem številu povezav v razbitju, da celo razbitje še stisne v eno vozlišče; tu izberemo 0, saj želimo vse države z ene celine združiti. Nato nas vpraša po razbitju, ki je ne želimo stisniti oziroma skrčiti; izberemo 3 – Evropa. Tako smo dobili novo omrežje (slika 14.11), ki nam podaja trgovanje znotraj Evrope, vidne pa so tudi medcelinske povezave. Za uporabniku prijaznejši prikaz lahko preimenujemo razbitja; kliknemo na *Edit Partition* in v stolpcu *Label* razbitja, ki se začnejo z # preimenujemo; npr. #Argentina → Južna Amerika, #Bangladesh → Azija ....

Če nas zanima globalna trgovina in samo posli med celinami, s podobnim postopkom stisnemo vse kontinente. Naložimo omrežje `Imports_manufacturers.net` in razbitje `Continent.clu` ter uporabimo ukaz *Operations* → *Network + Partition* → *Shrink Network*. Vpišemo najprej 0 za minimalno število povezav v razbitju, da jo še stisnemo in nato še enkrat 0, da povemo Pajek-u naj stisne vsa razbitja. Tako dobimo omrežje trgovanj med celinami. Za lažjo obdelavo podobno kot prej še preimenujemo razbitja, da namesto predstavnikov kontinenta vidimo kar ime posamezne celine. Pri večjih omrežjih, sploh če so usmerjena, običajno težko opazimo neobstoječe povezave, tukaj npr. ne vidimo takoj ali Južna Amerika izvaža surovine v Azijo. Za nazornejšo predstavo uporabimo *File*

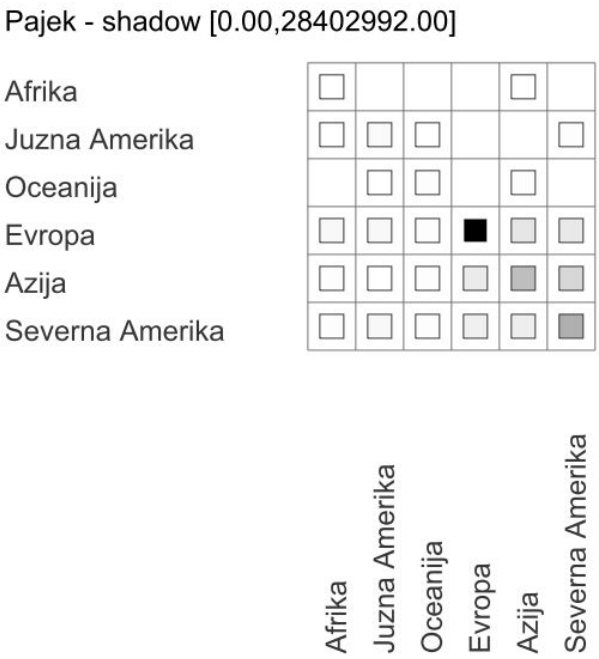


Slika 14.10: Trgovanje evropskih držav in njihova razvitost



Slika 14.11: Trgovanje evropskih držav

→ *Network* → *Export as Matrix to EPS* → *Original*. Dobili smo grafični prikaz matrike povezav (slika 14.12), kjer takoj opazimo, da evropske države med sabo močno trgujejo (temnejši kvadratek) in da Južna Amerika ne izvaža surovin niti v Evropo niti v Azijo.



Slika 14.12: Matrika medcelinskih trgovanj

14.6.2 Delo z vektorji

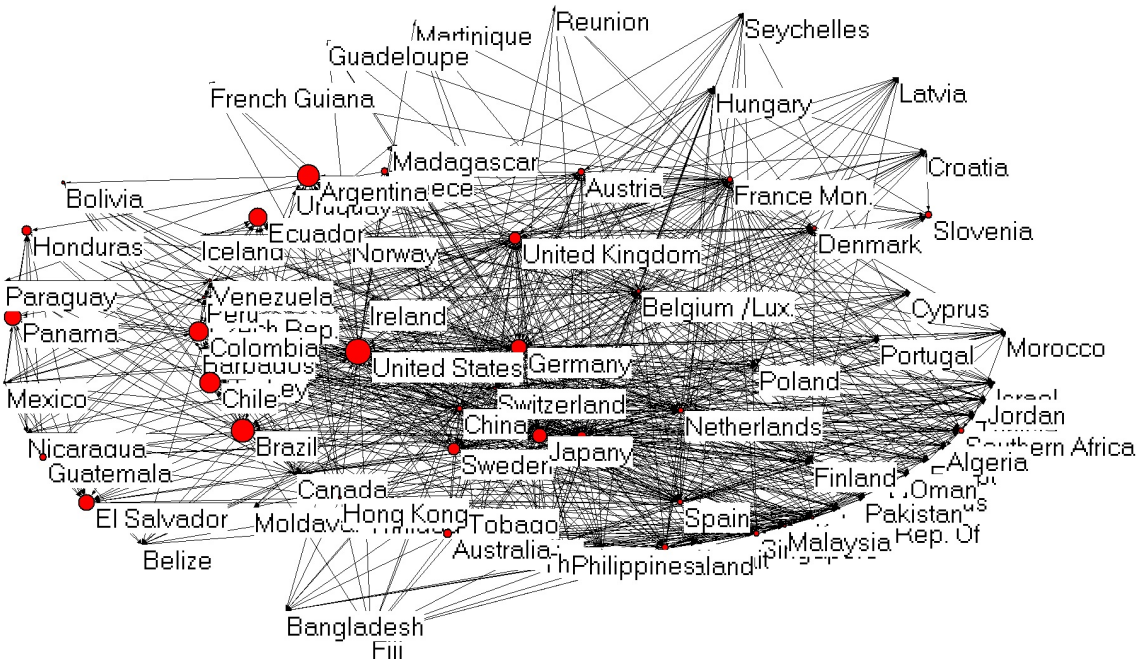
Poglejmo si primer konkretne uporabe kar na že znanem omrežju `Imports_manufacturers.net`. Ko smo v polje *Networks* naložili omrežje, poiščimo *Betweenness centrality*, ki nam meri delež najkrajših poti med vsemi vozlišči, ki gredo skozi izbrano vozlišče. V našem primeru bi to lahko pomenilo uspešnost preprodaje ali pomembnost države na trgu. Preprosto izberemo *Network* → *Create Vector* → *Centrality* → *Betweenness*. Narišimo to omrežje, upoštevajoč vrednosti v vektorju z *Draw* → *Network + First Vector*.

Dobili smo zelo nepregledno sliko 14.13, vendar vidimo, da so nekatere države npr. ZDA, Brazilija, Nemčija in Argentina pomembnejše za svetovno trgovino. Recimo, da nas sedaj zanimajo centralnosti samo na državah Južne Amerike, kjer izključimo preostali svet; vektor je potrebno oklestiti. Naložimo razbitje `Continent.clu`, vektor obdržimo od prej. Izberimo *Operations* → *Vector + Partition* → *Extract Subvector* in vpišimo 6 – številka razbitja, ki ponazarja Južno Ameriko (Argentina je v Južni Ameriki, razredu z vrednostjo 6). Pajek nam vrne podvektor.

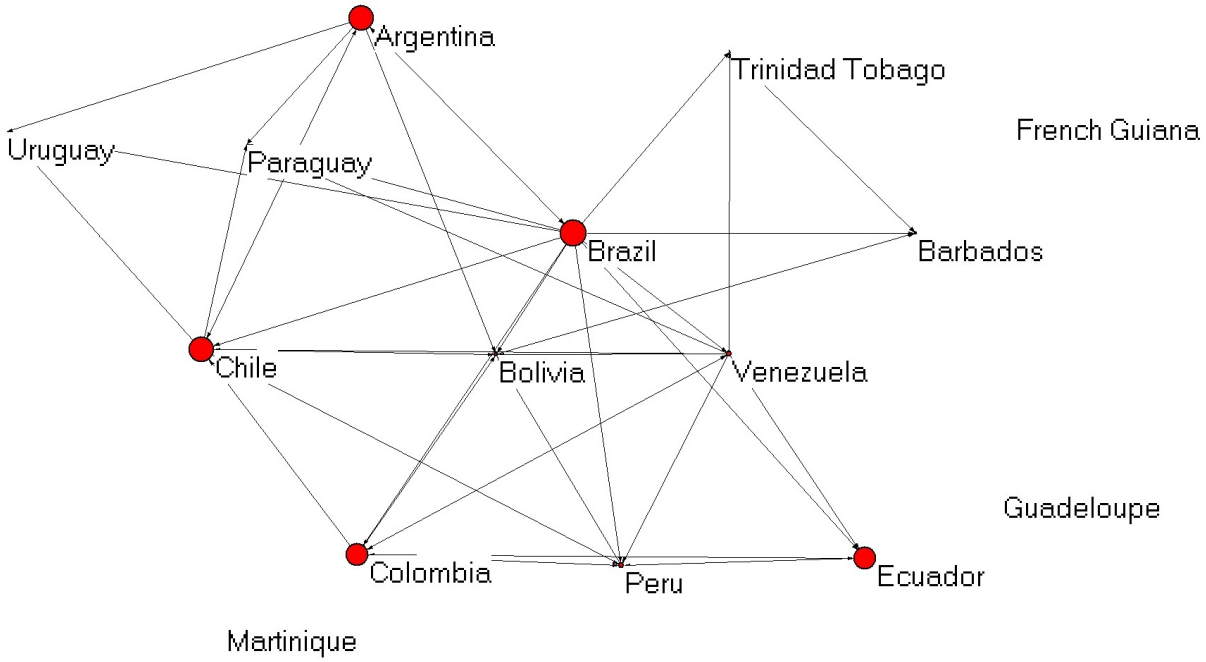
Če hočemo podatke prikazati, je potrebno ekstrahirati še omrežje. Kot smo se že naučili, izberemo *Operations* → *Network + Partition* → *Extract SubNetwork* in vnesemo 6, da dobimo samo Južno Ameriko. Sedaj to omrežje, upoštevajoč vrednosti v vektorju, še narišimo z *Draw* → *Network + First Vector*. Iz slike 14.14 je očitna pomembnost Brazilije, Čila in Argentine, medtem ko so Gvadalupa, Martinik in Francoska Gvajana popolnoma brez stikov z drugimi državami Južne Amerike.

14.7 Zaključek

Pajek je program, ki je namenjen analizi in prikazu velikih omrežij in to svojo nalogo opravlja tako dobro, da je v svetovnem merilu poznan skoraj vsakemu, ki se ukvarja z



Slika 14.13: Centralnost vseh držav



Slika 14.14: Centralnost držav Južne Amerike

velikimi omrežji.

Poleg splošne uporabnosti programa Pajek za grafično prikazovanje tudi manjših omrežij ter prej opisanih ukazov program podpira še mnogo vsestranskih operacij, ki so sistematično razporejene in opisane v [48].

Za nadaljevanje spoznavanja programa Pajek je priporočena knjiga Exploratory Social Network Analysis with Pajek[50], kjer se bralec korak za korakom in na praktičnih primerih iz analize socialnih omrežij nauči uporabljati program.

Poglavje 15

Omrežne strukture v resničnem svetu in njihovi mehanični omrežni modeli

Simon Škerjanec

15.1 Omrežne strukture v resničnem svetu

1. Skoraj vsa kompleksna omrežja imenujemo majhni svetovi (small worlds), oz. z drugimi besedami, so lokalno gosto povezana [178];

2. Večina vseh kompleksnih omrežij je brez-lestvičnih (scale-free) - večina vozlišč ima nizko stopnjo, nekatera pa imajo tudi izjemno visoko [166];

3. Večina jih je segmentiranih, torej je mogoče razkosati grafe v skupine zgoščenih podgrafov, katerih medsebojne povezave se redke [168][175];

4. Nekatera izmed omrežji delujejo fraktalno [177].

Ostale strukture, predvsem majhni podgrafi v usmerjenih omrežjih, tako imenovanih omrežnih motivih, so bili prepoznani v samo nekaj omrežjih, predvsem bioloških [173] [174]. Mi se bomo posvetili samo prvima dvema strukturama, ker sta poleg analize fraktalov med najbolj uporabnimi za računalniško obdelavo.

15.1.1 Mali svetovi

Leta 1998 sta Watz in Strogatz [178] poročala, da so v mnogih omrežjih resničnega sveta najvišja vozlišča lokalno visoko povezana oz. segmentirana, ob enem pa je med vozlišči moč zaznati majhno povprečno medsebojno razdaljo. Da bi izmerili segmentiranost sta vpeljala tako imenovan *koeficient segmentiranosti* $CC(v)$ vozlišča v:

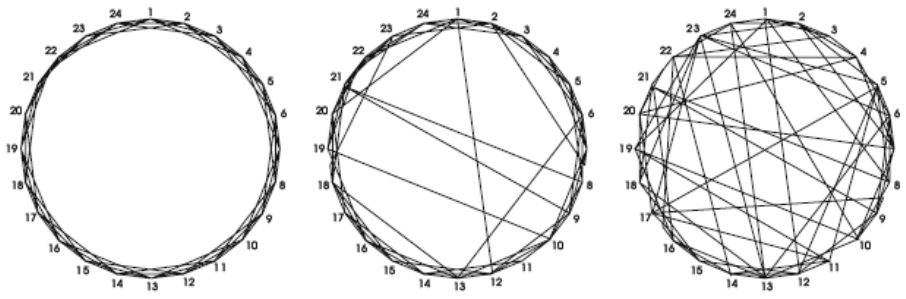
$$\mathrm{CC}(v) = \frac{2e(v)}{\deg(v)(\deg(v)-1)},$$

kjer $e(v)$ označuje število povezav med sosedi v. Ker imenovalec daje možno število teh povezav, *koeficient segmentiranosti* posamezne najvišje točke označuje možnost, da sta dva izmed sosedov tudi povezana s povezavo. *Koeficient segmentiranosti* $CC(G)$ grafa je povprečje vseh skupnih koeficientov njegovih vozlišč. Visoko povprečje keoficienta v grafu kaže, da so najvičja vozlišča lokalno povezana in da bo povprečna razdalja med vozlišči, glede na graf, velika. Nepričakovano je premer omrežij resničnega sveta, ki je bil analiziran v spisu *Wattsa* in *Strogatza* [178] bolj podoben tistim ustreznim naključnim grafom (corresponding random graph) iz $G(n,p)$ [180] modela, ne glede na njihov visok koeficient. Naključen graf naj bi bil ustrezen omrežju resničnega sveta, če ima enako število vozlišč in enako število povezav. Tak graf lahko dosežemo, če p postavimo na $2m/(n(n-1))$. Seveda bo pričakovan koeficient vozlišč v takem grafu enak p, saj je verjetnost da sta katerakoli dva soseda vozlišča v povezana enaka p. Izpadlo je, da je imelo omrežje resničnega sveta, ki sta ga avtorja analizirala koeficient 1000 krat višji kot pri ustreznem naključnem grafu.

Prvi mehanistični model, z lastnostmi visokih koeficientov v kombinaciji z majhno povprečno razdaljo, je bil podan s strani *Wattsa* in *Strogatza* [178]. Začela sta z določenimi n vozlišči v krožnem zaporedju, kjer je vsako vozlišče povezano s svojimi k, v smeri urinega kazalca, naslednjimi sosedi. Povezave so neusmerjene. Takemo grafu pravimo *cirkulanten graf* (circulant graph). Vsaka povezava je kasneje ponovno povezana z verjetnostjo $0 \leq p \leq 1$. Da bi lahko povezali povezavo $e = (v,w)$ je novo ciljano vozlišče w' izbrano naključno in (v,w) je nadomeščen z (v,w'). Jasno je, da če je $p = 0$ je koeficient določen z:

$$CC(G) = \frac{3(k-1)}{2(2k-1)}$$

ki se v limiti za velike k približuje $3/4$. Povprečna razdalja med pari vozlišč, ki se povečuje linearno s številom vozlišč je enaka $n/4k$. ÄŚe zdaj analiziramo povprečen koeficient in povprečno razdaljo z upoštevanjem p-ja, lahko vidimo da povprečna razdalja pada veliko hitreje kot povprečen koeficient.



(a) $G_c(24,3)$, $p = 0$ (b) $p = 0.1$ (c) $p = 1$

Slika 15.1: a) Krožni graf z 24 vozlišči, kjer je vsaka točka povezana s tremi sosednjimi; b) Vsaka povezava je bila povezana z verjetnostjo $p = 0.1$; c)Vsaka povezava je bila povezana z verjetnostjo $p = 1$.

Sledi, da je za majhen p, povprečen koeficient še vedno precej visok, med tem, ko je povprečna razdalja že padla na vrednost, ki je primerljiva z ustreznim naključnim grafom.

Režim v katerem se to zgodi definira celoto omrežij, ki kažejo efekt majhnega sveta ali na kratko: celoto omrežij majhnega sveta.

Poleg zgornjega modela so drugi avtorji predlagali, da bi omrežja majhnega sveta oblikovali s sestavljanjem mrežastega grafa. Najbolj preprost model je, če vzamemo d-dimenzionalni mrežasti [181][182] graf na n vozliščih in dodatno povežemo vsak par vozlišč z verjetnostjo p. Ta hibridni model grafa je določen z $G_d(n,p)$. Seveda, če je p okoli $(\log n)^{1+\epsilon}/n$ za neko konstanto $\epsilon > 0$ bodo povezave odredile naključen del grafa, ki bo napeljal na povprečno razdaljo $O(\log n)$ [167]. Kot smo pokazali, tudi majhne vrednosti p-ja zadostujejo za zmanjševanje premera poli-logaritmičnega izraza:

Lema 15.1 *[170] [179] Za $p = \frac{1}{cn}$, c iz pozitivnih realnih števil in $\epsilon > 0$ je premer $G_p(n,p)$ asimptotično omejen z visoko verjetnostjo*

$$d\left(\sqrt[d]{c(\log n)^{1+\epsilon}} - 1\right)\left(\frac{\log n}{(1+\epsilon)\log\log n - \log 2} + 1\right).$$

Na splošno povedano: premer kombiniranega grafa je linearno odvisen od dimenzije d na spodaj ležečem omrežju $O(\log n^{1+(1+\epsilon)/d})$. Ta rezultat je lahko posplošen kot:

Lema 15.2 *[170] [179] Za vsako funkcijo $(\log n)^{-1+\epsilon} \leq f(n) \leq n^{1-\delta}$, ϵ, $\delta > 0$ in $p = \frac{1}{f(n)n}$ se premer $G_d(n,p)$ asimptotsko približuje z visoko verjetnostjo*

$$d\left(\sqrt[d]{f(n)(\log n)^{1+\epsilon}} - 1\right)\left(\frac{\log(n/f(n))}{\log(\log n)^{1+\epsilon}}\right).$$

Na grobo, če je p nastavljen na $1/(n\log n)$ je le vsako *logn*-to vozlišče povezano z naključno povezavo, povprečna razdalja v $G_d(n,p)$ modelu za velik n in $d = 2$ pa je enaka $O(\log^{2+\epsilon/2} n)$. Ta rezultat je pomemben za načrt omrežja: če predvidevamo, da se strošek za povezovanje dveh komunikacijskih naprav z njuno razdaljo stopnjuje. ÄŚe so narejene povezave samo do določene razdalje, se bo premer dobljenega omrežja večal približno linearno glede na največjo razdaljo. Naš rezultat pokaže, da mora biti povezanih le malo naključnih povezav, da bi lahko skrčili premer omrežja na kvadratno logaritmičen izraz.

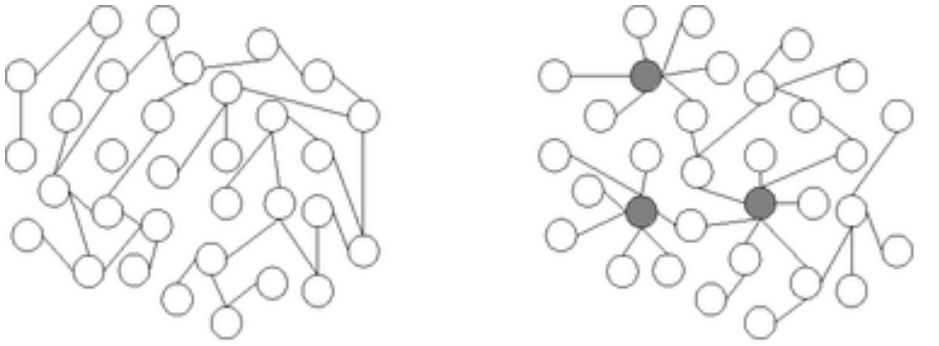
Lahko bi pokazali, da so majhni svetovi navzoči povsod. Ĺ e posebej pomembno za računalniške znanstvenike, imajo vsa tehnična komunikacijska omrežja, kot npr. internet in različna pokrivna omrežja, lastnost da kažejo majhno povprečno razdaljo skupaj z visoko segmentacijo. ÄŚe simulirajo novo različico internetnega protokola ali nov P2P sistem, je zelo pomembno, da ju simuliramo na omrežnem modelu, ki vključuje ti dve lastnosti.

Učinek majhnega sveta predpostavlja, da je povprečna razdalja omejena z $O(\log^k n)$ za neko konstanto k. Zgoraj navedeni modeli bodo za rastoče n pokazali naraščajočo povprečno razdaljo.

15.1.2 Brez-lestvična omrežja

Naj za dani graf G, $P(k)$ predstavlja verjetnost, da iz vseh vozlišč grafa G, vozlišče s stopnjo k, izberemo naključno in enakomerno. Kompleksno omrežje je brez lestvično, ko $P(k)$ raste z $k^{-\gamma}$, kjer je γ pozitivna konstanta [166]. Za omrežja resničnega sveta je $2 \leq \gamma \leq 3$ [176]. Najlažji način za zaznavo brez-lestvične porazdelitve je, da ocenimo $P(k)$ in pokažemo njegovo odvisnost od k-ja v dvojno logaritmskem diagramu.

Kot je očitno pri strukturi naključnega grafa, je stopnja za velike n porazdeljena normalno. Torej ima brez lestvični graf z enakim številom vozlišč in povezav v primerjavi z naključnim grafom, veliko več nižje-stopenjskih vozlišč in pa nekaj vozlišč z veliko višjo stopnjo kot bi bilo pričakovati v naključnem grafu. Tem visoko-stopenjskim vozliščem pravimo tudi žarišče (hub).



(a) Random network (b) Scale-free network

Slika 15.2: a) Naključni graf in b) brez lestvično omrežje. V brez lestvičnem omrežju, so žarišča potemnjena.

Prvi model, ki razloži kako se lahko taka brez-lestvična porazdelitev pojavi je *prednostna povezanost* (preferential attachment) ali the-rich-get-richer model Barbasija in Alberta [166]: to je dinamični omrežni model, kjer na vsakem koraku i dodamo novo vozlišče v_i skupaj s k slučajnimi povezavami. Začne se z majhnim naključnim grafom, ki ima vsaj k vozlišč. Sledi, da novo vozlišče v_i izberemo izmed že obstoječih k vozlišč, vsako z verjetnostjo, ki je proporcionalna stopnji na tej točki in ustvarimo povezavo. Bolj natančno, verjetnost, da bo na novo vpeljano vozlišče v_i izbralo vozlišče w, je proporcionalna z $deg(w)$. ÄŚe ima vozlišče že visoko stopnjo, ima večjo možnost da dobi novo povezavo v vsakem časovnem koraku, to je proces, ki mu rečemo *prednostna povezanost*. Ta postopek v limiti za velike n ustvari brez lestvično omrežje.

Veliko omrežij realnega sveta prikazuje, da imajo brez lestvično porazdelitev. Na primer internet [169], weblink graf [166], ali stran človeških spolnih kontaktov [171][172]. Predvsem ugotovitve na omrežjih spolnih kontaktov imajo pomembne posledice: lahko razložimo zakaj se spolne bolezni tako enostavno razširijo, z iskanjem aktivnih okuženih ljudi, pa lahko pomaga preprečiti širjenje bolezni, in jih ozdravi. To sledi iz dejstva, da je brez lestvično omrežje lahko prekiniti že, če je majhna frakcija visoko stopenjskih vozlišč odstranjena. Z brezlestvičnimi omrežji lahko pojasnimo zakaj nekateri računalniški virusi ostanejo zelo dolgo v omrežju: Pastor Satorras in Vespignani debatirata o modelu virusa, ki se širi po brezlestvičnem omrežju in pokažeta, da v takem omrežju ni kužne meje, ni minimalne gostote infekcije, ki bi omogočala infekcijo celotnega omrežja. V takem omrežju lahko vsak virus potencialno okuži vso omrežje.

15.1.3 Robustnost naključnega in brez lestvičnega omrežja

Zanimiva ugotovitev Alberta in ostalih [165] je, da je drugačna omrežna struktura pokazala veliko razliko v robustnosti v primerjavi z naključnimi propadi in usmerjenimi

napadi na njihovo strukturo [165]. Robustnost omrežja definirajo kot povprečno razdaljo v omrežij, ko je bil iz grafa že odstanjen določen % vozlišč. Odstranitev je modelirana na dva načina: da prikažemo naključen propad npr. internetnega serverja, je verjetnost za propad vsakega vozlišča enakomerno porazdeljena, da prikažemo usmerjen napad na zlobnega nasprotnika, ki pozna strukturo omrežja, moramo iz omrežja odstranili vozlišče z najvišjo stopnjo. Albert in ostali so tako prikazali, da je v naključnem scenariju propada, robustnost brez lestvičnega omrežja veliko višja, kot robustnosti primerljivega naključnega grafa. Po odstranjevanju več in več vozlišč, naključni graf končno razpade v veliko manjših povezanih komponent, med tem ko vozlišča v brezlestvičnem omrežju še vedno oblikuje eno veliko povezano komponento. Pri napadalnem scenariju je popolnoma obratno, ko naključni graf ostane povezan in pokaže majhno povprečno razdaljo, bo brez lestvično omrežje razpadlo, ko bo odstranjenih samo nekaj visoko stopenjskih vozlišč.

Tako vedenje zlahka pojasnimo: v naključnem scenariju propada, bo imela večina odstranjenih omrežij v brez lestvičnem omrežju zelo majno stopnjo ker ima večina vozlišč v brez lestvičnem omrežju nizko stopnjo. V naključnem grafu, imajo skoraj vsa vozlišča enako stopnjo in pomembnost pri povezanosti grafa. Ta lastnost služi omrežju v primeru usmerjenega napada. Vendar brez lestvično omrežje zelo hitro izgubi visok procent svojih povezav, če odstranimo njegove visoko stopenjska vozlišča, zato je tudi tako omrežje zelo občutljivo na tak napad. Ta rezultat je precej slab, ker je večina naših komunikacijskih in nekaterih transportnih omrežij, predvsem letalskih, brez lestvičnih. Albert in ostali so še pokazali, kako lahko je ta omrežja prekiniti.

Literatura

[1] B. Plestenjak, *Uvod v numerične metode*, dostopno na: http://ucilnica.fmf.uni-lj.si/course/view.php?id=85, dne 12.5.2012.

[2] A. Broder, *Graph structure in the web*, Computer Networks **33** (2000) 309–320.

[3] C. Demetrescu, G. F. Italiano *A new approach to dynamic all pairs shortest paths* In Proceedings of the 35th Annual ACM Symposium on the Theory of Computing (STOC'03), 159–166.

[4] E. W. Dijkstra *A note on two problems in connection with graphs*, Numerische Mathematik (1959)1:269–271.

[5] D. Eppstein in J. Wang, *Fast Approximation of Centrality*, Journal of Graph Algorithms and Applications **8** (2004) 27-38.

[6] R. W. Floyd *Algorithm 97: Shortest path*, Communications of the ACM (1962), 5(6):345.

[7] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of American Statistical Association **301**(1963) 713-721.

[8] A. Kamvar, *Exploiting the Block Structure of the Web for Computing PageRank*, Stanford University Technical Report (2003).

[9] L. Roditty, U. Zwick, *On dynamic shortest paths problems*, In Proceed- ings of the 12th Annual European Symposium on Algorithms (ESA'04), volume 3221 of Lecture Notes in Computer Science (2004), 580–591

[10] M. Thorup, *Fully dynamic all-pairs shortest paths: Faster and allowing neg- ative cycles* In Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT'04), volume 3111 of Lecture Notes in Computer Science (2004), 384–396.

[11] S. Warshall, *A theorem on boolean matrices*, Journal of the ACM (1962), 9(1):11–12.

[12] U. Brandes, T. Erlebach, *Network Analysis*, Springer **2005**.

[13] R. Cvahte, *Povzporejanje metahevristik za NP-polne probleme*, diplomsko delo, FRI, **2010**.

[14] *http://www.math.ucsd.edu/ fan/complex/*, **2012**

[15] A. Broder, R. Kumar, F. Maghoul, *Computer Networks: The International Journal of Computerand Telecommunications Networkin*, page 30-320, **2000**.

[16] D. Magoni , J. J. Pansiot, *Analysis of the autonomous system network topology. Computer Communication Review*, **2001**.

[17] P. Radoslavov, H. Tangmunarunkit, R. Govindan *On characterizing network topologies and analyzing their impact on protocol design. Technical Report 00-731*, Computer Science Department, University of Southern California, **2000**.

[18] Ulrich Brandes, Thomas Erlebach, *Network Analysis*, Springer-Verlag, Berlin, Heidelberg, 2005.

[19] Brendan D. McKay. Practical graph isomorphism. Congressus Numerantium, 30:45–87, 1981.

[20] Victor Nicholson, Chun Che Tsai, Marc A. Johnson, and Mary Naim. A subgraph isomorphism theorem for molecular graphs. In Proceedings of The International Conference on Graph Theory and Topology in Chemistry, pages 226–230, 1987.

[21] Apostolos Papadopoulos and Yannis Manolopoulos. Structure-based similarity search with graph histograms. In DEXA Workshop, pages 174–178, 1999

[22] Michael R. Garey and David S. Johnson. Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, 1979.

[23] James J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. Software - Practice and Experience, 12(1):23–24, 1982.

[24] *Graph Isomorphism*, [ogled 3. 5. 2012], dostopno na `http://webspace.ship.edu/deensley/discretemath/flash/ch7/sec7_3/isomorphism.html`.

[25] *The Graph Isomorphism Algorithm*, [ogled 4. 5. 2012], dostopno na `http://www.dharwadker.org/tevet/isomorphism/`.

[26] *Graph Isomorphism*, [ogled 3. 5. 2012], dostopno na `http://en.wikipedia.org/wiki/Graph_isomorphism`.

[27] *Graph Isomorphism applied to Fingerprint Matching*, [ogled 4. 5. 2012], dostopno na `http://euler.mat.ufrgs.br/~trevisan/workgraph/regina.pdf`.

[28] C. Pich, *Applications of Multidimensional Scaling to Graph Drawing*, Doktorsko delo, univ. v Konstanzu, 2009.

[29] L. Arnold, *On the asymptotic distribution of the eigenvalues of random matrices*, Journal of Mathematical Analysis and Applications, **20**, (1967), 262-–268.

[30] A.L. Barabási, R. Albert, *Emergence of scaling in random networks*, Science **286**(5439) (1999), 509—512.

[31] U. Brandes, T. Erlebach, *Network Analysis*, Springer Verlag (2008), LNCS 3418.

[32] D. M. Cvetković, M. Doob, H. Sachs, *Spectra of Graphs*, Johann Ambrosius Barth Verlag (1995).

[33] D.J. de Solla Price, *A general theory of bibliometric and other cumulative advantage processes*, Journal of the American Society for Information Science **27** (1976), 292—306.

[34] P. Erdős, A. Rényi, *On the evolution of random graphs*, Publications of the Mathematical Institute of the Hungarian Academy of Sciences **5** (1960), 17–61.

[35] P. Erdős, A. Rényi, *"On Random Graphs I*, Publicationes Mathematicae **6** (1959), 290—297.

[36] P. Erdős, A. Rényi, *The Evolution of Random Graphs*, Magyar Tud. Akad. Mat. Kutató Int. Közl. **5** (1960), 17—61.

[37] I. Farkas, I. Derényi, A. L. Barabási, T. Vicsek, *Spectra of "real-world" graphs: Beyond the semicircle law*, Physical Review E, (64), (2001).

[38] Z. Füredi, J. Komlós, *The eigenvalues of random symmetric matrices*, Combinatorica, 1(**3**), (1981), 233-–241.

[39] F. Juhász, *On the spectrum of a random graph*, Colloquia Mathematica Societatis János Bolyai, **25**, (1978), 313—316.

[40] H.A. Simon, *On a Class of Skew Distribution Functions*, Biometrika **42**(3–4) (1955): 425—440.

[41] D.J. Watts, S.H. Strogatz, *Collective dynamics of 'small-world' networks*, Nature **393**(6684) (1998), 409—410.

[42] H. Wiener, *Structural determination of paraffin boiling points*, Journal of the American Chemical Society, **69**, (1947), 17—20.

[43] E. P. Wigner, *Characteristic vectors of bordered matrices with infinite dimensions*, Annals of Mathematics, **62**, (1955), 548—564.

[44] U. Yule, *A Mathematical Theory of Evolution Based on the Conclusions of Dr. J. C. Willis, F.R.S*, Publications of the Mathematical Institute of the Hungarian Academy of Sciences **88**(3) (1925), 433—436.

[45] D.M. Pennock etal. (2002) *Winners dont't take all*, PNAS, **99**/8, 5207-5211.

[46] Batagelj V., Brandes U. (2005): *Efficient Generation of Large Random Networks. Physical Review* E **71**, 036113, 1-5.

[47] Albert R., Barabasi A.L.: *Topology of evolving networks: local events and universality.* `http://xxx.lanl.gov/abs/cond-mat/0005085`

[48] V. Batagelj, A. Mrvar: *Pajek and Pajek-XXL Programs for Analysis and Visualization of Very Large Networks Reference Manual*, dostopno na `http://mrvar.fdv.uni-lj.si/pajek/pajekman.pdf`, junij 2012.

[49] A. Mrvar: prosojnice iz predmeta *Analiza socialnih omreĽlij*, dostopno na `http://mrvar.fdv.uni-lj.si/sola/info4/defaults.htm`, junij 2012

[50] de Nooy W., Mrvar A., Batagelj V.: *Exploratory Social Network Analysis with Pajek*, CUP, januar 2005.

[51] de Nooy W., *World trade in miscellaneous manufactures of metal, 1994*, dostopno na `http://vlado.fmf.uni-lj.si/pub/networks/data/esna/metalWT.htm`, junij 2012.

[52] Mark Jerrum. Large cliques elude the Metropolis process.*Randoms Structures and Algorithms*, 3(4):347-359, 1992.

[53] Robert E. Tarjan and Anthony E. Trojanowski. Finding a maximum independent set. SIAM *Journal on Computing*, 6(3):537-546, 1977.

[54] John Michael Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425-440, 1986.

[55] Richard Beigel. Finding maximum independent sets in sparse and general graphs. *In Proceedings of the 10th Annual* ACM-SIAM *Symposium on Discrete Algorithms* (SODA′99), pages 856-857. IEEE Computer Society Press, 1999.

[56] John W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23-28, 1965.

[57] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. *Thatcher, editors, Complexity of Computer Computations*, pages 85-103. Plenum Press, 1972.

[58] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251-280, 1990.

[59] Klaus Holzapfel, Sven Kosub, Moritz G. Maaß, Alexander Offtermatt-Souza, and Hanjo Taubig *A zero-one law for densities of higher order*, Manuscript, 2004.

[60] Bela Bollobas, *Extremal graph theory*, Academic Press, 1978.

[61] Gabriel A. Dirac, *Extensions of Turan's theorem on graphs*, Acta Mathematica Academiae Scientiarum Hungaricae, 14:417-422, 1963.

[62] Tamas Kovari, Vera T. Sos, and Pal Turan, *On a problem of Zarankiewicz*, Colloquium Mathematicum, 3:50-57, 1954.

[63] Jerrold R. Griggs, Miklos Simonovits, and George Rubin Thomas, *Extremal graphs with bounded densities of small subgraphs*, Journal of Graph Theory, 29(3):185-207, 1998.

[64] Jean-Claude Picard and Maurice Queyranne, *A network flow solution to some nonlinear 0-1 programming problems, with an application to graph theory*, Networks, 12:141-159, 1982.

[65] Andrew V. Goldberg, *Finding a maximum density subgraph*, Technical Report UCB/CSB/ 84/171, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 1984.

[66] Giorgio Gallo, Michail D. Grigoriadis, and Robert E. Tarjan, *A fast parametric maximum flow algorithm and applications*, SIAM Journal on Computing, 18(1):30-55, 1989.

[67] Andrew V. Goldberg and Robert E. Tarjan, *A new approach to the maximumflow problem*, Journal of the ACM, 35(4):921-940, 1988.

[68] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin, *Network Flows: Theory, Algorithms, and Applications* Prentice Hall, 1993.

[69] Moses Charikar, *Greedy approximation algorithms for finding dense components in a graph. In Proceedings of the 3rd International Workshop on Approximatin Algorithms for Combinatorial Optimization*, (APPROX'00) volume 1931 of Lecture Notes in Computer Science, 84-95. Springer-Verlag, 2000.

[70] Ravi Kannan and V. Vinay, *Analyzing the structure of large graphs*, Manuscript, 1999.

[71] Eugene L. Lawler, Jan Karel Lenstra, and Alexander H. G. Rinnooy Kan. *Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. SIAM Journal on Computing,* 9(3):558-565, 1980

[72] Emmanuel Loukakis and Konstantinos-Klaudius Tsouros. *A depth first search algorithm to generate the family of maximal independent sets of a graph lexicographically. Computing,* 27:249-266, 1981

[73] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. *On generating all maximal independent sets. Information Porcessing Letters,* 27(3):119-123, 1988

[74] Jon M. Kleinberg. *Authoritative sources in a hyperlinked environment. Journal of the ACM,* 46(5):604-632, 1999.

[75] Stephen B. Seidman. Network structure and minimum degree. *Social Networks,* 5:269-287, 1983.

[76] Stephen B. Seidman and Brian L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology,* 6:139-154, 1978.

[77] Stephen B. Seidman and Brian L. Foster. A note on the potential for genuine cross-fertilization between anthropology and mathematics. *Social Networks,* 172:65-72, 1978

[78] John W. Monn. On the diameter of a graph. *Michigan Mathematical Journal,* 12(3):349-351, 1965.

[79] Martin G. Everett. Graph theoretic blockings k-plexes and k-cutpoints. *Journal of Mathematical Sociology,* 9:75-84, 1982.

[80] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$- Completeness.* W.H. Freeman and Company, 1979.

[81] Vladimir Batagelj and Matjaž Zaveršnik. An $O(m)$ algorithm for cores decomposition of networks. Technical Report 798, IMFM Ljubljana, Ljubljana, 2002.

[82] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science,* 8:189-201, 1979.

[83] Stephen B. Seidman. Clique-like strucutres in directed networks. *Journal of Social and Biological Structures,* 3:43-54, 1980

[84] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing,* 6(3):505-517, 985.

[85] Uriel Feige, Guy Kortsarz, and David Peleg, *Algorithmica*, The dense k-subgraph problem (2001), 410-421.

[86] Stephen B. Seidman, *Clique-like structures in directed networks*, Journal of Social and Biological Structures (1980), 43-54.

[87] Uriel Feige and Michael A. Seltser, On the densest k-subgraph problem. Technical Report CS97-16, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel, 1997.

[88] Yuichi Asahiro, Refael Hassin, Kazuo Iwama, Complexity of finding dense subgraphs, *Discrete Applied Mathematics*, 121(1-3):15-26, 2002.

[89] Klaus Holzapfel, Sven Kosub, Moritz G. Maaß, and Hanjo Täubig, The complexity of detecting fixed-density clusters. In *Proceedings of the 5th Italian Conference on Algorithms and Complexity (CIAC'03)*, volume 2653 of *Lecture Notes in Computer Science*, 201-212, Springer-Verlag, 2003.

[90] David Harel and Yehuda Koren, On clustering using random walks, *Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'01)*, volume 2245 of *Lecture Notes in Computer Science*, 18-41, Springer-Verlag, 2001.

[91] G. Kishi, *On centrality functions of a graph*, Lecture Notes in Computer Science (108) (1980), 45–52.

[92] G. Sabidussi, *The centrality index of a graph*, Psychometrika **31** (1966), 581—603.

[93] G. Kishi, M. Takeuchi, *On centrality functions of a non-directed graph*, Proceedings of the 6th Colloquium on Microwave Communication (1978).

[94] A. N. Langville, C. D. Meyer, *Deeper inside PageRank*, Internet Mathematics **3**(1) (2004).

[95] A. Y. Ng, A. X. Zheng, M. I. Jordan, *Link analysis, eigenvectors and stability*, Proceedings of the senventeenth international joint conference on artificial intelligence (2001), 903–910.

[96] T. H. Haveliwala, S. D. Kamvar, *The second eigenvalue of the Google matrix*, Technical report, Stanford University (2003).

[97] M. Bianchini, M. Gori, F. Scarselli, *Inside PageRank*, ACM Transactions on Internet Technology (2004).

[98] R. Lempel, S. Moran, *Rank-stability and rank-similarity of linkbased web ranking algorithms in authority-connected graphs*, Information Retrieval, special issue on Advances in Mathematics/Formal Methods in Information Retrieval (2004).

[99] A. Borodin, G. O. Roberts, J. S. Rosenthal, P. Tsaparas, *Finding authorities and hubs from link structures on the world wide web*, Proceedings of the 10th International World Wide Web Conference (2001), 415–429.

[100] R. van den Brink, R. P. Gilles, *An axiomatic social power index for hierarchically structured populations of economic agents*, In Robert P. Gilles and Picter H.M. Ruys, editors, Imperfections and Behaviour in Economic Organizations, pages 279—318. Kluwer Academic Publishers Group, 1994.

[101] R. van den Brink, R. P. Gilles, *Measuring dominantion in directed networks*, Social Networks, 22(2):141–157, May 2000.

[102] B. Ruhnau, *Eigenvector-centrality – a node-centrality?*, Social Networks, 22:357–365, 2000.

[103] L. C. Freeman, *Centrality in social networks: Conceptual clarification I.*, Social Networks, 1:215–239, 1979.

[104] L. C. Freeman, *A set of measures of centrality based upon betweeness*, Sociometry, 40:35–41, 1977.

[105] O. R. Oellermann, *On the l-connectivity of a graph*, Graphs and Combinatorics, 3:285–291, 1987.

[106] B. Möller. *Zentralitäten in Graphen*, Diplomarbeit, Fachbereich Informatik und Informationswissenschaft, Universität Konstanz, July 2002.

[107] S. White, P. Smyth.*Algorithms for estimating relative importance in networks*, Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03), 2003.

[108] T. H. Haveliwala, *Sepandar D. Kamvar, and Glen Jeh. An analytical comparison of approaches to personalized PageRank*, Technical report, Stanford University, June 2003.

[109] T. H. Haveliwala. *Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search*, IEEE Transactions on Knowledge and Data Engineering, 15(4):784–796, 2003.

[110] G. Jeh, J. Widom. *Scaling personalized web search*, Proceedings of the 12th International World Wide Web Conference (WWW12), 271—279, Budapest, Hungary, 2003.

[111] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, G. H. Golub. *Exploiting the block structure of the web for computing PageRank*, Technical report, Stanford University, March 2003.

[112] C. J. Alpert, A. B. Kahng, *Recent Directions in Netlist Partitioning: A Survey*, Integration: The VLSI Journal **19(1-2)** (1995), 1–81.

[113] S. Arora, S. Rao, U. Vazirani, *Expander flows, geometric embeddings and graph partitioning*, In Proceedings of the 36th Annual ACM Symposium on the Theory of Computing (STOC'04), ACM Press (2004), 222–231.

[114] G. Ausiello, P. Crecenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, *Complexity and Approximation - Combinatorial Optimization Problems and Their Approximability Properties*, Springer-Verlag, 2nd edition (2002).

[115] S. Basu, M. Bilenko, R. J. Mooney, *Comparing and Unifying Search-Based and Similarity-Based Approaches to Semi-Supervised Clustering*, In Proceedings of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining Systems (2003), 42–49, dostopno na `http://www.cs.utexas.edu/~ml/papers/semi-icml-wkshp-03.pdf`, [ogled 20. 7. 2012].

[116] V. Batagelj, M. Zaveršnik, *An $O(m)$ Algorithm for Cores Decomposition of Networks* (2002), dostopno na `http://vlado.fmf.uni-lj.si/pub/networks/doc/cores/cores.pdf`, [ogled 2. 8. 2012].

[117] U. Brandes, M. Gaertler, D. Wagner, *Experiments on Graph Clustering Algorithms*, In Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03) (volume 2832 of Lecture Notes in Computer Science) (2003), 568–579.

[118] `http://en.wikipedia.org/wiki/Degeneracy_%28graph_theory%29`, [ogled 2. 8. 2012].

[119] J. J. C. Gallego, D. Rodriguez, M. A. Sicilia, M. G. Rubio, A. G. Crespo, *Software Project Effort Estimation Based on Multiple Parametric Models Generated Through Data Clustering*, Journal of Computer Science and Technology **22(3)**, 371–378, dostopno na `http://www.cc.uah.es/drg/jif/CuadradoEtAl_JCST07.pdf`, [ogled 21. 7. 2012].

[120] S. M. van Dongen, *Graph Clustering by Flow Simulation*, PhD thesis, University of Utrecht (2000).

[121] P. Drineas, A. M. Frieze, R. Kannan, S. Vempala, V. Vinay, *Clustering Large Graphs via the Singular Value Decomposition*, Machine Learning **56** (2004), 9–33.

[122] D. Eppstein, *Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs*, j-ea **5** (2000), 1–23.

[123] J. Handl, J. Knowles, *An Evolutionary Approach to Multiobjective Clustering*, IEEE Transactions on Evolutionary Computation (2005), dostopno na `http://dbkgroup.org/handl/HandlKnowles_TEC.pdf`, [ogled 10. 9. 2011].

[124] E. Hartuv, R. Shamir, *A Clustering Algorithm Based on Graph Connectivity*, Information Processing Letters **76(4-6)** (2000), 175–181.

[125] A. K. Jain, R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall (1988).

[126] J M. Kleinberg, *An Impossibility Theorem for Clustering*, Proceedings of 15th Conference: Neiral Information Processing Systems, Advances in Neural Information Processing Systems (2002).

[127] A. K. Jain, M. N. Murty, P. J. Flynn, *Data Clustering: a Review*, ACM Computing Surveys **31(3)** (1999), 264–323.

[128] G. L. Nemhause, L. A. Wolesy, *Integer and Combinatorial Optimization*, Wiley (1988).

[129] M. E. J. Newman, M. Girvan, *Findind and Evaluating Community Structure in Networks*, Physical Review E **69(2)** (2004).

[130] S. Paterlini, T. Minerva, *Evolutionary Cluster Analysis*, dostopno na `http://www.dep.unimore.it/materiali_discussione/0370.pdf`, [ogled 20. 7. 2012].

[131] S. B. Seidman, *Network Structure and Minimum Degree*, Social Networks **5** (1983), 269–287.

[132] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Springer-Verlag (2003).

[133] S. Vempala, R. Kannan, A. Vetta, *On Clustering - Good, Bad and Spectral*, In Proceedings of the 41st Annual IEEE Symposium and Fundations of Computer Science (FOCS'00) (2000), 367–378.

[134] `http://en.wikipedia.org/wiki/VLSI_Technology`, [ogled 21. 7. 2012].

[135] Karl Menger, *Zur allgemeinen Kurventheorie*, Fundamenta Mathematicae (1927), 10:96–115.

[136] Frank Harary and Yukihiro Kodama, *On the genus of an n–connected graph*, Fundamenta Mathematicae (1964), 54:7–13.

[137] David W. Matula, *The cohesive strength of graphs*, Iz The Many Facets of Graph Theory, Proc., volume **1**10 of Lecture Notes in Mathematics (1969), Springer–Verlag, 215– 221.

[138] László Lovász, *Connectivity in digraphs*, Journal of Combinatorial Theory Series B (1973), 15(2):174–177.

[139] Lester R. Ford, Jr. and Delbert R. Fulkerson, *Maximal flow through a network*, Canadian Journal of Mathematics (1956), 8:399–404.

[140] Hassler Whitney, *Congruent graphs and the connectivity of graphs*, American Journal of Mathematics (1931), 54:150–168.

[141] Lowell W. Beineke and Frank Harary, *The connectivity function of a graph*, Mathematika (1967), 14:197–202.

[142] Frank Harary, *The maximum connectivity of a graph*, Proceedings of the National Academy of Science of the United States of America (1962), 48(7):1142–1146.

[143] Gary Chartrand, *A graph–theoretic approach to a communications problem*, SIAM Journal on Applied Mathematics (1966), 14(5):778–781.

[144] Yefim Dinitz, Alexander V. Karzanov, and M. V. Lomonosov, *On the structure of the system of minimum edge cuts in a graph*, Iz A. A. Fridman, Studies in Discrete Optimization (1976), Nauka, 290–306.

[145] Alexander V. Karzanov and Eugeniy A. Timofeev, *Efficient algorithm for finding all minimal edge cuts of a nonoriented graph*, Cybernetics (1986), 22(2):156–162.

[146] Ralph E. Gomory and T.C. Hu, *Multi–terminal network flows*, Journal of SIAM (December 1961), 9(4):551–570.

[147] Dan Gusfield, *Very simple methods for all pairs network flow analysis*, SIAM Journal on Computing (1990), 19(1):143–155.

[148] Lisa Fleischer, *Building chain and cactus representations of all minimum cuts from Hao–Orlin in the same asymptotic run time*, Journal of Algorithms, (October 1999), 33(1):51–72.

[149] Robert E. Bixby, *The minimum number of edges and vertices in a graph with edge connectivity n and m n–bonds*, Networks, (1981),5:253–298.

[150] Shimon Even and Robert E. Tarjan, *Network flow and testing graph connectivity*, SIAM Journal on Computing, (December 1975), 4(4):507–518.

[151] Shimon Even, *Graph Algorithms*, (1979), Computer Science Press.

[152] Alexander V. Karzanov, *On finding maximum flows in networks with special structure and some applications*, Matematicheskie Voprosy Upravleniya Proizvodstvom, volume 5, (1973), pages 66–70, Moscow State University Press. (In Russian).

[153] Abdol-Hossein Esfahanian and S. Louis Hakimi, *On computing the connectivities of graphs and digraphs*, (1984), 14(2):355–366, Networks.

[154] David W. Matula, *Determining edge connectivity in O(nm)*, Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science (FOCS'87), (October 1987), pages 249–251.

[155] F. Lorrain, H. C. White, *Structural equivalence of individuals in social networks*, Journal of Mathematical Sociology, (1), (1971), 49–80.

[156] L. D. Sailer, *Structural equivalence: Meaning and definition, computation and application*, Social Networks, (1), (1978), 73–90.

[157] D. R. White, K. P. Reitz, *Graph and semigroup homomorphisms on networks of relations*, Social Networks, (5), (1983), 193–234.

[158] R. Paige, R. E. Tarjan, *Three partition refinement algorithms*, SIAM Journal on Computing, 16(6), (1987), 973–983.

[159] M. G. Everett, S. P. Borgatti, *Regular equivalence: General theory*, Journal of Mathematical Sociology, 18(1), (1994), 29–52.

[160] M. Marx, M. Masuch, *Regular equivalence and dynamic logic*, Social Networks, 25, (2003), 51–65.

[161] M. G. Everett, S. P. Borgatti, *Role colouring a graph*, Mathematical Social Sciences, 21, (1991), 183–188.

[162] U. Brandes, T. Erlebach, *Network analysis: methodological foundations*, (2005), 216–252.

[163] J. Fiala, D. Paulusma, *The Computational complexity of the role assigment problem*, (2003).

[164] M. G. Everett, S. P. Borgatti, *Two algorithms for computing regular equivalence*, Social Networks, 15(4), (1993), 361–376.

[165] Albert, R., Jeong, H., Barabasi, A.-L.: *Error and attack tolerance of complex networks*. Nature 406, 378–382 (2000)

[166] Barabasi, A.-L., Albert, R.: *Emergence of scaling in random networks*. Science 286(5439), 509–512 (1999)

[167] Bollobas, B., Riordan, O.M.: *Mathematical results on scale-free random graphs*. In: Handbook of Graphs and Networks, pp. 1–34. Springer, Heidelberg (2003)

[168] Clauset, A., Newman, M.E.J., Moore, C.: *Finding community structure in very large networks*. Physical Review E 70, 066111 (2004)

[169] Faloutsos, M., Faloutsos, P., Faloutsos, C.: *On power-law relationships of the internet topology*. Computer Communications Review 29, 251–262 (1999)

[170] Lehmann, K.A., Post, H.D., Kaufmann, M.: *Hybrid graphs as a framework for the small-world effect*. Physical Review E 73, 056108 (2006)

[171] Liljeros, F.: *Sexual networks in contemporary western societies*. Physica A 338, 238–245 (2004)

[172] Liljeros, F., Edling, C.R., Amaral, L.A.N., Stanley, H.E., Aberg, Y.: *The web of human sexual contacts*. Nature 411, 907–908 (2001)

[173] Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., Alon, U.: *Superfamilies of evolved and designed networks*. Science 303, 1538– 1542 (2004)

[174] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: *Network motifs: Simple building blocks of complex networks*. Science 298, 824–827 (2002)

[175] Mark, E.J.: *The structure of scientific collaboration networks*. Proceedings of the National Academy of Sciences 98(2), 404–409 (2001)

[176] Newman, M.E.J., Barabasi, A.-L., Watts, D.J. (eds.): *The Structure and Dynamics of Networks*. Princeton University Press, Princeton (2006)

[177] Song, C., Havlin, S., Makse, H.A.: *Origins of fractality in the growth of complex networks*. Nature 2, 275–281 (2006)

[178] Watts, D.J., Strogatz, S.H.: *Collective dynamics of 'small-world' networks*. Nature 393, 440–442 (1998)

[179] Zweig, K.A.: *On Local Behavior and Global Structures in the Evolution of Complex Networks*. Ph.D thesis, University of Tubingen, Wilhelm-Schickard-Institut fur Informatik (2007)

[180] Berners-Lee, T., Hall, W., Hendler, J.A., O'Hara, K., Shadbolt, N., Weitzner, D.J.: *A framework for web science*. Foundations and Trends in Web Science 1(1), 1–130 (2006)

[181] Andersen, R., Chung, F., Lu, L.: *Analyzing the small world phenomenon using a hybrid model with local network flow*. In: Leonardi, S. (ed.) WAW 2004. LNCS, vol. 3243, pp. 19–30. Springer, Heidelberg (2004)

[182] Chung, F., Lu, L.: *The small world phenomenon in hybrid power law graphs*. In: Ben-Naim, E., Frauenfelder, H., Toroczkai, Z. (eds.) Complex Networks, pp. 91–106 (2004)

[183] Jin Akiyama, Francis T. Boesch, Hiroshi Era, Frank Harary, Ralph Tindell, *The cohesiveness of a point of a graph*, Networks (1981), 11(1):65–68.

[184] Douglas Bauer, S. Louis Hakimi, Edward F. Schmeichel, *Recognizing tough graphs is NP-hard*, Discrete Applied Mathematics (1990), 28:191–195.

[185] Francis T. Boesch, R. Emerson Thomas. *On graphs of invulnerable communication nets*, IEEE Transactions on Circuit Theory (1970), CT-17.

[186] Vašek Chvátal, *Tough graphs and hamiltionian circuits*, Discrete Mathematics (1973), 5.

[187] Frank Harary, *Conditional connectivity*, Networks (1983), 13:347–357.

[188] Frank Harary, *General connectivity*, In Khee Meng Koh and Hian-Poh Yap, editors, Proceedings of the 1st Southeast Asian Graph Theory Colloquium, volume 1073 of Lecture Notes in Mathematics (1984), pages 83–92, Springer-Verlag.

[189] Francis T. Boesch, Frank Harary, and Jerald A. Kabell. Graphs as models of communication network vulnerability: Connectivity and persistence. *Networks*, 11:57-63, 1981.

[190] V. Krishnamoorthy, K. Thulasiraman, and M.N.S. Swamy. Incremental distance and diameter sequences of a graph: New measures of network performance. *IEEE Transactions on Computers*, 39(2):230-237, February 1990.

[191] Melvin Tainiter. Statistical theory of connectivity I:Basic definitions and properties. *Discrete Mathematics*, 13(4): 391-398, 1975.

[192] Melvin Tainiter. A new deterministic network reliability measure. *Networks*, 6(3):191-204,1976.

[193] Réka Albert, Hawoong Jeong, and Albert - László Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378-382, July 2000.

[194] Hawoong Jeong, Sean P. Mason, Albert - László Barabási, and Zoltan N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411, 2001. Brief communnications.

[195] Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vespiguani. Dynamical and correlation properties of the internet. *Physical Review Letters*, 87(258701), 2001.

[196] Petter Holme, Beom Jun Kim, Chang No Yoon, and Seung Kee Han. Attack vulnerability of complex networks. *Physical Review* E, 65(056109), 2002.

[197] Reuren Cohen, Keren Erez, Daniel ben Arraham, and Shlomo Havlin. Resilience of the internet to random breakdown. *Physical Review Letters*, 21(85): 4626-4628, November 2000.

[198] Duncan S. Callaway, Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Network robustness and fragility: Percolation on random graphs, *Physical Review Letters*, 25(85):5468-5471, December 2000.

[199] Honsuda Tangmunarunkit, Ramesh Goviduan, Sugih Jamin, Scott Shenker, and Walter Willinger. Network topologies, power laws, and hierarchy. *ACM SIGCOMM Computer communication Review*, 32(1): 76, 2002.

[200] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-completeness*. W.H.Freeman and company, 1979.

[201] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359-392, 1998..

[202] Christopher Palmer, Georgos Siganos, Michalis Falontsos, Christos Falontsos, and Phillip Gibbons. The connectivity and fault-tolerance of the internet topology. In *Workshop on Network-Related Data Management (NRDM 2001)*, 2001.

[203] Philipe Flajolet, Kostas P. Hatzis, Sotiris Nikoletseas, and Paul Spirakis. On the robustness of interconnections in random graphs : A symbolic approach. *Theoretical Computer Science*, 287(2):515-534, September 2002.

[204] Robert R. Boorstyn and Howard Frank. Large scale network topological optimization. *IEEE Transaction on Communications*, Com-25:29-37, 1977.

[205] André Pönitz and Peter Tittmann. Computing network reliability in graphs of restricted pathwidth. http://www.peter. htwm.de/publications/Reliability.ps, 2001.

[206] Arnon S. Rosenthal. *Computing Reliability of Complex Systems*. PhD thesis, University of California, 1974.

[207] Walid Najjar and Jean-Luc Gaudiot. Network resilience: A measure of network fault tolerance, *IEEE Transactions on Computers*, 39(2):174-181, February 1990.

[208] D. M. Cvetković, M. Doob in H. Sachs, *Spectra of Graphs*, Johann Ambrosius Barth Verlag, 1995.

[209] F. R. K. Chung, *Spectral Graph Theory*, CBMS Regional Conference Series in Mathematics, American Mathematical Society, 1997.

[210] C. Godsil in G. Royle, *Algebraic Graph Theory*, Graduate Texts in Mathematics, Springer-Verlag, 2001.

[211] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, 1965.

[212] U. Brandes in T. Erlebach, *Network Analysis: methodological foundations*, Springer, Berlin, 2005.

[213] N. Alon, *Eigenvalues and expanders*, Combinatorica, 6(2):83–96, 1986.

[214] F. R. K. Chung, V. Faber in T. A. Manteuffel, *An upper bound on the diameter of a graph from eigenvalues associated with its laplacian*, SIAM Journal on Discrete Mathematics, 7(3):443–457, 1994.

[215] R. Elsässer in B. Monien, *Load balancing of unit size tokens and expansion properties of graphs*, In Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'03), pages 266–273, 2003.

[216] M. Fiedler, *Algebraic connectivity of graphs*, Czechoslovak Mathematical Journal, 23(98):289–305, 1973.

[217] C. Godsil, *Tools from linear algebra*, Research report, University of Waterloo, 1989.

[218] W. H. Haemers, *Eigenvalue methods*, In Alexander Schrijver, editor, Packing and Covering in Combinatorics, pages 15–38, Mathematisch Centrum, 1979.

[219] B. Mohar, *Isoperimetric number of graphs*, Journal of Combinatorial Theory, Series B, 47(3):274–291, 1989.

[220] B. Mohar, *Eigenvalues, diameter and mean distance in graphs*, Graphs and Combinatorics, 7:53–64, 1991.

[221] B. Mohar, *The laplacian spectrum of graphs*, In Yousef Alavi, Gary Char-trand, Ortrud R. Oellermann, and Allen J.Schwenk, editors, Graph Theory, Combinatorics, and Applications, pages 871–898, Wiley, 1991.

[222] B. Mohar in S. Poljak, *Eigenvalues in combinatorial optimization*, In Richard A. Brualdi, Shmuel Friedland, and Victor Klee, editors, Combinatorial and Graph-Theoretical Problems in Linear Algebra, pages 107–151. Springer-Verlag, 1993.

[223] L. Lovász, *On the Shannon capacity of a graph*, IEEE Transactions on Information Theory, 25:1–7, 1979.

[224] D. B. West, *Introduction to Graph Theory*, Prentice Hall, 2nd edition, 2001.

[225] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Reading, MA: Addison-Wesley, 1990.

[226] R. L. Waldho, *The Matching Model for Routing Permutations on Graphs*, Annandale-on-Hudson, New York, 1997, [ogled 9. 9. 2012], dostopno na http://rwald.tripod.com/math/routing/mmrpg.pdf.

[227] J. Stoer in R. Bulirsch, *Introduction to Numerical Analysis*, Springer- Verlag, 1993.

[228] W. H. Press, S. A. Teukolsky, W. T. Vetterling in B. P. Flannery *Numerical Recipes in C*, Cambridge University Press, 1992.

[229] F. S. Acton, *Numerical Methods that Work*, Mathematical Association of America, 1990.

[230] J. A. Scott, *An Arnoldi code for computing selected eigenvalues of sparse real unsymmetric matrices*, ACM Transactions on Mathematical Software, 21:423–475, 1995.

[231] M. Bauer in O. Golinelli, *Random incidence matrices: moments of the spectral density*, Journal of Statistical Physics, 103:301–307, 2001. arXiv cond-mat/0007127.