UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – 1. stopnja

Teja Turk

Večkriterijska optimizacija z evolucijskimi algoritmi

Delo diplomskega seminarja

Mentor: izred. prof. dr. Riste Škrekovski
Somentorica: Vida Vukašinović, prof. mat.

Ljubljana, 2012

Kazalo

Večkriterijska optimizacija z evolucijskimi algoritmi

Povzetek

Vsak dan se srečujemo z večkriterijskimi optimizacijskimi problemi, kot je denimo, kateri avto kupiti, da dobimo za najmanj denarja čimboljšo kvaliteto. Težava je, da odgovor na to vprašanje velikokrat ne obstaja, saj ponavadi najboljši avto ni tudi najcenejši. V tem delu definiramo večkriterijski optimizacijski problem $(D, f, \min)$ in relacijo dominiranosti $\preceq$ na prostoru $f(D) \subseteq \mathbb{R}^m$. Prasliko množice minimalnih elementov delno urejene množice $(f(D), \preceq)$ proglasimo za rešitev večkriterijskega optimizacijskega problema. Poimenujemo jo množica Pareto optimalnih rešitev.

V nadaljevanju predstavimo evolucijske algoritme za reševanje zgornjih problemov. To so stohastični algoritmi polinomske časovne zahtevnosti. Vrnejo le pod-optimalne rešitve, vendar so kljub temu zelo uporabni, ker so veliko hitrejši od deterministčnih algoritmov za NP težke probleme. Uspe nam dokazati konvergenco k množici Pareto optimalnih rešitev algoritma VV pod nekaterimi pogoji.

Na koncu podrobno opišemo genetski algoritem NSGA II. Uporabimo ga za reševanje večkriterijskega problema trgovskega potnika. Delovanje algoritma ocenimo z izračunom metrike hipervolumna.

Multiobjective optimization using evolutionary algorithms

Abstract

In our daily life, we face many multiobjective optimization problems such as which car to buy to get best quality for as little money as possible. The problem is that the answer to this question usually does not exist since the best car is not always the cheapest one. In this paper we define multiobjective optimization problem $(D, f, \min)$ and dominance relation $\preceq$ on space $f(D) \subseteq \mathbb{R}^m$. The preimage of set of minimal elements of partially ordered set $(f(D), \preceq)$ is proclaimed to be the solution of the multiobjective optimization problem. We name it a set of Pareto optimal solutions.

Furthermore we present evolutionary algorithms for solving such problems. These are stochastic algorithms with polynomial time complexity. They return only suboptimal solutions but they are still very useful because they are much faster than deterministic algorithms for NP hard problems. We manage to prove convergence to the set of Pareto optimal solutions of algorithm VV under some conditions.

Finally we describe genetic algorithm NSGA II in details. We use it for solving multiobjective Travelling Salesman problem. We assess its performance by computing hypervolume indicator.

Math. Subj. Class. (2010): 90C59, 90C29, 60J20, 78M50, 78M32

Ključne besede: večkriterijska optimizacija, Pareto fronta, evolucijski algoritmi, markovske verige, konvergenca evolucijskih algoritmov, NSGA II, metrika hipervolumna

Keywords: multiobjective optimization, Pareto front, evolutionary algorithms, Markov chains, convergence of evolutionary algorithms, NSGA II, hypervolume indicator

1. Uvod

Danes se optimiziranju po več kriterijih hkrati skoraj ne moremo več izogniti. Probleme večkriterijske optimizacije najdemo tako v ekonomiji kot tudi v prometu, medicini, industriji, itd. Rešitev takega problema je lahko veliko, saj so lahko nekatere rešitve boljše po enem kriteriju, druge pa po drugem. Čim hitreje želimo poiskati čim več rešitev, da bi se lahko potem odločili, katera od teh rešitev nam najbolj ustreza. Po vzoru Darwinove teorije o evoluciji so v 40. letih 20. stoletja začeli razvijati evolucijske algoritme, ki posnemajo evolucijski razvoj živih bitij in njihovo težnjo k prilagajanju. Ker ti algoritmi kot rezultat vračajo množico rešitev in niso časovno zahtevni, so jih začeli uporabljati za reševanje večkriterijskih optimizacijskih problemov.

V delu diplomskega seminarja najprej podrobno predstavimo problem večkriterijske optimizacije, vpeljemo nove pojme, ki jih potrebujemo, da opišemo, kaj sploh so rešitve večkriterijskega optimizacijskega problema. Potem se osredotočimo na evolucijske algoritme kot možne algoritme za reševanje tovrstnih problemov. Ne pozabimo omeniti nekaterih težav, na katere pri tem naletimo. Eno od pomembnih vprašanj je seveda, če ti algoritmi sploh vrnejo rešitve večkriterijskega optimizacijskega problema. Stohastično konvergenco k rešitvi bomo dokazali na primeru enostavnega genetskega algoritma. Nazadnje predstavimo še praktičen problem: zastavimo si problem, opišemo algoritem za reševanje, rešimo problem in statistično ovrednotimo dobljene rešitve.

2. Večkriterijska optimizacija

2.1. **Večkriterijski optimizacijski problem.** V vsakdanjem življenju se pogosto srečujemo z optimiziranjem po več kriterijih. Želimo npr. povečati udobje bivanja, a hkrati minimizirati stroške, da bi lahko tudi kaj privarčevali. Pri tem moramo paziti tudi na nekatere omejitve, kot je denimo mesečni prihodek. Ti kriteriji so med sabo največkrat konfliktni, saj izboljšanje po enem kriteriju povzroči poslabšanje po drugem. Tako se hitro zgodi, da več zapravljamo za dobrine, ki pripomorejo k večjemu udobju, kar pa žal pomeni tudi, da povečujemo stroške in obratno.

Postavlja se nam vprašanje, kako vedeti, katera rešitev je najboljša, ker ponavadi le-te niso primerljive. Tako se ne moremo odločiti, ali je bolje poskrbeti za udobje in več zapravljati ali pa je bolje varčevati, četudi potem ne živimo tako udobno, kot bi želeli. Zadrego bomo rešili tako, da bomo iskali vse rešitve, za katere ne obstaja rešitev, ki bi bila boljša ali enaka po vseh kriterijih.

Večkriterijski optimizacijski problem je iskanje dopustne rešitve x, ki zadošča vsem omejitvam in optimizira vektorsko funkcijo

$$f = (f_1, f_2, \ldots, f_m),$$

katere elementi, $f_1, \ldots, f_m$, so kriterijske funkcije. Optimizirati vektorsko funkcijo pomeni poiskati tako rešitev, da so vse vrednosti kriterijskih funkcij sprejemljive in ne moremo najti boljše rešitve. Kriterijske funkcije so v splošnem lahko zvezne ali diskretne.

Omejitve so največkrat dane v obliki neenačb $g_i(x) \geq 0$, $i \in \{1, 2, \ldots, l\}$ in enačb $h_j(x) = 0$, $j \in \{1, 2, \ldots, p\}$. Omenimo, da lahko poljubno omejitev v obliki enačbe $h(x) = 0$ podamo v obliki dveh neenačb. Množica ničel funkcije h je namreč enaka

rešitvi sistema neenačb

$$\begin{aligned} h(x) &\geq 0 \\ -h(x) &\geq 0. \end{aligned}$$

Zato bomo vse omejitve podali kar v obliki neenačb $g_i(x) \geq 0$, $i \in \{1, 2, \ldots, k\}$.

Opomba 2.1. Vsako iskanje maksimuma funkcije f se da enostavno prevesti na minimiziranje funkcije, saj velja

$$\max_{x \in D} f(x) = -\min_{x \in D} -f(x).$$

V nadaljevanju se bomo tako brez škode izognili obravnavanju obeh možnosti in se bomo omejili le na iskanje minimumov.

Naj Ω označuje množico možnih rešitev problema, ki lahko vsebuje tudi takšne elemente, ki ne zadoščajo vsem omejitvam. Ta množica je lahko povsem abstraktna in je odvisna od narave problema, ki ga rešujemo. Če npr. iščemo najkrajšo pot v grafu, ki ustreza določenim zahtevam, tedaj Ω vsebuje vse možne poti v danem grafu.

Definicija 2.2. Splošni *večkriterijski (minimizacijski) optimizacijski problem* je poiskati tak dopusten element $x \in \Omega$, ki zadošča omejitvam

$$g_j(x) \geq 0,\ j \in \{1, 2, \ldots, k\}$$

ter minimizira vektorsko funkcijo $f \colon \Omega \to \mathbb{R}^m$,

$$f = (f_1, f_2, \ldots, f_m), \quad f_i \colon \Omega \to \mathbb{R},\ i \in \{1, 2, \ldots, m\},$$

tj. x minimizira vse kriterijske funkcije f_i.

Označimo z D *množico dopustnih rešitev*, tj. množico vseh sprejemljivih elementov, za katere je definirana vektorska funkcija f, ki jo minimiziramo. Torej

$$D := \{x \in \Omega;\ g_i(x) \geq 0 \text{ za vsak } i \in \{1, 2, \ldots k\}\},$$

večkriterijski optimizacijski problem iskanja minimuma te funkcije pa označimo kot $(D, f, \min)$. Preslikavo f bomo imenovali *kriterijska funkcija*.

Zgodi se lahko, da ta večkriterijski optimizacijski problem nima rešitve. Razloge za neobstoj rešitve lahko razdelimo v tri kategorije.

- Če je množica D prazna, očitno problem nima rešitve. Tedaj pravimo, da je problem *nedopusten*, v nasprotnem primeru pa je *dopusten*.
- Dopusten večkriterijski optimizacijski problem nima rešitve, če katera od kriterijskih funkcij na D ni navzdol omejena. Rečemo, da je problem *neomejen*. Težav z neomejenostjo zagotovo nimamo, če je D končna ali pa če je D kompaktna in so vse kriterijske funkcije zvezne.
- V resnici le v redkih primerih obstaja tak vektor, ki optimizira vektorsko funkcijo f. Običajno namreč nek vektor x optimizira kriterijsko funkcijo f_i, ne pa tudi f_j, $i \neq j$. Ta problem je specifičen za večkriterijski optimizacijski problem, saj na prva dva naletimo že pri klasični enokriterijski optimizaciji.

Nastopi tudi problem s primerjanjem rešitev, saj relacija $\leq$ ni sovisna v $\mathbb{R}^m$, če je $m > 1$.

Primer 2.3. Naj bo $D = [0, 2\pi] \times [0, 1]$ in $f \colon D \to \mathbb{R}^2$, $f(\varphi, r) = (f_1(\varphi, r), f_2(\varphi, r))$, kjer sta

$$\begin{aligned} f_1(\varphi, r) &= r \cos \varphi \\ f_2(\varphi, r) &= r \sin \varphi. \end{aligned}$$

Poskusimo poiskati rešitev problema $(D, f, \min)$.

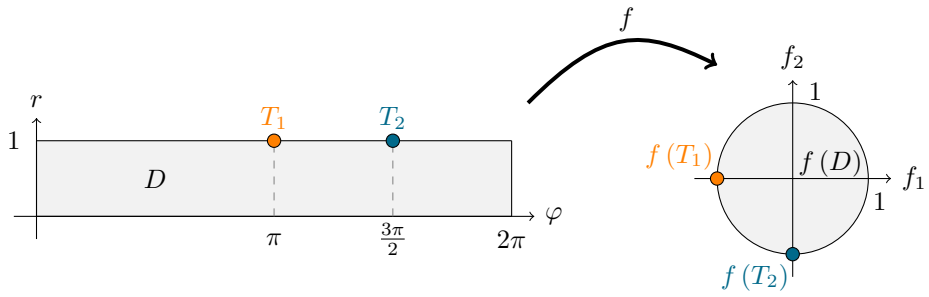
Očitno $D \neq \emptyset$, torej je problem dopusten. Ker je D kompaktna in sta f_1 ter f_2 zvezni, obe funkciji na D zagotovo dosežeta minimum. Poiščimo, v katerih točkah na D se to zgodi.

- Najmanjša vrednost f_1 je očitno -1, doseže pa jo le v točki
$$T_1 = (\varphi_1, r_1) = (\pi, 1).$$
- Podobno kot pri f_1 je najmanjša vrednost f_2 na D enaka -1, doseže pa jo samo v točki
$$T_2 = (\varphi_2, r_2) = \left(\frac{3\pi}{2}, 1\right).$$

Vendar pa $T_1 \neq T_2$, zato ne obstaja rešitev danega problema. Kljub temu se vprašajmo, katera od dobljenih točk pa je boljša.

$$\begin{aligned} f(T_1) &= f(\varphi_1, r_1) = (-1, 0) \\ f(T_2) &= f(\varphi_2, r_2) = (0, -1) \end{aligned}$$

Na sliki 1 se lepo vidi, da ne moremo povedati, katera od točk T_1 in T_2 je boljša,



Slika 1. Množica D s točkama T_1 in T_2 ter množica $f(D)$ s točkama $f(T_1)$ in $f(T_2)$.

saj je T_1 boljša po prvem kriteriju in slabša po drugem, T_2 pa obratno. To pa ne pomeni nič drugega kot, da smo kljub dvema kandidatoma za optimalno rešitev ostali praznih rok.

Zaradi omenjenih težav pri večkriterijski optimizaciji malce popustimo in iščemo vse vektorje x, za katere ne obstaja drug dopusten vektor y, pri katerem so vrednosti kriterijskih funkcij $f(y)$ manjše ali enake vrednostim kriterijskih funkcij $f(x)$.

2.2. **Dominiranost in Pareto fronta.** Na $\mathbb{R}^m$ uvedemo novo relacijo.

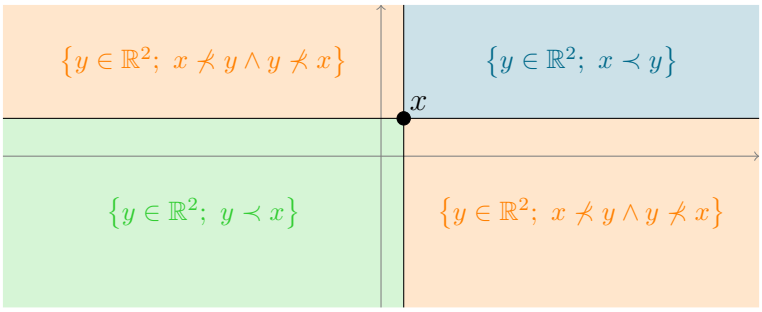
Definicija 2.4. Naj bosta $x = (x_1, x_2, \ldots, x_m), y = (y_1, y_2, \ldots, y_m) \in \mathbb{R}^m$. Pravimo, da x *strogo dominira* y ($x \prec y$), če

(1) nobena komponenta x ni večja od pripadajoče komponente y,

$$x_i \leq y_i \text{ za vsak } i \in \{1, 2, \ldots, m\},$$

in

(2) vsaj ena komponenta x je manjša od ustrezne komponente y, tj. obstaja i, pri katerem je $x_i < y_i$.



Slika 2. Grafični prikaz relacije dominiranosti v $\mathbb{R}^2$.

Očitno $x \not\prec x$ pri tako definirani relaciji. Zato relacijo popravimo, da postane refleksivna. Definiramo

$$x \preceq y \iff x \prec y \text{ ali } x = y$$

in pravimo, da x *dominira* y.

Trditev 2.5. $(\mathbb{R}^m, \preceq)$ *je delno urejena množica.*

Dokaz. Opazimo, da za $x = (x_1, x_2, \ldots, x_m)$ in $y = (y_1, y_2, \ldots, y_m)$ velja $x \preceq y$ natanko tedaj, ko velja $x_i \leq y_i$ za $i \in \{1, 2, \ldots, m\}$. Ker pa je $(\mathbb{R}, \leq)$ delno urejena množica, po komponentah sledi tudi delna urejenost $(\mathbb{R}^m, \preceq)$. □

Če je $X \subseteq \mathbb{R}^m$, je tudi $(X, \preceq)$ delno urejena množica. Zato lahko na njej iščemo minimalne elemente. $x^* \in X$ je minimalni element, če iz $x \in X$ in $x \preceq x^*$ sledi $x = x^*$. Spomnimo se, da lahko v delno urejeni množici obstaja več minimalnih elementov. Zato je smiselno vpeljati množico minimalnih elementov množice $(X, \preceq)$:

$$\mathcal{M}(X, \preceq) = \{x \in X;\ x \text{ je minimalni element } X\}.$$

Vrnimo se nazaj k našemu optimizacijskemu problemu $(D, f, \min)$. Z relacijo $\preceq$ lahko ocenjujemo le elemente množice $f(D) \subseteq \mathbb{R}^m$. Vendar pa bi raje primerjali elemente iz množice D, saj iščemo rešitev med temi elementi. Zato relacijo $\preceq$ s pomočjo preslikave f „prenesemo" na množico D.

Definicija 2.6. Naj bo $f\colon D \to \mathbb{R}^m$. Vpeljemo relacijo $\preceq_f$ na D, in sicer za poljubna $x, y \in D$ velja

$$x \preceq_f y \text{ natanko tedaj, ko } f(x) \preceq f(y).$$

Opomba 2.7. Včasih bomo malomarno rekli, da x dominira y, čeprav bo iz konteksta jasno razvidno, da $x \preceq_f y$.

Opomba 2.8. Relacija $\preceq_f$ ni nujno antisimetrična na množici D. $x \preceq_f y$ in $y \preceq_f x$ natanko tedaj, ko $f(x) \preceq f(y)$ in $f(y) \preceq f(x)$, od koder zaradi antisimetričnosti $\preceq$ sledi $f(x) = f(y)$. Če je f injektivna, dobimo $x = y$, sicer pa za poljubna $x, y \in D$ to ne velja. Refleksivnost in tranzitivnost $\preceq_f$ sta posledici delne urejenosti $(\mathbb{R}^m, \preceq)$ ne glede na to, ali je f injektivna ali ne. Zato je $(D, \preceq_f)$ delno urejena, če in samo če je f injektivna.

Opomba 2.9. Če $x \preceq_f y$ in $f(x) \prec f(y)$, pišemo $x \prec_f y$.

Pri večkriterijski optimizaciji bomo iskali *Pareto optimalne rešitve*.

Definicija 2.10. $x \in D$ je *Pareto optimalna rešitev* večkriterijskega optimizacijskega problema $(D, f, \min)$ natanko tedaj, ko je $f(x)$ minimalni element v $(f(D), \preceq)$.

V nadaljevanju bomo potrebovali tudi pojem *Pareto optimalne rešitve glede na* $A \subseteq D$. Rečemo, da je $x \in A$ Pareto optimalna glede na A natanko tedaj, ko je $f(x)$ minimalni element $(f(A), \preceq)$.

Opomba 2.11. Fiksirajmo x. Po definiciji je $f(x) \in \mathcal{M}(f(D), \preceq)$ natanko tedaj, ko iz $f(y) \preceq f(x)$ za $f(y) \in f(D)$ sledi $f(x) = f(y)$. Ekvivalentno lahko iz $y \preceq_f x$ za $y \in D$ sklepamo $f(x) = f(y)$. S tem smo dobili ekvivalentno definicijo Pareto optimalne rešitve.

Ker je minimalnih elementov lahko več, je lahko več tudi Pareto optimalnih rešitev. Iskali bomo *množico Pareto optimalnih rešitev*, ki jo označimo

$$\mathcal{P}^* = \{x \in D;\ x \text{ je Pareto optimalna rešitev}\}.$$

Opomba 2.12. S $\mathcal{P}_A^*$ označimo množico Pareto optimalnih rešitev glede na množico $A \subseteq D$.

Definicija 2.13. Za $(D, f, \min)$ je *Pareto fronta* množica minimalnih elementov $(f(D), \preceq)$.

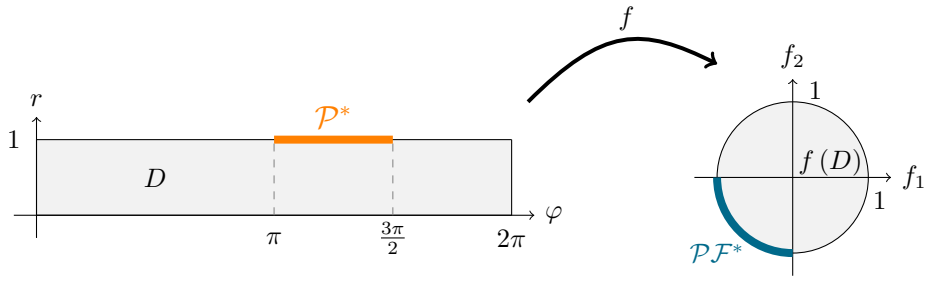
Pareto fronto bomo označili s $\mathcal{PF}^*$. Očitno je enaka sliki množice Pareto optimalnih rešitev $\mathcal{P}^*$, saj je

$$\begin{aligned} f(\mathcal{P}^*) &= f(\{x \in D;\ f(x) \in \mathcal{M}(f(D), \preceq)\}) \\ &= \{f(x) \in f(D);\ f(x) \in \mathcal{M}(f(D), \preceq)\} \\ &= \mathcal{M}(f(D), \preceq). \end{aligned}$$

Velja torej $f(\mathcal{P}^*) = \mathcal{PF}^* = \mathcal{M}(f(D), \preceq)$.

Naj bo $f(x) = (f_1(x), \ldots, f_m(x)) \in \mathcal{PF}^*$. Predpostavimo, da lahko $f(x)$ še zmanjšamo po enem kriteriju, ne da bi jo hkrati povečali po nekem drugem, npr. $f_i(x') < f_i(x)$ za nek i in $f_j(x') \leq f_j(x)$ za vse j. Od tod bi sledilo $f(x') \prec f(x)$, kar je v protislovju s tem, da je $f(x)$ minimalni element v $(f(D), \preceq)$, torej tega ne moremo storiti. Zato bo naš cilj pri reševanju večkriterijskega optimizacijskega problema najti množico Pareto optimalnih rešitev oz. Pareto fronto.

Primer 2.14. Naj bo $(D, f, \min)$ isti večkriterijski optimizacijski problem kot v primeru 2.3. S slike množice $f(D)$ se hitro vidi, da je



Slika 3. Pareto fronta $\mathcal{PF}^*$ in njena praslika $\mathcal{P}^*$.

$$\mathcal{M}(f(D), \preceq) = \mathcal{PF}^* = \left\{(x_1, x_2) \in \mathbb{R}^2;\ x_1^2 + x_2^2 = 1,\ x_1 \leq 0,\ x_2 \leq 0\right\}.$$

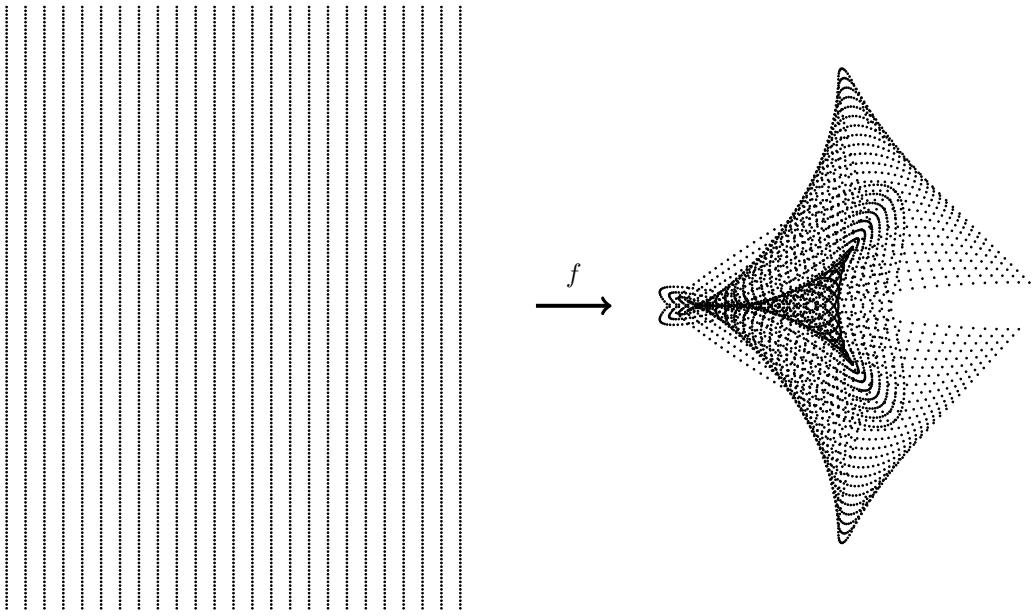
Da bi rešili problem, moramo poiskati še prasliko Pareto fronte. Tudi te v tem primeru ni težko najti, namreč

$$\mathcal{P}^* = \left[\pi, \frac{3\pi}{2}\right] \times \{1\}.$$

V zadnjem primeru ni bilo težko najti množice Pareto optimalnih rešitev. Vendar v splošnem to ni tako preprosto. Naj bo $D = \{-3, -2.75, \ldots, 3\} \times \{-4, -3.95, \ldots, 4\}$ in $f\colon D \to \mathbb{R}^2$ podana s predpisom

$$f(x, y) = \left(\frac{1}{2}(x - y)\sin(x + y), \frac{1}{2}(x + y)\cos(x - y)\right).$$

S slike množice $f(D)$ (slika 4) bi mogoče s težavo še lahko razbrali Pareto fronto problema $(D, f, \min)$, medtem ko množice Pareto optimalnih rešitev ne moremo enostavno najti. Če pa ima f več kot 3 kriterije, si že množice $f(D)$ ne moremo več grafično predstavljati, kaj šele da bi hitro našli Pareto fronto. Zato bomo v



Slika 4. Za množici D in $f(D)$ težko najdemo $\mathcal{P}^*$ in $\mathcal{PF}^*$.

nadaljevanju predstavili evolucijske algoritme kot orodje za reševanje večkriterijskih optimizacijskih problemov.

2.3. **Obstoj optimalnih rešitev in polnost množice.** Še preden nadaljujemo, se vprašajmo, kakšno povezavo ima množica Pareto optimalnih rešitev z množico rešitev večkriterijskega optimizacijskega problema. Velja naslednja trditev.

Trditev 2.15. *Rešitev večkriterijskega optimizacijskega problema* $(D, f, \min)$ *obstaja natanko tedaj, ko v množici* $(f(D), \preceq)$ *obstaja najmanjši element.*

Spomnimo se, da je $x \in X$ najmanjši element delno urejene množice $(X, \preceq)$, če je $x \preceq y$ za vsak $y \in X$.

Dokaz. ($\Rightarrow$)**:** Predpostavimo, da obstaja $x \in D$, ki reši problem $(D, f, \min)$. Tedaj x minimizira vse kriterijske funkcije $(f_1, f_2, \ldots, f_m)$, torej $f_i(x) \leq f_i(y)$ za vse $y \in D$ in vse $i \in \{1, 2, \ldots, m\}$. Po definiciji relacije $\preceq$ na množici

$f(D)$ pa to ne pomeni nič drugega, kot $f(x) \preceq f(y)$ za vse $y \in D$ oz. $f(x) \preceq z$ za vse $z \in f(D)$. Sledi, da je $f(x)$ najmanjši element.

($\Leftarrow$): Naj bo z najmanjši element množice $(f(D), \preceq)$. Ker $f^{-1}(z) \subseteq D$ ni prazna množica, izberimo iz nje poljuben element x. $f(x) = z$ je najmanjši element, zato je $f_i(x) \leq y_i$ za vse $i \in \{1, 2, \ldots, m\}$ in vsak $y \in f(D)$. Posledično x minimizira f po vseh kriterijih hkrati, kar pomeni, da je x rešitev problema $(D, f, \min)$.

□

Opomba 2.16. Preprosto je videti, da je z najmanjši element v $(f(D), \preceq)$ natanko tedaj, ko velja $x \preceq_f y$ za vsak $x \in f^{-1}(z)$ in vse $y \in D$.

Trditev 2.15 nam torej zagotavlja, da z iskanjem množice Pareto optimalnih rešitev popustimo v smislu, da bomo, kadar rešitve večkriterijskega optimizacijskega problema $(D, f, \min)$ obstajajo, poiskali te rešitve, v nasprotnem primeru pa takšne, ki so še vedno relativno dobre. Tako bo v nadaljevanju reševanje problema $(D, f, \min)$ pomenilo iskanje $\mathcal{P}^*$.

Pomembno vprašanje, ki se nam ob tem porodi, je, ali sploh obstaja optimalna rešitev. Lahko se namreč zgodi, da delno urejena množica $(f(D), \preceq)$ ne vsebuje nobenega minimalnega elementa.

Primer 2.17. Množica $A = \left\{\frac{1}{n};\ n \in \mathbb{N}\right\}$, opremljena z običajno relacijo $\leq$, ne vsebuje minimalnega elementa. Poljuben element $x \in A$ se da zapisati v obliki $\frac{1}{k}$, kjer je $k \in \mathbb{N}$. Ta element ne more biti minimalen, saj za $y = \frac{1}{k+1}$ velja $y \in A$ in $y \leq x$. Zato $(A, \leq)$ ne vsebuje minimalnega elementa.

Lema 2.18. *Vsaka neprazna končna delno urejena množica $(\mathcal{F}, \preceq)$ vsebuje minimalni element.*

Dokaz te leme zaenkrat izpustimo.

Definicija 2.19. Naj bo $(\mathcal{F}, \preceq)$ delno urejena množica. $\mathcal{M}(\mathcal{F}, \preceq)$ je *polna*, če za vsak $x \in \mathcal{F}$ obstaja $x^* \in \mathcal{M}(\mathcal{F}, \preceq)$, da je $x^* \preceq x$.

Polnost množice zahteva več kot le obstoj minimalnega elementa. Končna množica $\mathcal{F}$ z relacijo $\preceq$ je vedno polna, kar nam zagotavlja spodnja lema.

Lema 2.20. *Če je $(\mathcal{F}, \preceq)$ neprazna končna delno urejena množica, potem je množica minimalnih elementov $\mathcal{M}(\mathcal{F}, \preceq)$ polna.*

Dokaz. Lemo dokažemo tako, da za vsak $x \in \mathcal{F}$ poiščemo minimalni element x^*, za katerega je $x^* \preceq x$.

Naj bo $x \in \mathcal{F}$ poljuben. Če je $x \in \mathcal{M}(\mathcal{F}, \preceq)$, ni česa dokazovati, zato privzemimo, da x ni minimalen. Tedaj obstaja $x_1 \in \mathcal{F}$, za katerega je $x_1 \prec x$, saj bi v primeru neobstoja le-tega sledilo, da je x minimalni element. Če je $x_1 \in \mathcal{M}(\mathcal{F}, \preceq)$, smo končali, sicer postopek ponovimo za x_1 in najdemo $x_2 \in \mathcal{F}$, ki dominira x_1, itd. Ker je množica $\mathcal{F}$ končna, slej ali prej pridemo do minimalnega elementa x_n. Namreč, če $x_n \notin \mathcal{M}(\mathcal{F}, \preceq)$, bi moral obstajati manjši element, kar pa ne gre več. Pregledali smo namreč celotno množico, vendar iskanega nismo našli. Vprašati se moramo le še, ali je $x_n \preceq x$. To je res zaradi tranzitivnosti relacije $\preceq$, saj

$$x_n \preceq x_{n-1} \preceq \cdots \preceq x_2 \preceq x_1 \preceq x.$$

Torej je $(\mathcal{F}, \preceq)$ polna. □

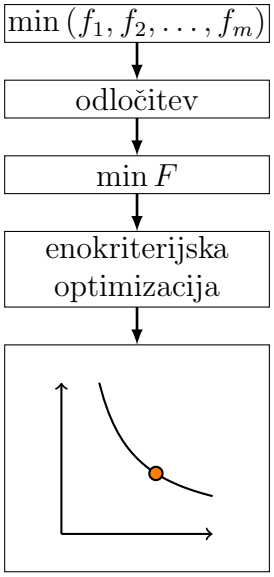
Opomba 2.21. Mimogrede smo dokazali, da vsaka neprazna končna delno urejena množica vsebuje minimalni element in s tem lemo 2.18.

3. Evolucijski algoritmi za večkriterijsko optimizacijo

3.1. **Metode za reševanje večkriterijskih optimizacijskih problemov.** Kot smo videli v prejšnjem razdelku, lahko $\mathcal{P}^*$ vsebuje več elementov, kar ne pomeni nič drugega kot to, da se bomo morali slej ali prej odločiti za eno od Pareto optimalnih rešitev. Glede na to, kdaj nastopi čas te odločitve, ločimo dva pristopa:

- prednostni pristop in
- idealni pristop.

3.1.1. *Prednostni pristop.* Katero rešitev bomo izbrali, lahko določimo že pred začetkom iskanja. Izbor ene izmed vseh Pareto optimalnih rešitev namreč odraža to, kateri kriterij je za nas pomembnejši. Denimo, da iščemo minimum preslikave $f = (f_1, f_2, \ldots, f_m)$ na območju D in naj bo $\mathcal{P}^*$ množica Pareto optimalnih rešitev problema $(D, f, \min)$. f poskusimo preoblikovati v funkcijo $F\colon D \to \mathbb{R}$ tako, da F doseže minimum v neki točki iz $\mathcal{P}^*$, ki upošteva naše preference kriterijev. S funkcijo F na neki način določimo smer iskanja minimuma funkcije f. S tem problem prevedemo na enokriterijski optimizacijski problem, ki ga rešimo z metodami enokriterijske optimizacije.



Slika 5. Shematski prikaz prednostnega pristopa.

Slabost tega pristopa je "izbiranje" rešitve vnaprej in določanje funkcije F. Pri tem se nam porodi vprašanje o obstoju takšne funkcije F oz. o tem, ali lahko za vsak $x \in \mathcal{P}^*$ najdemo funkcijo F_x z omenjenimi lastnostmi. Če bi torej hoteli dobiti več rešitev iz $\mathcal{P}^*$, bi morali večkrat pognati algoritem, ki reši enokriterijski problem $(D, F_x, \min)$.

V naslednjem primeru bomo predstavili najlažji in največkrat uporabljen način iskanja funkcije F.

Primer 3.1 (Metoda uteženih vsot)**.** Naj bo $(D, f, \min)$ izbrani večkriterijski optimizacijski problem, kjer je $f = (f_1, f_2, \ldots, f_m)$. Z utežmi $\vec{w} = (w_1, w_2, \ldots, w_m)$

ponazorimo pomembnost posameznih kriterijev. Običajno jih izberemo tako, da je $w_i \geq 0$ za vsak $i \in \{1, 2, \ldots, m\}$ in $\sum_{i=1}^{m} w_i = 1$, vendar to ni nujno. Funkcijo $F \colon D \to \mathbb{R}$ podamo s predpisom

$$F(x) = \sum_{i=1}^{m} w_i \cdot f_i(x) = \vec{w} \cdot f(x)$$

in iščemo tiste $x \in D$, pri katerih je vrednost $F(x)$ minimalna na D.

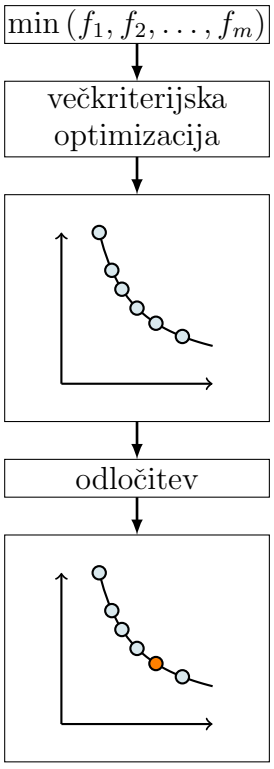
Enostavno je videti, da je vsak x, ki minimizira F, tudi Pareto optimalna rešitev prvotnega problema v primeru, ko je $w_i > 0$ za vse i. To dokažemo s protislovjem. Naj bo $x \in D$ točka, v kateri F doseže minimum. Pokazati želimo, da $x \in \mathcal{P}^*$. Denimo nasprotno, in sicer $x \notin \mathcal{P}^*$. Torej obstaja $y \in D$, da $f(y) \prec f(x)$. To pa pomeni, da je $f_i(y) < f_i(x)$ za nek $i \in \{1, 2, \ldots, m\}$ in $f_j(y) \leq f_j(x)$ za vse $j \neq i$. Tedaj je

$$F(y) = \sum_{i=1}^{m} w_i f_i(y) < \sum_{i=1}^{m} w_i f_i(x) = F(x),$$

torej je $F(y) < F(x)$ za nek $y \in D$, kar je v nasprotju s tem, da F v x doseže minimum, zato $x \in \mathcal{P}^*$.

Ali lahko z različnimi izbirami vektorja uteži $\vec{w}$ pridemo do vsake Pareto optimalne rešitve? V splošnem to ne gre, velja pa za probleme z določenimi lastnostmi.

3.1.2. *Idealni pristop.* Pri idealnem pristopu se izognemo izbiri rešitve vnaprej, saj se o njej odločamo po izvedenem iskanju. V enem zagonu algoritma iščemo več Pareto optimalnih rešitev hkrati, zato zadostuje en zagon algoritma. Vendar pa je večkriterijsko iskanje zahtevnejše od reševanja enokriterijskih problemov.



Slika 6. Shematični prikaz idealnega pristopa.

Narava evolucijskih algoritmov nam omogoča, da jih lahko med drugim uporabljamo tudi za reševanje tovrstnih problemov z idealnim pristopom.

3.2. **Evolucijski algoritmi.** Po Darwinovi evolucijski teoriji imajo v naravi večjo možnost preživetja močnejša bitja in tista, ki so bolje prilagojena okolju. Poleg tega imajo takšni osebki tudi večjo verjetnost uspešnega parjenja, z razmnoževanjem pa nastajajo nova bitja, ki nekatere lastnosti podedujejo od staršev. Dodatno gonilo razvoja pa so mutacije, ki omogočajo pojavitev lastnosti, ki jih starši še niso imeli. Če je ta lastnost dobra, bo mutirani potomec dobro pripravljen na boj za preživetje in razmnoževanje, zato se bo ta lastnost ohranila. Če pa je mutirana lastnost nezaželena, se osebek ne bo mogel prilagoditi okolju, zato se bo težko paril (in tako težko ustvaril sebi podobne slabe potomce) in bo slej ali prej izumrl, s tem pa bo izginila tudi njegova šibka dedna zasnova. Darwin je menil, da so zato živa bitja čedalje bolj prilagojena življenjskemu okolju.

Proces evolucije prenesemo na večkriterijsko optimizacijo. Intuitivno naj dopustne rešitve predstavljajo vse možne osebke, biti boljši pa naj pomeni imeti manjše vrednosti kriterijskih funkcij. Če izberemo še operatorje selekcije osebkov, križanja in mutacije ter začetno populacijo osebkov, lahko implementiramo algoritem, ki posnema proces naravne evolucije. Torej so za križanje z večjo verjetnostjo izbrani boljši osebki, ki imajo tudi več možnosti, da vstopijo v naslednjo generacijo. Če drži Darwinova teorija, da postajajo zaradi naravnega izbora, razmnoževanja in mutacij živa bitja vedno bolj prilagojena, zakaj ne bi z evolucijskim algoritmom dobili generacije rešitev, ki imajo čedalje nižje vrednosti kriterijskih funkcij in so zato poljubno blizu iskani množici Pareto optimalnih rešitev?

V nadaljevanju bomo podrobneje predstavili elemente evolucijskega algoritma.

3.2.1. *Predstavitev osebkov.* Dopustno rešitev $x \in D$ imenujemo tudi *fenotip*. Za izvedbo evolucijskega algoritma moramo dopustne rešitve izraziti s strukturo, ki bo omogočala križanje in mutacijo. Vsakemu fenotipu tako priredimo *genotip*, funkcijo prirejanja $e\colon D \to R$ pa imenujemo *funkcija kodiranja*. Tu R označuje prostor genotipov oz. $R = e(D)$. Dodatno zahtevamo, da vsak fenotip enolično določa genotip in obratno, da bomo lahko po končanem zagonu algoritma vedeli, katero dopustno rešitev nam predstavlja dobljeni genotip. Funkcija e je torej bijektivna, zato je smiselno vpeljati njen inverz $d\colon R \to D$, ki ga imenujemo *funkcija dekodiranja*. Za poljubna $x \in D$ in $y \in R$ je zato

$$(d \circ e)(x) = x \text{ in } (e \circ d)(y) = y.$$

Napogosteje se uporabljajo genotipi v obliki

- permutacij,
- vektorjev naravnih števil,
- vektorjev realnih števil,
- dvojiških nizov,
- naravnih števil, ...

Predstavitev je odvisna od problema, ki ga rešujemo in od izbranega križanja ter mutacije. Operatorje in predstavitev moramo izbrati preudarno, saj poljubna kombinacija le-teh ne zagotavlja konvergence algoritma.

3.2.2. *Generacija.* $P_t \in \mathbb{P}(R)$ je *generacija* ob času $t \in \mathbb{N}_0$, kjer je $\mathbb{P}(R)$ množica vseh multimnožic množice genotipov R oz. $\mathbb{P}(R) = \{P\colon R \to \mathbb{N}_0\}$. Z μ_t označimo

velikost populacije P_t in je enaka

$$\mu_t = \sum_{x \in R} P_t(x).$$

Če je $P_t(x) > 0$ za neskončno $x \in R$, definiramo $\mu_t = \infty$. Velikost μ_t se lahko s časom spreminja, največkrat pa ostane enaka tekom celotnega algoritma, torej $\mu_t = \mu_0$ za vsak $t \in \mathbb{N}_0$. Nekateri elementi se lahko v generaciji pojavijo večkrat. Težili bomo k čimvečji raznolikosti generacije. Pojem bomo razložili pozneje.

P_0 tako predstavlja začetno populacijo, ki jo dobimo z naključnim naborom μ_0 elementov množice R.

Omejili se bomo le na končne generacije, tj. $\mu_t < \infty$. Slej ali prej bomo namreč algoritem implementirali, delo z računalnikom pa nas omeji na končno podmnožico D, saj lahko numerično predstavimo le končno mnogo števil.

Opomba 3.2. V nadaljevanju bomo s pisanimi črkami označevali množico osebkov populacije, npr. $\mathcal{P}_t := \{x \in R;\ P_t(x) > 0\}$. Bistvena razlika med $\mathcal{P}_t$ in P_t je, da slednja vsebuje informacijo o kratnosti ponovitve osebkov v generaciji, medtem ko samo z zapisom $\mathcal{P}_t$ ne moremo vedeti, koliko kopij nekega osebka je v dani populaciji.

V nadaljevanju bomo uporabljali izraz osebek za genotip, rešitev pa za fenotip, saj bomo tako lažje potegnili vzporednice s procesom naravne evolucije.

3.2.3. *Funkcija uspešnosti.* Kateri osebki so boljši in kateri slabši? To nam pove *funkcija uspešnosti*, definirana na prostoru genotipov R. Če imamo problem zastavljen v obliki $(D, f, \min)$, kakovost rešitev opisuje kar funkcija f. Manjše kot so vrednosti kriterijskih funkcij, boljše so rešitve. Ker je domena funkcije f enaka D, vzamemo formalno za funkcijo uspešnosti optimizacijskega problema kompozitum $g = f \circ d$, ki slika iz R v $\mathbb{R}^m$.

Opomba 3.3. Množico Pareto optimalnih rešitev smo definirali kot podmnožico množice D. Včasih bomo uporabljali ekvivalentno definicijo, in sicer bo to podmnožica množice R. Tako za poljubno $A \subseteq R$ označimo

$$\mathcal{P}_A^* = \{x \in A \subseteq R;\ g(x) \in \mathcal{M}(g(A), \preceq)\}.$$

Katero od definicij bomo uporabljali, ne bo posebej navedeno, saj bo jasno razvidno, ali je $A \subseteq R$ ali $A \subseteq D$. Prav tako bomo tudi množico Pareto optimalnih rešitev glede na celoten R označili kar s $\mathcal{P}^*$.

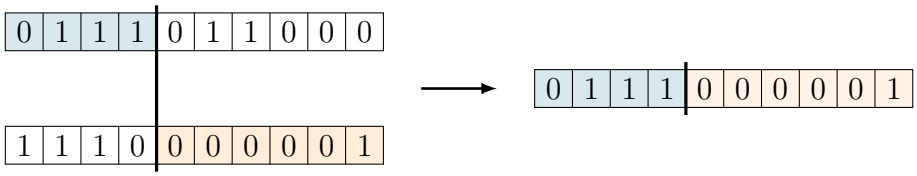
3.2.4. *Selekcija in križanje.* Selekcija in križanje skupaj tvorita proces, v katerem iz dane generacije P nastane generacija potomcev Q.

V dani generaciji so nekateri osebki boljši, zato jim damo večjo možnost, da postanejo starši. To pomeni, da so izbrani z večjo verjetnostjo. Vendar dopustimo, da imajo tudi slabši neko majhno (pozitivno) verjetnost, da bi bili izbrani, saj bi se nam sicer lahko zgodilo, da bi algoritem prehitro skonvergiral proti lokalnemu ekstremu, česar pa ne želimo. Večina operatorjev križanj izbira po dva in dva osebka, čeprav je z matematičnega (ne pa tudi iz naravnega) vidika možno večstarševsko križanje.

Sama izvedba selekcije staršev se razlikuje od algoritma do algoritma, najočitnejša pa je razlika med večkriterijsko in enokriterijsko optimizacijo. Pri slednji namreč že funkcija uspešnosti sama porodi linearno urejenost na množici osebkov, pri večkriterijski pa ne. Zato algoritmi za večkriterijsko optimizacijo največkrat najprej osebkom priredijo rang glede na nek kriterij, v drugi fazi pa linearno uredijo še osebke

istega ranga. Pri rangiranju moramo biti pozorni predvsem na to, da pri tem ne dajemo prednosti nobeni od rešitev (kot to naredimo pri reševanju tovrstnih problemov s prednostnim pristopom). Podrobneje bomo v zadnjem razdelku pogledali nedominirano razvrščanje kot primer večkriterijskega izbora staršev, zato bomo na tem mestu podrobnosti izpustili. Omenimo še, da je ponavadi prav selekcija staršev tista, ki determinira algoritem.

Ko enkrat poznamo starše, ustvarimo potomca, ki od vsakega od staršev podeduje nekatere lastnosti. Izbira načina ustvarjanja potomca je odvisna od predstavitve osebkov, ponavadi pa potekajo tako, da vnaprej določimo točke križanja. Domišljija pri kombiniranju staršev ne pozna meja, paziti moramo le na konvergenco algoritma, saj nam poljubna izbira operatorjev tega ne zagotavlja nujno. Najosnovnejši primer križanja je enotočkovno križanje dvojiških nizov dolžine n. Denimo, da je točka križanja na i-tem bitu. Potem potomca dobimo tako, da prvih i bitov potomca zapolni niz prvih i bitov prvega izmed staršev, preostalih $n-i$ bitov pa prepišemo iz zadnjih $n-i$ bitov drugega od staršev. Enotočkovno križanje lahko posplošimo na dvotočkovno, tritočkovno, itd. Slika 7 prikazuje omenjeni način križanja, ki zadošča za razumevanje osnovnega koncepta križanja. Zanimivejši primer bomo spoznali kar na primeru uporabe algoritmov.



Slika 7. Enotočkovno križanje v predstavitvi z binarnim nizom.

Formalno podamo operator selekcije in križanja z naslednjo definicijo.

Definicija 3.4. Naj bo χ_{c} množica parametrov in $\mathbb{X}_{\mathrm{c}}\colon \mathbb{N}_0 \to \chi_{\mathrm{c}}$ preslikava, ki v vsakem času določa parametre. *Selekcija in križanje* skupaj tvorita operator

$$\mathrm{c}\colon \mathbb{P}(R) \times \chi_{\mathrm{c}} \to \mathbb{P}(R).$$

Označimo s $\mathrm{c}^{(t)}$ operator selekcije in križanja ob času t, torej je $\mathrm{c}^{(t)}\colon \mathbb{P}(R) \to \mathbb{P}(R)$ podan s predpisom

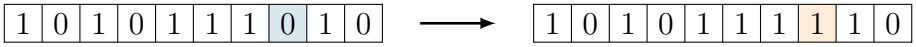
$$\mathrm{c}^{(t)}(P) = \mathrm{c}(P, \mathbb{X}_{\mathrm{c}}(t)).$$

Pri tem parametri zajemajo informacijo o izvedbi selekcije staršev in samem poteku križanja, preslikava $\mathbb{X}_{\mathrm{c}}$ pa je običajno konstantna. Zaradi prisotnosti naključnosti pri izbiri staršev in točk križanja je ta operator stohastičen.

3.2.5. *Mutacija.* Mutacija poteka na novonastalih potomcih Q, vendar ne prizadane vseh. Osebki, ki so ji podvrženi, so naključno izbrani, pri čemer so verjetnosti ponavadi zelo majhne in neodvisne od vrednosti funkcije uspešnosti potomcev. Izbrani osebki so naključno malo spremenjeni. Torej tudi mutacija poteka v dveh stopnjah, ki pa nista tako izraziti kot pri križanju. Bistveno pri mutaciji je, da ne ločuje osebkov po kakovosti; za ta operator so vse točke definicijskega območja enake. Za razliko od križanja mutacija ne ustvarja novih potomcev, ampak le spreminja že obstoječe.

Operator mutacije je precej preprostejši in tudi tu si lahko skoraj poljubno izbiramo način implementacije. Paziti moramo le, da pri tem ne izgubimo konvergence

algoritma. Posamezni načini izvedbe so vezani na določen tip predstavitve osebka. Enostaven primer mutacije na dvojiških nizih je, da naključen bit spremenimo iz 1 v 0 oz. obratno, kar ponazarja skica 8.



Slika 8. Bitna mutacija.

Verjetnosti, da je osebek izbran za mutacijo, vrsto mutacije in njene lastnosti podajajo parametri mutacije χ_{m}. Tako dobimo definicijo mutacije.

Definicija 3.5. Naj bo χ_{m} množica parametrov in $\mathbb{X}_{\mathrm{m}} \colon \mathbb{N}_0 \to \chi_{\mathrm{m}}$ preslikava, ki v vsakem diskretnem času določa parametre. *Mutacija* je operator

$$\mathrm{m} \colon \mathbb{P}(R) \times \chi_{\mathrm{m}} \to \mathbb{P}(R).$$

Podobno kot pri križanju označimo z $\mathrm{m}^{(t)}$ operator mutacije ob času t, kar pomeni, da je $\mathrm{m}^{(t)} \colon \mathbb{P}(R) \to \mathbb{P}(R)$ podan kot

$$\mathrm{m}^{(t)}(P) = \mathrm{m}(P, \mathbb{X}_{\mathrm{m}}(t)).$$

Tudi mutacija je stohastičen operator, ker so osebki naključno izbrani, prav tako pa sprememba poteka slučajno.

3.2.6. *Določitev nove generacije.* Določitev nove generacije je operator, ki izmed osebkov prejšnje generacije P in novonastalih potomcev S izbere naslednjo populacijo. Za razliko od izbire pri križanju ta operator nima tolikšnega pomena. Lahko je zelo podoben selekciji staršev, ni pa nujno. Največkrat je ta operator povsem deterministčen, s čimer mislimo, da je za dano množico potomcev in staršev naslednja generacija vnaprej dana. To se zgodi npr. v primeru, ko obdržimo izključno najboljše osebke (kakorkoli jih že linearno uredimo). Možen deterministčen pristop je tudi izločanje glede na starost. Vsekakor ta operator ne ustvarja novih osebkov, ampak le odstrani nekatere že obstoječe.

Formalno določitev nove generacije opisuje sledeča definicija.

Definicija 3.6. Naj bo χ_{s} množica parametrov in $\mathbb{X}_{\mathrm{s}} \colon \mathbb{N}_0 \to \chi_{\mathrm{s}}$ preslikava, ki v vsakem diskretnem času določa parametre. *Določitev nove generacije* je operator

$$\mathrm{s} \colon \mathbb{P}(R) \times \mathbb{P}(R) \times \chi_{\mathrm{s}} \to \mathbb{P}(R).$$

Podobno kot pri selekciji in križanju ter mutaciji označimo z $\mathrm{s}^{(t)}$ operator določitve nove generacije ob času t, kar pomeni, da je $\mathrm{s}^{(t)} \colon \mathbb{P}(R) \times \mathbb{P}(R) \to \mathbb{P}(R)$ podan kot

$$\mathrm{s}^{(t)}(P, Q) = \mathrm{s}(P, Q, \mathbb{X}_{\mathrm{s}}(t)).$$

3.2.7. *Zaustavitveni pogoj.* Zaustavitveni pogoj pove ob katerem dogodku končamo s pravkar vpeljanimi operatorji ter vrnemo rešitev. Nekateri možni kriteriji za zaustavitev glavne zanke so:

- evolucijski algoritem se izvaja že več kot T sekund;
- generirali smo več kot $G_{\max}$ generacij oz. se je glavna zanka izvedla že vsaj $G_{\max}$-krat;
- zadnjih k generacij je enakih ali zelo podobnih;
- trenutna rešitev je za nas sprejemljiva.

Seveda si lahko izmislimo tudi kakšen drug zaustavitveni pogoj, paziti moramo le, da je ta pogoj enkrat izpolnjen. Izbira izhoda iz glavne zanke je lahko ključnega pomena za kakovost rešitve. Če namreč izvedemo premajhno število ponovitev, je lahko dobljena rešitev zelo daleč od optimalne (vendar zaradi prisotnosti slučajnosti v algoritmu to ni nujno!).

3.2.8. *Psevdokoda.* Formalno smo definirali vse elemente evolucijskega algoritma, zato lahko podamo psevdokodo evolucijskega algoritma, ki jo podaja algoritem 1.

Algoritem 1 Evolucijski algoritem

procedure EVOLUCIJSKI_ALGORITEM(R, g, $\mathbb{X}_\mathrm{c}$, $\mathbb{X}_\mathrm{m}$, $\mathbb{X}_\mathrm{s}$, τ)
  $t = 0$
  P_0 ▷ začetna generacija
  while $\neg\tau$ **do**
    $Q_t = \mathrm{c}^{(t)}(P_t)$ ▷ selekcija in križanje
    $S_t = \mathrm{m}^{(t)}(Q_t)$ ▷ mutacija
    $P_{t+1} = \mathrm{s}^{(t)}(P_t, S_t)$ ▷ določitev nove generacije
    $t = t + 1$
  end while
  return P_t
end procedure

3.2.9. *Elitizem.* Elitistični pristop zagotavlja, da pri prehodu na novo generacijo ne izgubimo najboljših osebkov. To pomeni, da obdržimo Pareto optimalne rešitve glede na množico $\mathcal{P}_t \cup \mathcal{S}_t$. To lahko dosežemo na dva načina.

(1) Pri določitvi naslednje generacije najprej izberemo najboljše, potem pa s selekcijo nadaljujemo na preostalih osebkih. To pomeni, da če je $x^* \in \mathcal{P}^*_{\mathcal{P}_t \cup \mathcal{S}_t}$, je $x^* \in \mathcal{P}_{t+1}$.

(2) Druga možnost je tako imenovano *arhiviranje* najboljših osebkov. Na začetku ustvarimo množico $\mathcal{A}_0$, ki vsebuje vse Pareto optimalne rešitve glede na $\mathcal{P}_0$. Naprej nadaljujemo induktivno, in sicer množico $\mathcal{A}_{t+1}$ dobimo iz $\mathcal{A}_t$ na sledeč način:

$$\mathcal{A}_{t+1} = \mathcal{P}^*_{\mathcal{A}_t \cup \mathcal{S}_t},$$

tj. v $\mathcal{A}_{t+1}$ vzamemo najboljše osebke izmed tistih, ki smo jih do trenutka t arhivirali ali ustvarili. Množice $\mathcal{A}_t$ imenujemo *arhiv*. Problem, ki nastane pri tej možnosti je, da je zaporedje množic $(\mathcal{A}_t)_{t\in\mathbb{N}_0}$ naraščajoče. Če evolucijski algoritem konvergira proti $\mathcal{P}^*$, kar je naš cilj, potem je

$$\lim_{t\to\infty} \mathcal{A}_t = \mathcal{P}^*.$$

To pa pomeni, da so v primeru neskončne množice $\mathcal{P}^*$ množice $\mathcal{A}_t$ poljubno velike.

3.2.10. *Raznolikost.* Kot smo že omenili, želimo pri evolucijskih algoritmih doseči čim večjo raznolikost. Radi bi, da so rešitve, ki jih algoritem vrne, čim enakomerneje razporejene vzdolž celotne Pareto fronte. Zato se želimo izogniti zgoščenim rešitvam. Načinov, kako to dosežemo, je več, na kratko pa bomo omenili le dva izmed njih.

- *Delitev uspešnosti*

  Ideja tega pristopa je enakomerno razporediti osebke po prostoru D. Več rešitev kot se nahaja v neposredni bližini nekega osebka, manjšo verjetnost bo slednji imel za uspešno reprodukcijo.

  To dosežemo npr. tako, da vsakemu osebku $x \in \mathcal{P}$ povečamo vrednost funkcije uspešnosti sorazmerno glede na število osebkov, ki si delijo isto okolico.

$$g'(x) = g(x) \cdot \sum_{y \in R} P(y) \cdot \mathrm{sh}(d(x,y)),$$

  kjer je $d(x,y)$ razdalja med točkama x in y, sh pa funkcija, podana z

$$\mathrm{sh}(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{\mathrm{share}}}\right)^{\alpha}, & d \leq \sigma_{\mathrm{share}} \\ 0 & d > \sigma_{\mathrm{share}}. \end{cases}$$

  To pomeni, da več osebkov kot je bližje x, višja bo vrednost $g'(x)$, zato bo x manj zanimiv za izbor v naslednjo generacijo. Pri tem naletimo na določene težave glede določanja parametrov. Pametno moramo topološko opredeliti kroglo na prostoru D (oz. metriko) in izbrati polmer σ_{share}.
- *Kopičenje*

  Kopičenje podobno kot delitev uspešnosti teži k raznolikosti populacije. To poskuša doseči z odstranjevanjem večkratnih pojavitev elementa v populaciji in nadomeščanjem le-teh z novimi. Eden od množnih načinov je, da pri ustvarjanju novih potomcev, potomci in starši tekmujejo glede na medsebojne razdalje. Pri tem razdaljo, za razliko od prejšnjega pristopa, definiramo na prostoru $f(D)$. Podrobneje bomo to predstavili na primeru.

3.3. **Problemi pri izvajanju evolucijskih algoritmov.** Kot vsak algoritem, tudi evolucijski algoritmi niso brez pomanjkljivosti. Najbolj moteče je mogoče ravno to, da zaradi slučajnosti ne dobimo zagotovo optimalne rešitve, ampak optimalno rešitev dobimo le z neko verjetnostjo. Rešitvi, ki jo vrne evolucijski algoritem, pravimo *suboptimalna rešitev*. Ni nujno optimalna, vendar je za nas dovolj dobra, da jo sprejmemo v zameno za časovno učinkovitejši algoritem. Tipično so kombinatorični optimizacijski problemi zelo zahtevni. Problem trgovskega potnika zahteva vsaj $2^{\mathcal{O}(n)}$ operacij v primeru iskanja optimalne rešitve. Kar se tiče časovne zahtevnosti, so evolucijski algoritmi s polinomsko zahtevnostjo veliko primernejši.

3.3.1. *Kaznovanje neprimernih osebkov.* Največkrat zaradi enostavnosti algoritma križanje in mutacijo definiramo tako, da lahko pri tem ustvarimo nedopustno rešitev. Zato moramo paziti, da teh osebkov ne prenesemo v naslednjo populacijo, saj bi se sicer lahko zgodilo, da bi algoritem skonvergiral proti nedopustni rešitvi. Ta problem lahko razrešimo na enega izmed treh ustaljenih načinov.

(1) Najpogosteje uporabljena metoda je izvedba kaznovanja s t. i. *kazensko funkcijo*. Takšna funkcija poskrbi, da imajo nedopustne rešitve večje vrednosti funkcije uspešnosti. Zato so ti osebki slabši in so z manjšo verjetnostjo izbrani za razmnoževanje ter se težje prebijejo v naslednje generacije.
(2) Druga možnost je, da morebitne nedopustne osebke popravimo v dopustne.
(3) Zelo preprosto pa je nedopustne osebke zavreči.

3.3.2. *Prehitra konvergenca.* Če algoritem prehitro skonvergira, bi se nam lahko zgodilo, da bi zašli v lokalni minimum. Zato se temu poskušamo izogniti. To problematiko rešujemo z nastavitvami vseh parametrov treh glavnih operatorjev $\mathbb{X}_{\mathrm{c}}$, $\mathbb{X}_{\mathrm{m}}$ in $\mathbb{X}_{\mathrm{s}}$.

4. O konvergenci evolucijskih algoritmov

Predstavili smo osnovno delovanje evolucijskih algoritmov. Intuitivno se nam sicer lahko zdi, da omenjeni algoritmi vrnejo želeno. Algoritmov, še posebej časovno zahtevnejših, ne smemo nikoli na slepo prepustiti računalniku v izvajanje. Pomembno je, da se prepričamo o pravilnosti delovanja, kar med drugim zajema tudi to, da algoritem res vrne tisto, kar pričakujemo. Če npr. iščemo najkrajšo pot, naj algoritem kot izhodni podatek vrne iskano pot in ne katere druge.

Pri preverjanju pravilnosti delovanja evolucijskih algoritmov pride do izraza naključnost, ki nam bistveno oteži dokazovanje. Vsaka generacija zase je slučajna, zato tudi končna. Torej ne moremo z gotovostjo vedeti, kakšen bo rezultat, lahko pa ocenimo verjetnosti možnih izidov algoritma.

Težko je dokazati, da poljuben evolucijski algoritem vrne množico Pareto optimalnih rešitev (oz. njeno podmnožico). Mi se bomo omejili na posebej lepe algoritme, in sicer na takšne, pri katerih so generacije sicer slučajne, vendar je vsaka naslednja populacija odvisna le od neposredne predhodnice, ne pa tudi od ostalih. Zato bomo vpeljali markovske verige, katerih stanja bodo ponazarjala trenutne generacije.

Kot smo omenili že v prejšnjem razdelku, nas reševanje z računalnikom omeji na končne podmnožice D. Zato se bomo zadovoljili s konvergenco algoritmov s končno mnogo možnimi generacijami.

4.1. Markovske verige v diskretnem času.

Najprej vpeljimo markovsko verigo s številno množico stanj.

Definicija 4.1. Zaporedje slučajnih spremenljivk $(X_t)_{t\in\mathbb{N}_0}$ z zalogo vrednosti S je *markovska veriga v diskretnem času*, če velja

$$\mathrm{P}\left[X_{t+1} = j \mid X_t = i, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0\right] = \mathrm{P}\left[X_{t+1} = j \mid X_t = i\right]$$

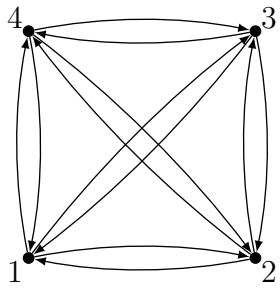
za poljuben $t \in \mathbb{N}_0$ in poljuben nabor stanj $s_0, s_1, \ldots, s_{t-1}, i, j \in S$.

Verjetnost $\mathrm{P}\left[X_{t+1} = j \mid X_t = i\right]$ imenujemo *prehodna verjetnost* iz stanja i v stanje j in je načelno odvisna od časa t. Če pa so vse prehodne verjetnosti neodvisne od t, potem pravimo, da je markovska veriga *homogena* in za $i, j \in S$ pišemo

$$p_{ij} := \mathrm{P}\left[X_{t+1} = j \mid X_t = i\right] \text{ za vsak } t \in \mathbb{N}_0.$$

V nadaljevanju se omejimo na homogene markovske verige s končno množico stanj S. Imenujemo jih homogene *končne markovske verige*. Brez izgube splošnosti stanja oštevilčimo z naravnimi števili. Verjetnosti p_{ij} lahko tedaj zapišemo v kvadratno *prehodno matriko* P, tako da se na (i, j)-tem mestu v matriki nahaja verjetnost p_{ij}.

Primer 4.2. Zamislimo si zajčka, ki skače po vozliščih polnega grafa na 4 vozliščih. V vsakem vozlišču z verjetnostjo $\frac{1}{3}$ skoči v enega od preostalih vozlišč, kot prikazuje slika 9. Vozlišča oštevilčimo s prvimi štirimi naravnimi števili. Naj X_n označuje vozlišče, v katerem je zajček po n skokih. X_n je slučajna spremenljivka, saj zajčkovo skakanje ni deterministično. Njegov položaj po n skokih je odvisen le od tega, iz katerega vozlišča je prišel v dano vozlišče, ne pa tudi od celotne poti, ki jo je do



Slika 9. Možni zajčkovi skoki.

takrat preskakal. Zato je zaporedje $(X_n)_{n\in\mathbb{N}_0}$ markovska veriga z množico stanj $S = \{1, 2, 3, 4\}$ in prehodnimi verjetnostmi

$$\mathrm{P}\left[X_{n+1} = j \mid X_n = i\right] = \frac{1}{3} \quad \text{in} \quad \mathrm{P}\left[X_{n+1} = i \mid X_n = i\right] = 0$$

za vsak $n \in \mathbb{N}_0$ in $j \neq i$. Opazimo, da so prehodne verjetnosti neodvisne od n, kar pomeni, da je vseeno, kdaj opazujemo zajčkovo skakanje oz. je markovska veriga homogena. Pripadajoča prehodna matrika je enaka

$$P = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}.$$

Zanima nas, s kakšno verjetnostjo se po določenem času n nahajamo v stanju $j \in \{1, 2, \ldots, m\}$. Označimo to verjetnost s $\pi_j^{(n)}$. Za $n \geq 1$ dobimo z uporabo formule o popolni verjetnosti za popoln sistem dogodkov $(\{X_{n-1} = i\})_{i=1}^m$

$$\begin{aligned} \pi_j^{(n)} &= \mathrm{P}\left[X_n = j\right] \\ &= \sum_{i=1}^{m} \mathrm{P}\left[X_{n-1} = i\right] \cdot \mathrm{P}\left[X_n = j \mid X_{n-1} = i\right] \\ &= \sum_{i=1}^{m} \pi_i^{(n-1)} p_{ij}. \end{aligned}$$

Če uvedemo še vektor

$$\pi^{(n)} = \begin{bmatrix} \pi_1^{(n)} & \pi_2^{(n)} & \cdots & \pi_m^{(n)} \end{bmatrix},$$

se zgornji račun v matrični obliki poenostavi v

$$\pi^{(n)} = \pi^{(n-1)} P.$$

Denimo, da poznamo začetno porazdelitev $\pi^{(0)}$ in prehodno matriko P homogene končne markovske verige. Potem natanko poznamo tudi poljubno verjetnost $\pi_j^{(n)}$, saj je

$$\pi^{(n)} = \pi^{(n-1)} P = \pi^{(n-2)} P^2 = \cdots = \pi^{(0)} P^n.$$

Za interpretacijo matrike P^n potrebujemo *prehodne verjetnosti iz stanja i v stanje j v k korakih*, definirane kot $p_{ij}(k) = \mathrm{P}\left[X_k = j \mid X_0 = i\right]$. Analogno označimo s $P^{(n)}$ matriko, katere (i, j)-ti element je $p_{ij}^{(n)}$. Za te verjetnosti velja naslednji izrek.

Izrek 4.3. *Naj bo* $(\Omega, \mathcal{F}, \mathrm{P})$ *dan verjetnostni prostor. Če je* X *homogena markovska veriga z množico stanj* $S = \{1, 2, \ldots, m\}$*, je*

$$p_{ij}^{(n)} = \sum_{k=1}^{m} p_{ik}^{(n-1)} p_{kj},$$

od koder sledi $P^{(n)} = P^n$*.*

Dokaz. Izrek bomo dokazali s pomočjo formule o popolni verjetnosti, vendar sistema popolnih dogodkov ne bomo vzeli iz σ-algebre $\mathcal{F}$, ampak iz njene σ-podalgebre

$$\mathcal{F}_{X_0^{-1}(i)} = \left\{ A \cap X_0^{-1}(i) \,;\, A \in \mathcal{F} \right\}.$$

Premislek, da je $\mathcal{F}_{X_0^{-1}(i)}$ res σ-algebra, bomo izpustili.

Opazimo, da je $(\{X_{n-1} = k \mid X_0 = i\})_{k=1}^m$ popoln sistem dogodkov v $\mathcal{F}_{X_0^{-1}(i)}$, ker je $X_0^{-1}(i) \in \mathcal{F}$ in $(\{X_{n-1} = k\})_{k=1}^m$ popoln sistem dogodkov v $\mathcal{F}$. Pri tem smo izpustili nekatere podrobnosti, ki pa za dokaz trditve niso bistvene. Uporabimo sedaj formulo o popolni verjetnosti v σ-algebri $\mathcal{F}_{X_0^{-1}(i)}$ in dobimo

$$\begin{aligned} p_{ij}^{(n)} &= \mathrm{P}\left[X_n = j \mid X_0 = i\right] \\ &= \sum_{k=1}^{m} \mathrm{P}\left[X_{n-1} = k \mid X_0 = i\right] \cdot \mathrm{P}\left[X_n = j \mid X_0 = i, X_{n-1} = k\right] \\ &= \sum_{k=1}^{m} p_{ik}^{(n-1)} \cdot \mathrm{P}\left[X_n = j \mid X_0 = i, X_{n-1} = k\right]. \end{aligned}$$

Vendar pa je $\mathrm{P}\left[X_n = j \mid X_0 = i, X_{n-1} = k\right] = \mathrm{P}\left[X_n = j \mid X_{n-1} = k\right]$ zaradi markovske lastnosti, od koder sledi $p_{ij}^{(n)} = \sum_{k=1}^m p_{ik}^{(n-1)} p_{kj}$, kar smo želeli dokazati.

V matrični obliki ima ta enakost obliko $P^{(n)} = P^{(n-1)}P$. Z induktivnim nadaljevanjem dobimo

$$P^{(n)} = P^{(n-1)}P = P^{(n-2)}P^2 = \cdots = P^{(1)}P^{n-1} = P^n,$$

ker je očitno $P^{(1)} = P$. Torej velja tudi $P^{(n)} = P^n$. □

Definicija 4.4. Nenegativna matrika $m \times n$ je *stohastična*, če je vsota vseh elementov vrstice enaka 1 za vsako vrstico, tj.

$$\sum_{j=1}^{n} p_{ij} = 1 \text{ za vsak } i = 1, 2, \ldots, m.$$

Ker je $(\{X_1 = k \mid X_0 = i\})_{k=1}^m$ popoln sistem dogodkov v $\mathcal{F}_{X_0^{-1}(i)}$, je

$$\sum_{k=1}^{m} p_{ik} = 1.$$

Poleg tega je verjetnost nenegativna preslikava na množici vseh dogodkov, zato je prehodna matrika markovske verige stohastična.

Definicija 4.5. Stohastična kvadratna matrika je *nerazcepna*, če za poljubni stanji $i, j \in S$ obstaja tak $k \in \mathbb{N}$, da je

$$p_{ij}^{(k)} > 0.$$

Nerazcepnost prehodne matrike markovske verige nam pove, da lahko iz vsakega stanja i dosežemo poljubno stanje j s pozitivno verjetnostjo v končnem času. Nerazcepnost bo torej ključna pri dokazovanju konvergence, saj nam bo zagotovila, da algoritem pregleda vse možne generacije s pozitivno verjetnostjo.

Pri dokazu konvergence se bomo opirali na naslednjo trditev.

Trditev 4.6. *Homogena markovska veriga s končno množico stanj in nerazcepno prehodno matriko obišče vsako stanje v končnem času z verjetnostjo* 1 *ne glede na začetno porazdelitev.*

Preden se lotimo dokazovanja, vpeljimo še čase prvih prehodov.

Definicija 4.7. Naj bo $(X_t)_{t\in\mathbb{N}_0}$ homogena markovska veriga in $X_0 = i$. *Čas prvega prehoda iz i v j* je slučajna spremenljivka z zalogo vrednosti $\mathbb{N} \cup \{\infty\}$, podana kot

$$T_{ij} = \min\{n \in \mathbb{N}; X_n = j \mid X_0 = i\}.$$

Čas prvega prehoda nam pove najmanjše število korakov, ki jih markovska veriga potrebuje, da pride prvič v stanje j pri pogoju, da začne v i.

Lema 4.8. *Če je X homogena markovska veriga s končno množico stanj in nerazcepno prehodno matriko, je*

$$\mathrm{P}[T_{ij} < \infty] = 1$$

za poljubni stanji i in j.

Opomba 4.9. Dogodek A se zgodi skoraj gotovo, če in samo če je $\mathrm{P}[A] = 1$.

Dokaz te leme bomo izpustili, saj zajema še nekatere druge rezultate iz teorije markovskih verig, ki jih mi nismo navedli. Najdemo ga npr. v [7, str. 304, 6. naloga]. Bomo pa z njeno pomočjo dokazali trditev 4.6.

Dokaz trditve 4.6. Računajmo verjetnost

$$\begin{aligned} p &= \mathrm{P}[X \text{ obišče vsa stanja v končnem času} \mid X_0 = i] \\ &= 1 - \mathrm{P}[\text{obstaja stanje, ki ga } X \text{ ne obišče v končnem času} \mid X_0 = i] \\ &= 1 - \mathrm{P}[T_{i,1} = \infty \cup T_{i,2} = \infty \cup \ldots \cup T_{i,m} = \infty]. \end{aligned}$$

Zaradi monotonosti in števne aditivnosti verjetnostne preslikave ob upoštevanju leme 4.8 velja

$$\begin{aligned} q &= \mathrm{P}[T_{i,1} = \infty \cup T_{i,2} = \infty \cup \ldots \cup T_{i,m} = \infty] \\ &\leq \mathrm{P}[T_{i,1} = \infty] + \mathrm{P}[T_{i,2} = \infty] + \ldots + \mathrm{P}[T_{i,m} = \infty] \\ &= 0 + 0 + \ldots + 0 \\ &= 0. \end{aligned}$$

Ker je verjetnost nenegativna, je $q = 0$ in $p = 1 - q = 1$. Rezultat je neodvisen od i, kar pomeni, da X skoraj gotovo obišče vsa stanja v končnem času ne glede na začetno porazdelitev. □

4.2. **Način konvergence evolucijskih algoritmov.** Preden se lotimo obravnave novih pojmov, povzemimo vse oznake in predpostavke, ki jih bomo uporabljali do konca tega razdelka.

- Rešujemo optimizacijski problem $(f, D, \min)$. Nadalje smo se omejili na končne množice dopustnih rešitev D, saj nam tako narekuje računalnik, s katerim algoritem implementiramo.

- $R = e(D)$ je prostor genotipov, e pa funkcija kodiranja, ki je bijektivna. Njen inverz je funkcija d. $g = f \circ d$ je funkcija uspešnosti, definirana na R.
- $\mathcal{PF}^* = \mathcal{M}(f(D), \preceq)$

Če hočemo dokazati konvergenco, moramo najprej pojem dobro opredeliti. Še prej pa potrebujemo metriko na prostoru generacij, saj je to osnovna enota evolucijskih algoritmov in tudi končen rezultat. V tem razdelku pri obravnavi konvergence ne bo pomembno, ali se nek osebek lahko v populaciji P_t pojavi več kot enkrat. Opazovali bomo le množico vseh osebkov, ki se pojavijo v trenutni populaciji (v prejšnjem razdelku smo jo označili s $\mathcal{P}_t$) in arhiv $\mathcal{A}_t$. Zato je dovolj izbrati primerno razdaljo na potenčni množici množice R.

Sledeča trditev definira metriko na postoru 2^R.

Trditev 4.10. *Naj bo R končna množica in A, B poljubni podmnožici. Potem je s predpisom*

$$d(A, B) = |A \cup B| - |A \cap B|$$

podana metrika na potenčni množici množice R.

Dokaz. Dokazali bomo, da je d enaka vektorski 1-normi, od koder bo takoj sledilo, da je d metrika.

Naj bo $R = \{r_1, r_2, \ldots, r_n\}$ in $\vec{a} = (a_1, a_2, \ldots, a_n)$ incidenčni vektor množice A, tj.

$$a_i = \begin{cases} 1; & r_i \in A \\ 0; & r_i \notin A. \end{cases}$$

Podobno naj bo $\vec{b}$ incidenčni vektor množice B. Enostavno je videti, da velja

$$|A| = \sum_{i=1}^{n} a_i \quad \text{in} \quad |B| = \sum_{i=1}^{n} b_i.$$

Če je $r_i \in A \cap B$, potem je $a_i b_i = 1$. Če pa r_i ni vsebovan v kateri od množic A in B, je $a_i b_i = 0$. Zato je lahko moči množic $|A \cap B|$ in $|A \cup B|$ izrazimo kot

$$\begin{aligned} |A \cap B| &= \sum_{i=1}^{n} a_i b_i \\ |A \cup B| &= |A| + |B| - |A \cap B| \\ &= \sum_{i=1}^{n} (a_i + b_i - a_i b_i). \end{aligned}$$

Če upoštevamo pravkar izračunano in uporabimo še $a_i, b_i \in \{0, 1\}$, končno sledi

$$\begin{aligned} d(A, B) &= \sum_{i=1}^{n} (a_i + b_i - a_i b_i) - \sum_{i=1}^{n} a_i b_i \\ &= \sum_{i=1}^{n} (a_i - 2a_i b_i + b_i) \\ &= \sum_{i=1}^{n} |a_i - b_i| \\ &= \|a - b\|_1. \end{aligned}$$

□

Ta metrika dobro odraža podobnost dveh množic. Če se namreč razlikujeta v skupno k elementih in je k zelo majhen, sta A in B skoraj enaki, razdalja med njima pa blizu 0.

Definicija 4.11. Naj bo $\mathcal{P}_t$ množica vseh osebkov populacije P_t nekega evolucijskega algoritma ob času $t \geq 0$ in $F_t = g(\mathcal{P}_t)$ pripadajoča slika funkcije uspešnosti. Evolucijski algoritem *konvergira k celotni Pareto fronti z verjetnostjo* 1, če je

$$\mathrm{P}\left[\lim_{t\to\infty} d(F_t, \mathcal{PF}^*) = 0\right] = 1.$$

4.3. Izbrani algoritem in njegova konvergenca.

4.3.1. *Osnovni algoritem VV.* To je naosnovnejši algoritem, na katerem se da lepo preveriti konvergenco, sicer pa se v praksi skoraj ne uporablja, še posebej če množica Pareto optimalnih rešitev vsebuje ogromno elementov. Psevdokodo podaja algoritem 2.

Algoritem 2 Osnovni algoritem VV

```
1: procedure VV(R, g, 𝕏c, 𝕏m, 𝕏s,τ)
2:     P_0 ∈ 2^R poljubna                          ▷ začetna generacija
3:     A_0 = P*_{P_0}
4:     t = 0
5:     while ¬τ do
6:         Q_t = c^(t)(P_t)                        ▷ selekcija in križanje
7:         S_t = m^(t)(Q_t)                        ▷ mutacija
8:         P_{t+1} = s^(t)(P_t, S_t)               ▷ določitev nove generacije
9:         A_{t+1} = P*_{A_t ∪ P_{t+1}}            ▷ arhiviranje
10:        t = t + 1
11:    end while
12:    return A_t
13: end procedure
```

Trditev 4.12 (Konvergenca algoritma VV)**.** *Če je zaporedje* $(\mathcal{P}_t)_{t\geq 0}$ *homogena končna markovska veriga z nerazcepno prehodno matriko, potem je skoraj gotovo* $\lim_{t\to\infty} d(g(\mathcal{A}_t), \mathcal{PF}^*) = 0$.

Opomba 4.13. Konstantne preslikave $\mathbb{X}_\mathrm{c}$, $\mathbb{X}_\mathrm{m}$ in $\mathbb{X}_\mathrm{s}$ so potreben pogoj, da je $(\mathcal{P}_t)_{t\geq 0}$ homogena končna markovska veriga z nerazcepno prehodno matriko. To nam zagotovi le homogenost in markovsko lastnost, za nerazcepnost pa moramo paziti pri izbiri prehodnih verjetnosti. To ponavadi dosežemo s popravljanjem operatorja mutacije.

Preden dokažemo veljavnost trditve, preverimo veljavnost dveh pomožnih lem, ki nam bosta prišli prav pri dokazovanju tega izreka in hkrati dokaz naredili preglednejši.

Lema 4.14. *Za vsak* $x \in \mathcal{P}^*$ *in poljubno neprazno podmnožico* $Q \subseteq R$*, za katero je* $x \in Q$*, velja*

$$g(x) \in \mathcal{M}(g(Q), \preceq) \quad \textit{in} \quad x \in \mathcal{P}^*_Q.$$

Dokaz. Ker je $g(x) \in g(Q)$ in je $\mathcal{M}(g(Q), \preceq)$ polna zaradi končnosti ter neprazno-sti množice Q, obstaja $y \in \mathcal{M}(g(Q), \preceq)$, ki dominira $g(x)$, tj. $y \preceq g(x)$. Ampak ker je $g(x) \in \mathcal{M}(g(R), \preceq)$, je po definiciji minimalnega elementa $y = g(x)$, od koder sledi $g(x) \in \mathcal{M}(g(Q), \preceq)$. Torej je tudi $x \in \mathcal{P}_Q^*$. □

Lema 4.15. *Naj bo $x \in \mathcal{P}^*$. Če je x ob nekem času $t_0 \geq 0$ vsebovan v $\mathcal{A}_{t_0}$, ostane x v $\mathcal{A}_t$ za vse nadaljnje iteracije $t \geq t_0$.*

Dokaz. Najprej dokažemo, da za poljuben $x \in \mathcal{P}^* \cap \mathcal{A}_{t_0}$ sledi $x \in \mathcal{A}_{t_0+1}$. Veljavnost za čase večje od $t_0 + 1$ bo sledila induktivno.

Naj bo $x \in \mathcal{P}^* \cap \mathcal{A}_{t_0}$ poljuben. Tedaj je $x \in \mathcal{P}^* \cap (\mathcal{A}_{t_0} \cup \mathcal{P}_{t_0+1})$, zato je po lemi 4.14 $x \in \mathcal{P}^*_{\mathcal{A}_{t_0} \cup \mathcal{P}_{t_0+1}} = \mathcal{A}_{t_0+1}$, kar smo želeli dokazati. V resnici je $x \in \mathcal{P}^* \cap \mathcal{A}_{t_0+1}$. Če v zgornjem premisleku vzamemo t_0+1 namesto t_0, dobimo $x \in \mathcal{A}_{t_0+2}$. S ponavljanjem tega razmisleka za različne čase večje od t_0 tako induktivno pokažemo vsebovanost elementa x v $\mathcal{A}_{t_0+n}$ za vsak $n \in \mathbb{N}_0$. □

Dokaz trditve 4.12. Dokazali bomo, da $\mathcal{A}_t \to \mathcal{P}^*$ skoraj gotovo, ko gre $t \to \infty$, od koder sledi $g(\mathcal{A}_t) \to \mathcal{PF}^*$ oz. $d(\mathcal{A}_t, \mathcal{PF}^*) \to 0$ z verjetnostjo 1, ko pošljemo t čez vse meje. Najprej se bomo prepričali, da vsaka Pareto optimalna rešitev vstopi v nek $\mathcal{A}_t$, kjer tudi ostane. Po drugi strani pa bomo videli, da Pareto optimalne rešitve izrinejo vse druge rešitve iz arhiva. Torej v $\mathcal{A}_t$ po dolgo časa ostanejo le želeni osebki.

Če se v trenutni generaciji P_t pojavi nek optimalen osebek, tj. $x \in \mathcal{P}^*$, potem je po lemi 4.14 $x \in \mathcal{A}_t$, saj je $x \in \mathcal{P}^* \cap \mathcal{P}_t$, zato tudi $x \in \mathcal{P}^* \cap (\mathcal{A}_{t-1} \cup \mathcal{P}_t)$. Po drugi pomožni lemi 4.15, je $x \in \mathcal{A}_\tau$ za vsak $\tau \geq t$. S tem smo pokazali, kako optimalna rešitev iz neke generacije vstopi v arhiv, kjer tudi ostane. Razmisliti moramo še, da se vsaka Pareto optimalna rešitev pojavi v kateri izmed populacij. Po trditvi 4.6 naš osnovni algoritem VV v končnem času preteče vse možne generacije in zato tudi vse osebke z verjetnostjo 1. Torej je skoraj gotovo vsak Pareto optimalni osebek vsebovan v neki populaciji P_t (torej tudi v $\mathcal{A}_t$) za nek $t < \infty$.

Preveriti moramo še, da po dolgo časa skoraj gotovo ni v arhivu rešitve, ki ni Pareto optimalna. Naj bo $t_0 \geq 0$ dovolj pozen čas, da $\mathcal{P}^* \subseteq \mathcal{A}_{t_0}$ (seveda le z verjetnostjo 1). Takšen čas skoraj gotovo obstaja po pravkar dokazanem, če upoštevamo hkrati še trditev 4.6. Denimo, da obstaja $y \in \mathcal{A}_{t_0} \setminus \mathcal{P}^*$. Zaradi polnosti $\mathcal{PF}^*$ obstaja $z \in \mathcal{PF}^* \cap f(\mathcal{A}_{t_0})$, ki je boljši od $f(y)$ oz. $z \preceq f(y)$. Zato $f(y)$ ne more biti minimalni element v $(\mathcal{A}_{t_0-1} \cup \mathcal{P}_{t_0}, \preceq)$ in posledično $y \notin \mathcal{A}_{t_0}$, kar je v protislovju s predpostavko.

Torej je

$$\mathrm{P}\left[\lim_{t\to\infty} \mathcal{A}_t = \mathcal{P}^*\right] = 1.$$

Po definiciji metrike d sledi še

$$\mathrm{P}\left[\lim_{t\to\infty} d(\mathcal{A}_t, \mathcal{P}^*) = 0\right] = 1 \quad \text{in} \quad \mathrm{P}\left[\lim_{t\to\infty} d(f(\mathcal{A}_t), \mathcal{PF}^*) = 0\right] = 1.$$

□

5. Uporaba evolucijskega algoritma za problem trgovskega potnika

5.1. **Večkriterijski problem trgovskega potnika.** Imejmo povezan usmerjeni graf (V, E) brez zank in vzporednih povezav, vsaka povezava pa naj ima m uteži. Denimo, da želimo najti pot, ki obišče vsa vozlišča in minimizira m kriterijev, ki so enaki vsoti vseh ustreznih uteži povezav, ki jih prepotujemo. Brez izgube splošnosti

lahko vozlišča označimo z naravnimi števili, torej $V = \{1, 2, \ldots, n\}$. Uteži na povezavah podamo z $n \times n$ matrikami povezav $U^{(k)}$, $k \in \{1, 2, \ldots, m\}$. Če s $c_{ij}^{(k)}$ označimo vrednost k-te uteži na povezavi med vozliščema i in j, je

$$u_{ij}^{(k)} = \begin{cases} c_{ij}^{(k)}; & ij \in E \\ \infty; & ij \notin E. \end{cases}$$

(i, j)-ti element matrike $U^{(k)}$. Upoštevajmo oznake iz drugega razdelka tega dela in opredelimo večkriterijski optimizacijski problem.

Iščemo vrstni red obiskov vozlišč, kar pomeni, da je rešitev permutacija vozlišč, zato $\Omega = S_n$. Kriterijsko funkcijo $f_k \colon S_n \to \overline{\mathbb{R}}$ definiramo kot vsoto k-tih uteži povezav poti, ki obišče vozlišča v danem vrstnem redu. Za permutacijo $\pi \in S_n$ je zato

$$f_k(\pi) = \sum_{i=1}^{n-1} u_{\pi(i),\pi(i+1)}^{(k)}.$$

Seveda pa vsi vrstni redi obiskov niso dopustni, saj mogoče med katerima vozliščema ni neposredne povezave. Dopustne so natanko tiste permutacije, pri katerih so vrednosti vseh kriterijskih funkcij končne oz.

$$D = \{\pi \in S_n;\ \text{za vsak } k \in \{1, 2, \ldots, m\} : f_k(\pi) < \infty\}.$$

Če je $f = (f_1, f_2, \ldots, f_m)$, je z $(D, f, \min)$ optimizacijski problem natančno določen.

Primer 5.1. Zamislimo si sedaj naslednjo realno situacijo. Popotnik si je za svoj naslednji podvig izbral obisk vseh 27 prestolnic Evropske unije. Vseeno je, v katerih mestih začne in konča svoje popotovanje, le obiskati mora vsako glavno mesto natanko enkrat. Za poljubni dve mesti pozna zračno razdaljo in ceno ter trajanje javnega prevoza med njima, če le obstaja neposredna povezava med mestoma. Naš popotnik si načrtovanje potovanja olajša tako, da že vnaprej izbere eno povezavo, če je na določeni relaciji več ponudnikov javnega prevoza. Zanima ga le, v kakšnem vrstnem redu naj prepotuje prestolnice, da bo potovanje cenovno najugodnejše, časovno najkrajše in da bo skupna prepotovana zračna razdalja najmanjša. Problem ustreza zgoraj predstavljenemu v primeru, ko je $n = 27$ in $m = 3$.

V nadaljevanju si natančneje poglejmo evolucijski algoritem NSGA II za reševanje tega problema popotnika.

5.2. **NSGA II.** Določiti moramo vse ključne sestavine evolucijskega algoritma, čeprav sta izbira staršev in izbira osebkov za naslednjo generacijo ključna operatorja, ki determinirata NSGA II. Pred tem pa vpeljimo novo relacijo na domeni Ω kriterijskih funkcij.

NSGA II je elitistični evolucijski algoritem, ki osebke razvršča na dveh nivojih, in sicer v

- nedominirane fronte in
- glede na kopičenje rešitev znotraj ene nedominirane fronte.

Za večkriterijski optimizacijski problem $(D, f, \min)$ definirajmo nedominirano fronto ranga i.

Definicija 5.2. *Nedominirana fronta ranga i* glede na $X \subseteq D$ je množica Pareto optimalnih rešitev glede na množico $X \setminus \cup_{k=1}^{i-1} \mathcal{F}_k$, pri čemer $\mathcal{F}_k$ označuje nedominirano fronto ranga k glede na X.

Nedominirana fronta ranga 1 je tako enaka množici Pareto optimalnih rešitev glede na X.

Za $x \in \mathcal{F}_i$ bomo rekli, da ima rang i. Očitno je $f(\mathcal{F}_i)$ enaka množici minimalnih elementov v delno urejeni množici $\left(f\left(X \setminus \cup_{k=1}^{i-1} \mathcal{F}_k\right), \preceq\right)$, zato sta poljubna osebka z enakim rangom neprimerljiva.

Opomba 5.3. Ker se v generaciji nekateri osebki lahko pojavijo večkrat, obdržimo večkratnost tudi pri razvrščanju v nedominirane fronte. Zato bomo s F_i označili multimnožico tistih rešitev iz multimnožice X, ki imajo rang i.

Z razvrstitvijo v nedominirane fronte še ne dobimo linearne urejenosti, da bi lahko izbrali μ najboljših osebkov. Primerjati moramo znati še osebke istega ranga, pri čemer moramo paziti, da ne damo prednosti točno določeni rešitvi, ker bi s tem prišli v navzkrižje z idealnim pristopom. To dosežemo z nakopičenjem.

Vsakemu osebku določimo *metriko nakopičenosti* kot prikazuje algoritem 3. Ta je

Algoritem 3 Določanje metrike nakopičenosti

procedure METRIKA_NAKOPIČENOSTI(F_k, g)
  $l = |F_k|$
  for all $1 \leq i \leq l$ **do**
    nakopičenost $(F_k[i]) = 0$
  end for
  for all $1 \leq j \leq m$ **do**
    uredi $(\mathcal{F}_k, j)$ ▷ urejanje glede na j-ti kriterij
    nakopičenost $(F_k[1])$ = nakopičenost $(F_k[l]) = \infty$
    for all $2 \leq i \leq l-1$ **do**
      nakopičenost $(F_k[i])$ = nakopičenost $(F_k[i]) + \frac{f_j(F_k[i+1]) - f_j(F_k[i-1])}{f_j^{\max} - f_j^{\min}}$
    end for
  end for
end procedure

intuitivno enaka vsoti dolžin vseh robov s skupnim ogliščem v kvadru, ki ga določajo osebkovi najbljižji sosedje (po posameznih kriterijih), le da so dolžine normirane glede na razliko med največjo in najmanjšo vrednostjo k-te kriterijske funkcije f_k problema $(D, f, \min)$ na množici $\mathcal{F}_k$. Večja kot je metrika nakopičenosti, bolj osamljen je osebek, boljši je z vidika raznolikosti.

Rang osebka in metrika nakopičenosti skupaj določata linearno relacijo na množici osebkov $\mathcal{P}_t$. Označimo $x \preceq^{\mathrm{n}} y$, če

- $x_{\text{rang}} < y_{\text{rang}}$ ali
- $x_{\text{rang}} = y_{\text{rang}}$ in $x_{\text{nakopičenost}} \geq y_{\text{nakopičenost}}$.

Če za osebka istega ranga velja $x \preceq^{\mathrm{n}} y$ in $x_{\text{nakopičenost}} > y_{\text{nakopičenost}}$, pišemo $x \prec^{\mathrm{n}} y$. Enak zapis je tudi v primeru, ko je $x_{\text{rang}} < y_{\text{rang}}$.

Predstavitev osebkov. Iz definicije zastavljenega problema permutacija jasno izstopa kot možna reprezentacija rešitev.

Generacija. Velikost generacije P_t naj bo μ in naj se tekom izvajanja algoritma ne spreminja. Tako na začetku izberemo naključnih μ permutacij, ki so paroma lahko enake. V generaciji dopustimo tudi neprimerne osebke, ki pa jih kaznujemo s funkcijo uspešnosti. Raznolikosti ne obravnavamo na celotni generaciji, temveč le

med osebki, ki med sabo niso primerljivi. Doseči jo poskušamo z nakopičenostjo, več o tem pa bomo povedali, ko bomo določali naslednjo generacijo.

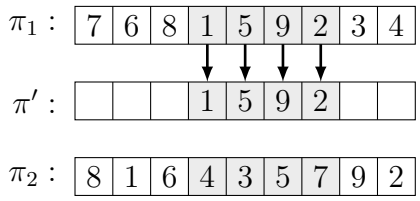
Funkcija uspešnosti. Za funkcijo uspešnosti vzamemo kar funkcijo f iz definicije problema. Definirana je na množici vseh permutacij, posamezna kriterijska funkcija pa lahko zavzame tudi vrednost ∞, če pot "vsebuje" kakšno povezavo, ki ne obstaja.

Selekcija in križanje. Vsakič ustvarimo največ μ potomcev. Križanje izvedemo z verjetnostjo p_c, ki je parameter algoritma.

Starše izbiramo s turnirsko selekcijo. Iz trenutne generacije naključno izberemo dva osebka π in ρ. Če $\pi \prec_f \rho$, zmaga π, če $\rho \prec_f \pi$, pa za enega izmed staršev določimo ρ. Lahko se zgodi, da ρ in π nista primerljiva. V tem primeru izberemo tistega, ki je boljši glede na relacijo $\prec^n$. Če še vedno ne moremo določiti zmagovalca, potem naključno izberemo enega od njiju (vsak je izbran z verjetnostjo $\frac{1}{2}$).

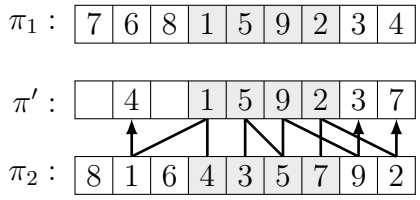
Dva osebka π_1 in π_2 križamo s križanjem z delno preslikavo.

1. **korak:** Najprej na slepo izberemo točki križanja, ki določata neko območje. Le-to je pri potomcu π' enako istoležnemu območju osebka π_1, kot prikazuje slika 10.



Slika 10. Območje med točkama križanja prepišemo.

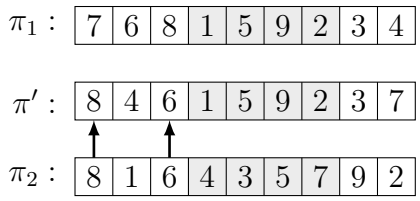
2. **korak:** Nato vrednosti, ki se pojavijo v danem območju osebka π_2 in se ne nahajajo v omenjenem območju potomca, prepišemo na ustrezna prosta mesta potomca π'. Če izberemo vrednost a, pogledamo katera je istoležna vrednost b v osebku π'. Če se vrednost b v osebku π_2 nahaja na nekem mestu in je v potomcu to isto mesto še nezasedeno, tja prepišemo vrednost a. Če pa je zasedeno z vrednostjo c, v π_2 poiščemo c in spet preverimo, ali je istoležno mesto v π' že zasedeno. Postopek ponavljamo, dokler ne najdemo prostega mesta. To nazorno prikazuje slika 11.



Slika 11. Iskanje prostega mesta.

3. **korak:** Če po teh dveh korakih v π' še niso zasedena vsa mesta, tja prepišemo istoložne vrednosti iz osebka π_2, kar ponazarja slika 12.

Tako iz enega para staršev dobimo prvega potomca. Če zamenjamo vlogi π_1 in π_2, dobimo še drugega potomca istih staršev.



Slika 12. Prepis preostalih vrednosti.

Opomba 5.4. Ali s križanjem z delno preslikavo vedno dobimo permutacijo? Naj bo $\mathcal{A}$ množica n elementov in π_1 ter π_2 permutaciji teh elementov, ki sta izbrani za križanje. Z $\mathcal{B}$ označimo množico indeksov, ki predstavljajo območje križanja. Razbijmo množico $\mathcal{A}$ na tri dele:

- $\mathcal{A}_1 = \left\{x \in \mathcal{A};\ \pi_1^{-1}(x) \in \mathcal{B}\right\}$,
- $\mathcal{A}_2 = \left\{x \in \mathcal{A};\ \pi_2^{-1}(x) \in \mathcal{B},\ x \notin \mathcal{A}_1\right\}$ in
- $\mathcal{A}_3 = \{x \in \mathcal{A};\ x \notin \mathcal{A}_1,\ x \notin \mathcal{A}_2\}$.

Množica $\mathcal{A}_1$ torej vsebuje elemente iz osebka π_1, ki se nahajajo v območju križanja, množica $\mathcal{A}_2$ elemente iz osebka π_2, ki so v istem območju, množica $\mathcal{A}_3$ pa tiste, ki so izven območja križanja v obeh osebkih.

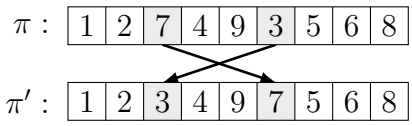
Očitno je $\pi'(i) = \pi_1(i)$ za vsak indeks $i \in \mathcal{B}$, torej se vsak element iz množice $\mathcal{A}_1$ vsaj enkrat pojavi v potomcu π'. Kaj se zgodi z elementi množice $\mathcal{A}_2$? Ker je pred začetkom 2. koraka območje križanja v π' že zasedeno, mesto $(\pi')^{-1}(x)$ zagotovo ni vsebovano v $\mathcal{B}$ za $x \in \mathcal{A}_2$. Ali se lahko zgodi, da bi pri iskanju prostega mesta v 2. koraku izven območja križanja naleteli na zasedeno mesto? To bi se zaradi bijektivnosti permutacij π_1 in π_2 lahko zgodilo le v primeru, ko bi za dva različna začetna elementa x in y območje $\mathcal{B}$ zapustili iz istega mesta.

Opazimo, da je $\pi'\left(\pi_2^{-1}(x)\right) \in \mathcal{A}_1$ za vsak $x \in \mathcal{A}_2$, kar pomeni, da dokler ne zapustimo območja križanja, pri iskanju prostega mesta srečujemo le elemente iz $\mathcal{A}_1$. Označimo s $\mathcal{C}_x$ elemente iz $\mathcal{A}_1$, na katere naletimo v π', preden zapustimo območje $\mathcal{B}$, če za izhodišče vzamemo $x \in \mathcal{A}_2$. Naj bo $y \in \mathcal{A}_2$ različen od x. Tedaj je $\mathcal{C}_x \cap \mathcal{C}_y = \emptyset$. Recimo, da obstaja z v preseku. Če je $\pi_2\left((\pi')^{-1}(z)\right) \in \mathcal{A}_2$, bi sledilo $x = y$ zaradi bijektivnosti π_1 (ki je na $\mathcal{B}$ enaka π') in dejstva, da na območju križanja srečujemo zgolj elemente iz $\mathcal{A}_1$ z izjemo izhodiščnega elementa. Če pa je $z' := \pi_2\left((\pi')^{-1}(z)\right) \in \mathcal{A}_1$, je tedaj zaradi bijektivnosti π_1 z' enolično določen in zanj velja $z' \in \mathcal{C}_x \cap \mathcal{C}_y$. Lahko bi se zgodilo, da sta z in z' enaka. To je možno le, kadar se element z pojavi na istem mestu v π_1 in π_2, vendar pa v tem primeru ne obstaja izhodiščni element $x \in \mathcal{A}_2$, da bi veljalo $z \in \mathcal{C}_x$. Do elementa z v π' pridemo le iz z' v π_2, ki pa sta enaka. Sledi, da $z \neq z'$. Postopek nato ponovimo na z', itn. Zaradi končnosti množice $\mathcal{C}_x \cap \mathcal{C}_y$ slej ali prej pridemo do elementa w, za katerega je $\pi_2\left((\pi')^{-1}(w)\right) \in \mathcal{A}_2$, od koder pa sledi $x = y$. To je protislovje s predpostavko, da je x različen od y, zato je $\mathcal{C}_x \cap \mathcal{C}_y = \emptyset$. Posledično je tudi zadnji element, ki ga obiščemo pred izhodom iz $\mathcal{B}$, različen za vsak element iz $\mathcal{A}_2$, torej je zaradi bijektivnosti π_1 za vsak $x \in \mathcal{A}_2$ drugačno tudi prvo mesto, na katerega pridemo takoj po izhodu iz območja križanja in kamor neovirano zapišemo izhodiščni element x.

Na koncu elemente iz $\mathcal{A}_3$ prepišemo na istoležna mesta v π', ki so še nezasedena, saj na 2. koraku teh mest sploh nismo pregledovali.

Za vsak element iz $\mathcal{A}$ smo našli enolično določeno mesto v potomcu π', ki je zato spet permutacija.

Mutacija. Uporabimo najpreprostejši način mutacije za permutacije. Poljuben potomec je za mutacijo izbran z verjetnostjo p_{m}. Potem na slepo izberemo dve točki, katerih vrednosti preprosto zamenjamo (slika 13).



Slika 13. Mutacija izbranega potomca.

Določitev nove generacije. Kot smo že omenili, NSGA II uporablja elitistični pristop, pri tem pa se opira na relacijo $\preceq^{\mathrm{n}}$.

Najprej osebke razvrstimo v nedominirane fronte, kot prikazuje algoritem 4. Ta najprej vsakemu elementu x določi, koliko rešitev iz trenutne populacije je boljših od x. Tisti osebki, za katere ne obstaja noben boljši, tvorijo prvo nedominirano fronto. Postopek nato ponovimo na preostalih elementih.

Osebki prve nedominirane fronte so boljši kot tisti, ki imajo rang večji od 1, zato v novo populacijo najprej vzamemo celotno prvo nedominirano fronto, če le vsebuje manj kot μ osebkov. Dokler je v novi generaciji dovolj prostora, izbiramo celotne nedominirane fronte. Slej ali prej pa se bo zgodilo, da ne bomo mogli izbrati vseh osebkov istega ranga, ampak le nekatere. Toda katere izmed teh zavreči in katerih ne? Na tem mestu uporabimo metriko nakopičenosti osebkov nedominirane fronte z najnižjim rangom, ki je bila prevelika, da bi celotno dodali v naslednjo generacijo. Med njimi sedaj izberemo toliko osebkov z največjo metriko nakopičenosti, kolikor jih še lahko. Na tem mestu se ne obremenjujemo z urejenostjo osebkov, ki so glede na relacijo $\preceq^{\mathrm{n}}$ enaki. Načelno je vrstni red enolično določen, če poznamo začetni vrstni red v generaciji in populaciji potomcev ter vrsto vseh urejanj, ki se izvedejo do takrat. Odvisno je namreč od tega, ali algoritmi urejanja ohranjajo vrstni red enakovrednih elementov ali ne. Zaradi slučajnosti glavnih operatorjev, je začetni vrstni red generacij naključen, zato lahko rečemo, da so osebki istega ranga z enako metriko nakopičenosti urejeni na slepo.

Naslednjo generacijo torej preprosto izberemo s tem, ko osebke linearno uredimo z relacijo $\preceq^{\mathrm{n}}$. Z vidika časovne zahtevnosti lahko privarčujemo nekaj operacij, če določamo nakopičenost osebkov le tiste fronte, za katero je to nujno potrebno, taka pa je največ ena. Celoten postopek urejanja je lepo razviden s slike 14.

Zaustavitveni pogoj. Algoritem ustavimo po določenem številu generacij, ki je parameter algoritma ($G_{\max}$).

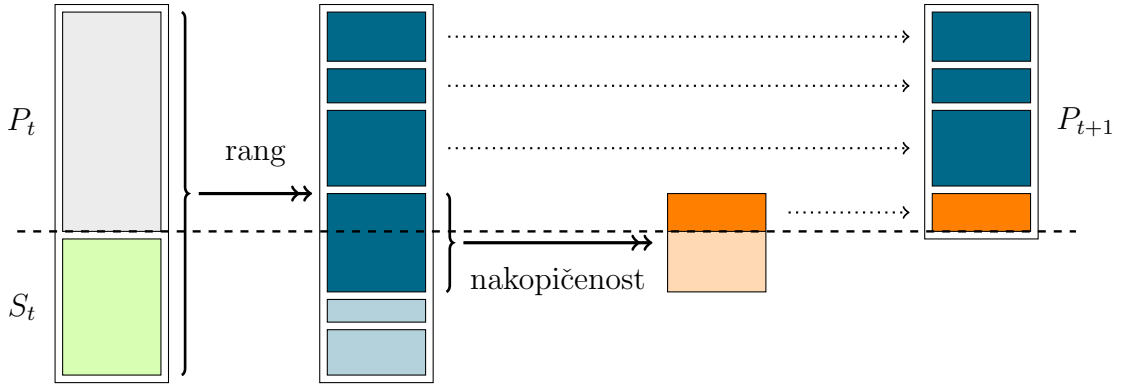
5.3. **Nastavljanje parametrov in analiza rezultatov.** Algoritem zaradi nedeterminističnosti ne vrne zagotovo optimalne rešitve. Pridobili smo polinomsko časovno zahtevnost in upamo, da smo kljub temu še vedno dobili rezultat, ki je za nas dovolj dober. Pri tem se nam porodi vprašanje, kako ovrednotiti kakovost rešitev in ali jih lahko z nastavljanjem parametrov izboljšamo.

Zaradi večkriterijske optimizacije ponovno naletimo na problem primerjave. Obstaja več metrik, ki ocenjujejo izvedbo evolucijskega algoritma. Mi bomo uporabili *metriko hipervolumna.*

Naj bo $\mathcal{A}$ neka podmnožica prostora $f(D) \subseteq \mathbb{R}^m$, ki predstavlja približek Pareto fronte, ki ga vrne evolucijski algoritem in $r \in \mathbb{R}^m$ takšen vektor, da je $\mathcal{A} \preceq_\forall r$

Algoritem 4 Hitro nedominirano razvrščanje

```
1: procedure HITRO_NEDOMINIRANO_RAZVRŠČANJE(P_t)
2:     for all x ∈ P do
3:         S_x = ∅                                  ▷ osebki, ki jih x dominira
4:         n_x = 0                                  ▷ število osebkov, ki dominirajo x
5:         for all y ∈ P do
6:             if x ≺_f y then
7:                 S_x = S_x ∪ {y}
8:             else if y ≺_f x then
9:                 n_x = n_x + 1
10:            end if
11:        end for
12:        if n_x = 0 then                          ▷ x je Pareto optimalen
13:            x_rang = 1
14:            F_1 = F_1 ∪ {x}
15:        end if
16:    end for
17:    i = 1                                        ▷ števec nedominiranih front
18:    while F_i ≠ ∅ do
19:        Q = ∅
20:        for all x ∈ F_i do
21:            for all y ∈ S_x do
22:                n_y = n_y − 1
23:                if n_y = 0 then                  ▷ y pripada naslednji fronti
24:                    y_rang = i + 1
25:                    Q = Q ∪ {y}
26:                end if
27:            end for
28:        end for
29:        i = i + 1
30:        F_i = Q
31:    end while
32: end procedure
```



Slika 14. Razvrščanje osebkov glede na rang in metriko nakopičenosti za izbor naslednje generacije.

za vsako podmnožico $\mathcal{A} \subseteq f(D)$ oz. da vsak element poljubne podmnožice strogo dominira r. Izberimo še poljuben $x \in \mathbb{R}^m$. Zapis $\mathcal{A} \preceq_\exists x$ pove, da obstaja $y \in \mathcal{A}$, ki dominira x (tj. $y \preceq x$).

Opomba 5.5. r imenujemo *referenčna točka.*

Definicija 5.6. *Funkcija dosegljivosti* za množico $\mathcal{A} \subseteq \mathbb{R}^m$ z referenčno točko r je funkcija $\nu_{r,\mathcal{A}} \colon \mathbb{R}^m \to \{0,1\}$, podana s predpisom

$$\nu_{r,\mathcal{A}}(x) = \begin{cases} 1; & \mathcal{A} \preceq_\exists x,\ x \preceq r, \\ 0; & \text{sicer.} \end{cases}$$

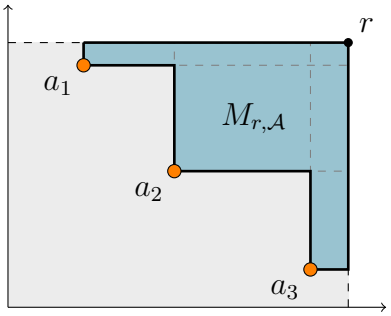
Funkcija dosegljivosti precej spominja na karakteristično funkcijo, kar niti ni naključje. $\nu_{\mathcal{A}}$ je namreč karakteristična funkcija množice

$$M_{r,\mathcal{A}} := \{x \in \mathbb{R}^m;\ \mathcal{A} \preceq_\exists x,\ x \preceq r\}$$

tistih vektorjev iz $\mathbb{R}^m$, ki strogo dominirajo r in jih dominira vsaj en element množice $\mathcal{A}$.

Definicija 5.7. *Metrika hipervolumna* množice $\mathcal{A}$ je enaka volumnu množice $M_{r,\mathcal{A}}$ oz.

$$I_r(\mathcal{A}) = \int_{\mathbb{R}^m} \nu_{r,\mathcal{A}}(x)\, dx.$$



Slika 15. $M_{r,\mathcal{A}}$ za $\mathcal{A} = \{a_1, a_2, a_3\}$.

Trditev 5.8. *Če je $f(D)$ končna, potem funkcija $I_r \colon 2^{f(D)} \to \mathbb{R}$, ki predstavlja metriko hipervolumna, doseže maksimum v Pareto fronti $\mathcal{PF}^*$.*

Dokaz. Preveriti moramo le, da $I_r(\mathcal{A}) \leq I_r(\mathcal{PF}^*)$ za poljuben približek $\mathcal{A}$. Zaradi karakteristične narave funkcije dosegljivosti je dovolj videti, da je $M_{r,\mathcal{A}} \subseteq M_{r,\mathcal{PF}^*}$.

Izberimo $\mathcal{A} \subseteq f(D)$ poljuben približek in $x \in M_{r,\mathcal{A}}$. Ali je $\mathcal{PF}^* \preceq_\exists x$? Ker je $x \in M_{r,\mathcal{A}}$, obstaja $y \in \mathcal{A}$, da je $y \preceq x$. $f(D)$ je končna, zato je po lemi 2.20 $(f(D), \preceq)$ polna, torej obstaja $z \in \mathcal{PF}^*$, ki dominira y. Zaradi tranzitivnosti relacije $\preceq$ je $z \preceq x$, od koder sledi $\mathcal{PF}^* \preceq_\exists x$, kar nam da $M_{r,\mathcal{A}} \subseteq M_{r,\mathcal{PF}^*}$. Tedaj velja $\nu_{r,\mathcal{A}}(x) \leq \nu_{r,\mathcal{PF}^*}(x)$, zato tudi $I_r(\mathcal{A}) \leq I_r(\mathcal{PF}^*)$, saj je integral monoton. □

Želimo konvergenco k množici Pareto rešitev, torej pričakujemo čim višjo vrednost metrike hipervolumna.

Problem, ki ostane je, kako določiti referenčno točko. S tem se ne bomo podrobneje ukvarjali, ker ni glavna tema tega dela. To problematiko obravnava članek [10].

Osredotočimo se na primer 5.1 z začetka tega razdelka. V našem primeru izberemo vektor $r = (r_1, r_2, r_3)$ (optimiziramo po treh kriterijih) s koordinatami

$$r_k = \sum_{i=1}^{27} \max_{1 \le j \le 27} \left\{ u_{ij}^{(k)}; u_{ij}^{(k)} < \infty \right\} - \min_{1 \le i \le 27} \max_{1 \le j \le 27} \left\{ u_{ij}^{(k)}; u_{ij}^{(k)} < \infty \right\}, k \in \{1, 2, 3\}$$

kjer so $u^{(k)}$ matrike medsebojnih razdalj, cen in trajanj. Za tako izbran r sicer velja $\mathcal{A} \preceq_\forall r$, če le $\mathcal{A}$ vsebuje samo dopustne rešitve. Če je v približku $\mathcal{A}$ kakšna nedopustna rešitev, določimo $I_r(\mathcal{A}) = 0$. Označimo s $\mathcal{K}$ kvader, ki dominira r, tj. $\mathcal{K} \preceq_\forall r$. Opazovali bomo delež $\mathcal{K}$, ki ga pokrije množica $M_{r,\mathcal{A}}$, torej ne bomo gledali absolutnega volumna $M_{r,\mathcal{A}}$, ampak relativnega.

$G_{\max}$	μ	p_c	p_m	$\overline{d_I}$	σ_{d_I}	r
7500	250	0.9	0.4	0.4867	0.0159	1.00
			0.04	0.4742	0.0178	1.00
		0.5	0.4	0.4834	0.0184	1.00
			0.04	0.4645	0.0177	1.00
	50	0.9	0.4	0.4697	0.0157	1.00
			0.04	0.2992	0.2173	0.66
		0.5	0.4	0.4492	0.0947	0.96
			0.04	0.2419	0.2261	0.54
750	250	0.9	0.4	0.4620	0.0691	0.98
			0.04	0.4355	0.0662	0.98
		0.5	0.4	0.4118	0.1267	0.94
			0.04	0.3723	0.1397	0.90
	50	0.9	0.4	0.2495	0.2162	0.58
			0.04	0.1969	0.1947	0.52
		0.5	0.4	0.1065	0.1806	0.28
			0.04	0.0900	0.1501	0.28

TABELA 1. Rezultati meritev za 50 zagonov algoritma pri različnih nastavitvah parametrov ($\overline{d_I}$ je povprečje vrednosti metrik hipervolumna, σ_{d_I} standardni odklon, r pa delež ponovitev, pri katerih je algoritem vrnil dopustno rešitev).

Meritve prikazuje tabela 1 za 50 ponovitev zagona algoritma pri posameznih nastavitvah parametrov. Lepo se vidi, da večje kot je število izračunanih generacij $G_{\max}$, bližje so rezultati Pareto fronti. Poleg tega je razvidno tudi, da je v vseh primerih ugodnejša verjetnost mutacije $p_m = 0.4$, podobno kot je za p_c ugodnejša vrednost 0.9 in 250 za velikost generacije.

Pri tem je treba pripomniti, da so standardni odkloni višji na dnu tabele (tj. pri manjšem številu dovoljenih generacij), saj pri teh nastavitvah parametrov pride velikokrat do tega, da algoritem v tako kratkem času še ne najde dopustne rešitve in so zato ti nabori parametrov neprimerni. Če pa jo najde, jo do izteka algoritma bolj ali manj uspešno izboljša. Hkrati lahko iz dobljenih rezultatov sklepamo, da je v začetku konvergenca precej hitra, po dolgem času pa se že skoraj ustavi.

Literatura

[1] C. A. Coello Coello, D. A. Van Veldhuizen in G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer, 2007.

[2] T. Robič in B. Filipič, *Večkriterijska optimizacija z evolucijskimi algoritmi*, v Proceedings of the thirteenth International Electrotechnical and Computer Science Conference - ERK 2004, Volume B (ur. Baldomir Zajc), Portorož, 2004, str. 149–152.

[3] G. Rudolph in A. Agapie, *Convergence Properties of Some Multi-Objective Evolutionary Algorithms*, v Proceedings of the 2000 Congress on Evolutionary Computation, Volume 2, Los Angeles, 2000.

[4] *Classic Methods for Multi-Objective Optimization*, [ogled 17. 12. 2011], dostopno na `http://cims.nyu.edu/~gn387/glp/lec2.pdf`.

[5] A. E. Eiben in J. E. Smith, *Introduction to Evolutionary Computing*, Natural Computing Series **XVI** Springer, New York, 2007.

[6] G. Grimmett in D. Stirzaker, *Probability and Random Processes*, Oxford University Press, New York, 2001.

[7] G. Grimmet in D. Stirzaker, *One Thousand Exercises in Probability*, Oxford University Press, New York, 2001.

[8] K. Deb, A. Pratap, S. Agarwal in T. Meyarivan, *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*, v IEEE Transactions on Evolutionary Computation (ur. David B. Fogel) **6**, IEEE Transactions on Evolutionary Computation, 2002, str. 182–197.

[9] E. Zitzler, D. Brockhoff in L. Thiele, *The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration*, v Conference on Evolutionary Multi-Criterion Optimization (EMO 2007) (ur. S. Obayashi in drugi), Lecture Notes in Computer Science **4403**, Springer, Berlin, 2007, str. 862–876.

[10] A. Auger, J. Bader, D. Brockhoff in E. Zitzler, *Theory of the Hypervolume Indicator: Optimal μ-Distributions and the Choice of the Reference Point*, Foundations of Genetic Algorithms (FOGA 2009), ACM, New York, 2009, str. 87–102.

[11] *NP-hard*, [ogled 10. 6. 2012], dostopno na `http://en.wikipedia.org/wiki/NP-hard`.

[12] *Time complexity*, [ogled 19. 3. 2012], dostopno na `http://en.wikipedia.org/wiki/Time_complexity`.