An Algorithm for Drawing Planar Graphs

Bor Plestenjak
IMFM/TCS, University of Ljubljana, Jadranska 19, SI-1111 Ljubljana, Slovenia
email: bor.plestenjak@fmf.uni-lj.si

SUMMARY

A simple algorithm for drawing 3-connected planar graphs is presented. It is derived from the Fruchterman and Reingold spring embedding algorithm by deleting all repulsive forces and fixing vertices of an outer face. The algorithm is implemented in the system for manipulating discrete mathematical structures Vega. Some examples of derived figures are given.

KEY WORDS Planar graph drawing Force-directed placement

INTRODUCTION

A graph G is planar if and only if it is possible to draw it in a plane without any edge intersections. Every 3-connected planar graph admits a convex drawing [1], i.e. a planar drawing where every face is drawn as a convex polygon. We present a simple and efficient algorithm for convex drawing of a 3-connected planar graph. In addition to a graph, most existing algorithms for planar drawing need as an input all the faces of a graph [2], while our algorithm needs only one face of a graph to draw it planarly.

Let G be a 3-connected planar graph with a set of vertices $V = \{v_1, \ldots, v_n\}$ and a set of edges $E = \{(u_1, v_1), \ldots, (u_m, v_m)\}$. Let $W = \{w_1, w_2, \ldots, w_k\} \subset V$ be an ordered list of vertices of an arbitrary face in graph G, which is chosen for the outer face in the drawing.

The basic idea is as follows. We consider graph G as a mechanical model by replacing its vertices with metal rings and its edges with elastic bands of zero length. Vertices of the outer face W are fixed in a vertices of a regular polygon of a size k while all other vertices are let free. Under the influence of attractive forces in bands free vertices will move until the system finally reaches an equilibrium. The layout in the equilibrium is a convex drawing.

The algorithm is included in package Vega [3, 4] under the name `SchlegelDiagram`. Vega is Mathematica based system for manipulating graphs and other combinatorial structures, such as groups, maps, etc., developed by IMFM/TCS, Ljubljana. Vega is mainly written in Mathematica, however it also includes external programs for time complex operations, e.g. the implementation of our algorithm.

ALGORITHM

The algorithm is based on the force-directed placement graph drawing algorithm by Fruchterman and Reingold [5]. In each step the algorithm calculates the effect of attractive forces on each vertex and then moves the vertex in the direction of the resultant force. The displacement in step i is limited to some maximum value $\mathit{cool}(i)$ that we call temperature and which satisfies $\lim_{i \to \infty} \mathit{cool}(i) = 0$. As the temperature decreases to zero, smaller and smaller displacements are allowed and after some number of steps displacements are so small that we can say that the layout freezes. The algorithm ends after the prescribed number of iterations and if the forces and the cooling function $\mathit{cool}(i)$ are chosen properly, the final layout is so near to the equilibrium that it is planar.

For the force between two adjacent vertices u and v we use the third order law

$$F_{uv} = Cd^3, \tag{1}$$

where C is a constant and $d = \|u.pos - v.pos\|$ is the distance. We tested different order powers and the result is a compromise between good results and an efficient calculation.

The algorithm consists of the following steps.

1. Position all vertices of an outer face W in vertices of a regular polygon of size k inscribed into the unit circle and put all other vertices in the origin.

2. For $i := 1$ to *iterations*:

   (a) For all vertices $v \in V$: set the resultant force F_v to zero.

   $$F_v = 0.$$

   (b) For all edges $(u, v) \in E$: calculate the attractive force F_{uv} and update the resultant forces F_u and F_v.

   $$F_{uv} = C(v.pos - u.pos)^3,$$
   $$F_u = F_u + F_{uv},$$
   $$F_v = F_v - F_{uv}.$$

   (c) For all vertices $v \in V - W$: move vertex v in the direction of the resultant force F_v for the size of the force, but not more than for the value of $cool(i)$.

   $$v.pos = v.pos + \min\Big(|F_v|, cool(i)\Big)\frac{F_v}{|F_v|}. \tag{2}$$

Constant C in (1) has to be in harmony with the maximum displacement. Namely, if C is too large then in (2) every vertex is shifted for the maximum displacement irrespective of the force size. On the other hand, if C is too small then the displacements are small and the algorithm does not reach the equilibrium in a prescribed number of steps.

If the temperature decreases too slowly then displacements are too large and too many steps are needed to reach the equilibrium. Yet, if the temperature decreases too fast then the layout freezes in a non planar drawing.

A suggested values for constant C and function $cool(i)$ are:

$$C = \sqrt{\frac{n}{\pi}}, \tag{3}$$

$$cool(i) = \frac{\sqrt{\frac{\pi}{n}}}{1 + \frac{\pi}{n}i^{3/2}}. \tag{4}$$

The expression $\frac{\pi}{n}$ is the average area belonging to one vertex and therefore we take $2\sqrt{\frac{\pi}{n}}$ for the average distance between adjacent vertices. This justifies the presence of term $\frac{\pi}{n}$ in (3) and (4).

Figures 1 to 3 demonstrate how the algorithm works. The graph is the icosahedral fullerene isomer of C_{180} [6], derived in Vega from the dodecahedron using two leapfrog transformations [7]. We usually pick the largest face for the outer face, but since pentagons are arranged in a shape of a dodecahedron, we choose a pentagon in order to exhibit symmetry. For C and $cool(i)$ we apply (3) and (4), respectively, It can be seen from these figures how the edges closer to the border untwist first and how they force those in the middle to unfold until finally the layout becomes planar.

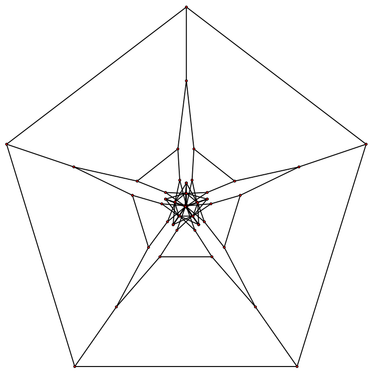
Figure 1. C_{180} *after 5 iterations*

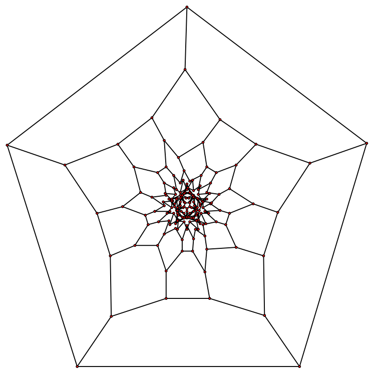
Figure 2. C_{180} *after 30 iterations*

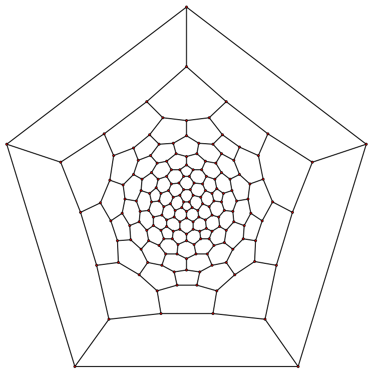
Figure 3. C_{180} *after 500 iterations*

MODIFICATIONS

Periphericity. If the algorithm is applied to a graph with a large number of faces, e.g. a fullerene on many vertices, then it produces a drawing with large faces on the border and large number of crowded small faces in the middle of a figure.

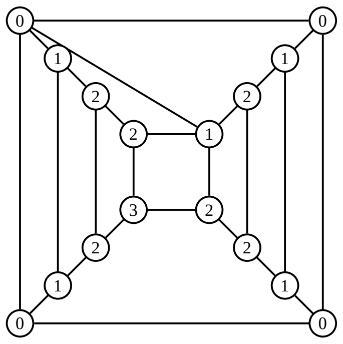
Figure 4. Periphericity of a vertex

In order for the algorithm to terminate with approximately equally arranged faces we assume that the coefficient C in (1) is not equal for all bands. Bands at the border should be stronger and bands in the middle should be weaker. For this purpose we introduce a periphericity of a vertex as the path distance from the vertex to the outer cycle. The definition of the periphericity is more evident in Figure 4, where vertices of a graph are labeled according to their periphericities.

The coefficient of the band between vertices u and v is a function of periphericities $u.per$ and $v.per$ and we denote it by $C(u.per, v.per)$. Let $maxper$ be the maximum periphericity in the graph. As we go through all edges $(u, v) \in E$ the bands with $u.per + v.per = 0$ should be the strongest and the bands with $u.per + v.per = 2maxper$ should be the weakest. Best results are obtained when the coefficients $C(u.per, v.per)$ form a geometric progression in the form

$$C(u.per, v.per) = \sqrt{\frac{n}{\pi}} \exp(A * \frac{2maxper - u.per - v.per}{maxper}). \tag{5}$$

For the constant A we propose $A = 2.5$, but other values can be used as well producing different planar layouts.

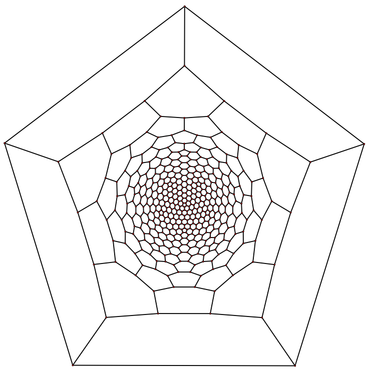
Figure 5. C_{540} *with* $A = 0$ *after 2000 iterations*

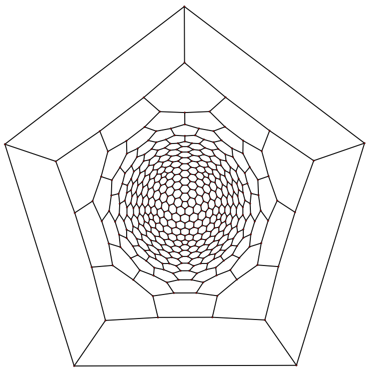
Figure 6. C_{540} *with* $A = 2.5$ *after 2000 iterations*

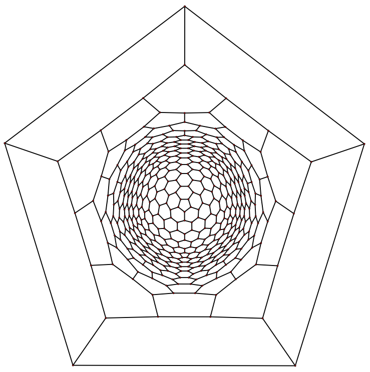
Figure 7. C_{540} *with* $A = 6$ *after 2000 iterations*

Figures 5 to 7 demonstrate the effect of different values of A. The graph is the icosahedral fullerene isomer of C_{540}. In Figure 5 $A = 0$, which equals the original algorithm. In Figure 6 $A = 2.5$ and faces in the middle are more equally arranged. The value $A = 6$ in Figure 7 is too large and the obtained drawing has large faces in the middle that squeeze the rest of the faces. In all three case the cooling function (4) is applied.

Oscillation. After some initial number of steps the vertices begin to oscillate between two states. If we compare the coordinates with the coordinates from two steps ago, we can exit the algorithm when the difference for every vertex is below the prescribed constant ϵ. This approach reduces the number of steps without doing any real harm to the layout quality. For the example we take the graph C_{180}, (4) and (5) with $A = 2.5$. For $\epsilon = 10^{-3}$, 10^{-4} and 10^{-5} the algorithm ends after 227, 627 and 1252 steps, respectively. Differences between layouts are not visible with the naked eye. For instance, the maximum difference between the coordinates of the same vertex after 627 and 1252 steps is $4 \cdot 10^{-3}$.

Non-planar graphs. Although the algorithm is derived for 3-connected planar graphs, it can be applied to other graphs as well. For every graph we can fix an arbitrary set of vertices on the regular polygon, apply a mechanical model to a graph and search for an equilibrium. Although the final layout does not need to be planar, it usually contains useful information about the symmetry and other properties of the graph.

EXAMPLES

The following examples are all obtained using Vega. We use Vega to determine faces of a planar graph using a PQ-tree algorithm [8] for planarity testing and embedding in linear time implemented by J. Marinček [9]. Each face of a planar graph is a good choice for the outer face, but usually we get nicer drawings if we choose the largest face. It is advisable to compare drawings for different choices of the outer face and then choose the most suitable one. For all examples in this section we use (4), (5) with $A = 2.5$ and $\epsilon = 10^{-5}$. When present, the term *it* in figure captions denotes the iteration in which the algorithm reached ϵ and exited.

Figure 8 and Figure 9 are drawings of the Herschel graph [10]. The Herschel graph is a well-known example of a planar nonhamiltonian polyhedron [10]. Notice how the different choice of the outer face affects the layout.

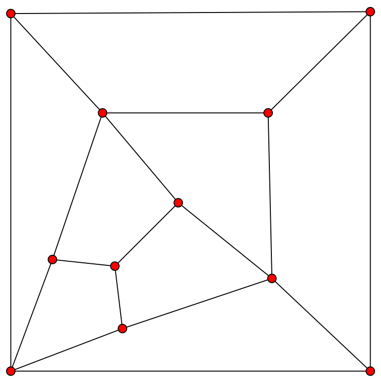

Figure 8. Herschel graph 1
$n = 11$, $m = 18$, *(it* $= 70$*)*

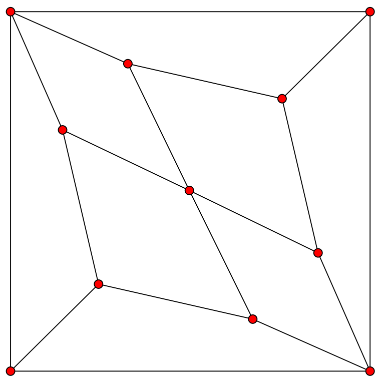

Figure 9. Herschel graph 2
$n = 11$, $m = 18$, *(it* $= 161$*)*

Next two figures represent the Tutte graph and the Grinberg graph. These two graphs are well-known examples of planar cubic 3-connected nonhamiltonian graphs [11]. Figure 10 is the figure of the Tutte graph using the largest face for the outer face, but nicer and more familiar figure is Figure 11 where the outer face has size 9. Figure 12 represents the Grinberg graph.

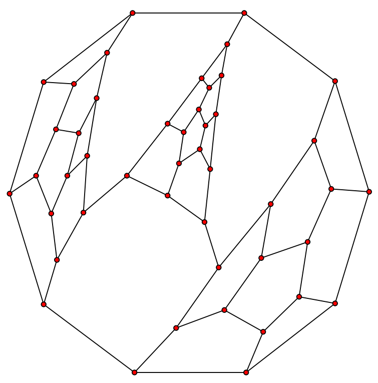

Figure 10. Tutte graph 1
$n = 46$, $m = 69$, *(it* $= 256$*)*

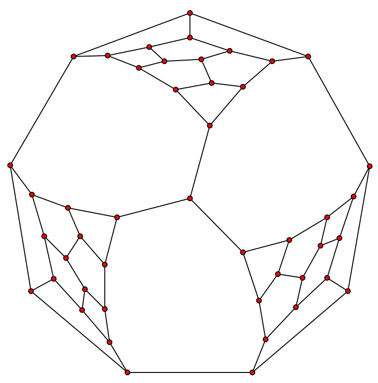

Figure 11. Tutte graph 2
$n = 46$, $m = 69$, *(it* $= 199$*)*

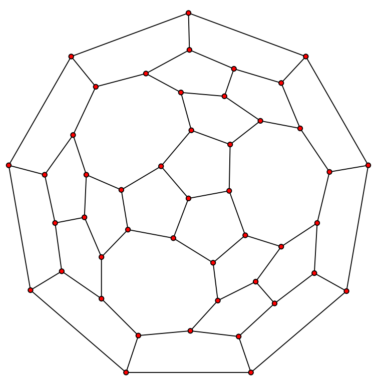

Figure 12. Grinberg graph
$n = 46$, $m = 69$, *(it* $= 143$*)*

Next example presents a planar graph, which is not 3-connected. Figure 13 is the drawing determined automatically by the algorithm, while Figure 14 is the figure calculated after we manually add two temporary edges that are colored gray. When we delete temporary edges we obtain Figure 15. By comparing Figure 13 and Figure 15, it is easy to see that some edges are overlapping in Figure 13.

This trick can be applied to every planar graph, which is not 3-connected. First we add so many temporary edges that the graph becomes 3-connected and then we use the algorithm which returns the convex drawing. Since the planar layout remains planar if we delete an arbitrary number of edges, we simply delete all the temporary edges and end with a planar drawing of the original graph.

The problem is how to add the temporary edges automatically. One of the solutions is to use the 2-dimensional subdivision [12] which adds a new vertex in the center of each face and joins it by edges with all vertices of the face. This transformation is implemented in Vega.

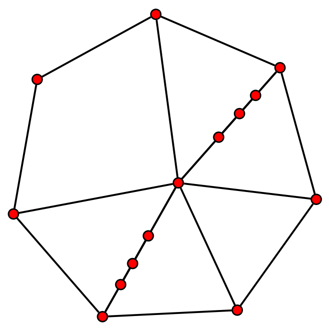

Figure 13. Graph with overlapping edges

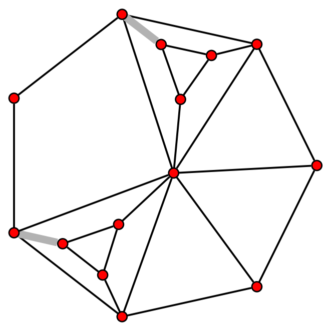

Figure 14. Graph in Figure 13 with added temporary edges

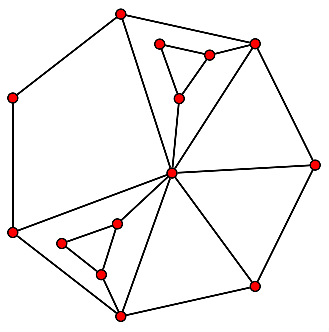

Figure 15. Graph in Figure 13 without overlapping edges

Planar layouts of polyhedra are often more surveyable as their three dimensional layouts [13]. An example is the snub dodecahedron [14] depicted in a planar and in a three dimensional layout in Figure 16 and Figure 17, respectively. The graph in Figure 18 is the medial [7] Grinberg graph derived from Grinberg graph using transformations on maps implemented in Vega.

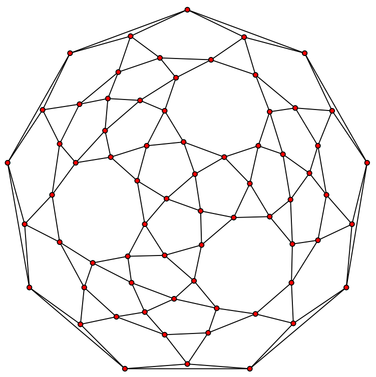
Figure 16. Planar layout of snub dodecahedron (it = 337)

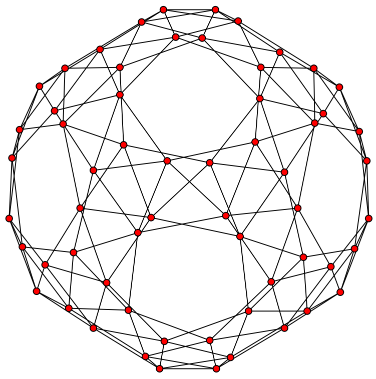
Figure 17. 3D layout of snub dodecahedron

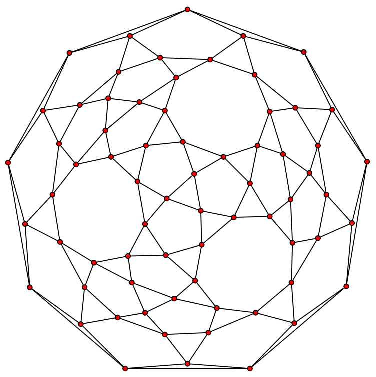
Figure 18. Medial Grinberg graph (it = 361)

Figure 19 displays the result of applying the algorithm to a well-known non-planar Petersen graph and choosing any cycle of length 5 for the outer cycle. Figures 20 and 21 are the results of the algorithm used on a triangulation of a triangle. In Figure 20 a regular polygon is chosen for the outer face and the figure does not have a triangular shape. The choice of three vertices of valence two in Figure 21 produces a non-triangular shape again. This problem can be solved by allowing the user to fix a set of vertices in an arbitrary two or even three dimensional configuration [15].

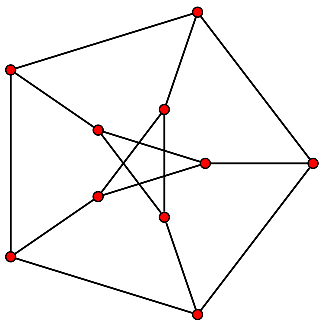
Figure 19. Petersen graph (it = 63)

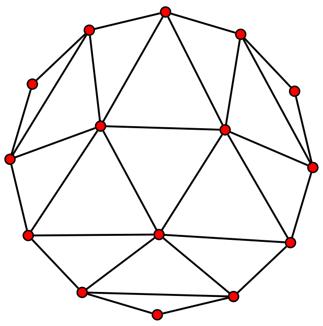
Figure 20. Triangulated triangle 1

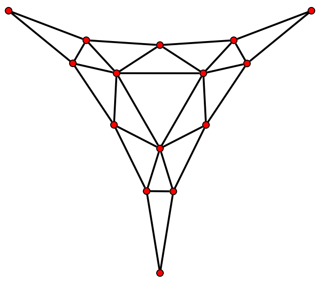
Figure 21. Triangulated triangle 2

CONCLUSIONS

Planar layouts of planar graphs which are often more surveyable as corresponding three dimensional layouts, can be easily produced using this algorithm. This particularly holds for polyhedral graphs, for instance fullerenes.

One of the benefits of the algorithm is its time complexity. One step of the algorithm has time complexity $\mathcal{O}(|E|+|V|)$, while for instance one step of the algorithm by Fruchterman and Reingold has time complexity

$\mathcal{O}(|E| + |V|^2)$. A weakness of the algorithm is that it can be used only for 3-connected planar graphs. This means that planar graphs like trees can not be drawn automatically using this algorithm.

The algorithm offers many improvements. One of them is to allow the user to fix a set of vertices in an arbitrary configuration. The algorithm can also be easily generalized to 3D layouts.

Acknowledgement. The author would like to thank Prof. T. Pisanski at IMFM/TCS Ljubljana for his support and many useful suggestions.

REFERENCES

1. W.T. Tutte, 'Convex Representation of Graphs', *Proc. London Math. Soc.*, **10**, 304-320 (1960).
2. G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, 'Algorithms for Drawing Graphs: an Annotated Bibliography', *Comput. Geom.* **4**, 235-282 (1994).
3. T. Pisanski et al., *VEGA. Version 0.2. Quick reference manual and Vega graph gallery*, DMFAS, IMFM, Ljubljana, 1995.
4. T. Pisanski et al., *Vega: System for manipulating discrete mathematical structures*, URL: `http:\\vega.www.ijp.si`.
5. T.M.J. Fruchterman and E.M. Reingold, 'Graph Drawing by Force-directed Placement', *Software-Practice and Experience*, **21**, (11), 1129-1164 (1991).
6. P.W. Fowler and D.E. Manolopoulos, *An Atlas of Fullerenes*, Clarendon Press, Oxford, 1995.
7. T. Pisanski and P.W. Fowler, 'Leapfrog Transformation and Polyhedra of Clar Type', *J. Chem. Soc. Faraday Trans.*, **90**, 2865-2871 (1994).
8. K.S. Booth, 'Testing for Consecutive Ones Property, Interval Graphs and Graph Planarity using PQ-Tree Algorithms', *J. Comp. Syst. Sci.*, **13**, 335-379 (1976).
9. J. Marinček, *Kuratowsky Theorem for Surfaces of Small Genus* (in slovene), diploma work, Ljubljana, 1991.
10. H. S. M. Coxeter, *Regular Polytopes*, Dover, New York, 1973.
11. J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, American Elsevier, 1976.
12. T. Pisanski, M. Randić, 'Bridges bewtween Geometry and Graph Theory', submitted.
13. T. Pisanski, B. Plestenjak and A. Graovac, 'NiceGraph Program and Its Applications in Chemistry', *Croat. Chem. Acta*, **68**, 283-292 (1995).
14. D. Wells, *The Penguin Dictionary of Curious and Interesting Geometry*, Penguin Books, London, 1991.
15. T. K. Strempel, *Zur Erzeugung, Klassifizierung und Einbettung kombinatorischer Manningfaltigkeiten und Komplexe* (in german), PhD Thesis, D 17 (Diss TH Darmstadt) Shaker Verlag, Aachen 1996.