



Manipulating Weights to Improve Stress-Graph Drawings of 3-Connected Planar Graphs

Alvin Chiu^(✉), David Eppstein, and Michael T. Goodrich

Department of Computer Science, University of California, Irvine, USA
chiua13@uci.edu

Abstract. We study methods to manipulate weights in stress-graph embeddings to improve convex straight-line planar drawings of 3-connected planar graphs. Stress-graph embeddings are weighted versions of Tutte embeddings, where solving a linear system places vertices at a minimum-energy configuration for a system of springs. A major drawback of the unweighted Tutte embedding is that it often results in drawings with exponential area. We present a number of approaches for choosing better weights. One approach constructs weights (in linear time) that uniformly spread all vertices in a chosen direction, such as parallel to the x - or y -axis. A second approach morphs x - and y -spread drawings to produce a more aesthetically pleasing and uncluttered drawing. We further explore a “kaleidoscope” paradigm for this xy -morph approach, where we rotate the coordinate axes so as to find the best spreads and morphs. A third approach chooses the weight of each edge according to its depth in a spanning tree rooted at the outer vertices, such as a Schnyder wood or BFS tree, in order to pull vertices closer to the boundary.

Keywords: Tutte embedding · convex drawing · vertex spreading

1 Introduction

Sixty years ago, Tutte provided what is arguably one of the first graph drawing algorithms [16] Given a simple, undirected 3-connected planar graph, G , Tutte’s algorithm produces a¹. straight-line, planar drawing of G such that each face is convex. Tutte’s algorithm produces such a drawing of G by solving a set of linear equations that determine the x - and y -coordinates of points to which the vertices of G are assigned. Intuitively, the equations are based on “pinning” the vertices of the outer face of G to the vertices of a convex polygon, and then considering all the edges of G to be springs with an idealized length of 0. Solving

¹ Proofs of Fáry’s Theorem, that any simple, planar graph can be embedded in the plane without crossings so each edge is drawn as a straight line segment, came earlier [7, 15, 17], but these proofs do not give specific coordinates for the vertices; hence, it is not clear they can be called “graph drawing algorithms.”.

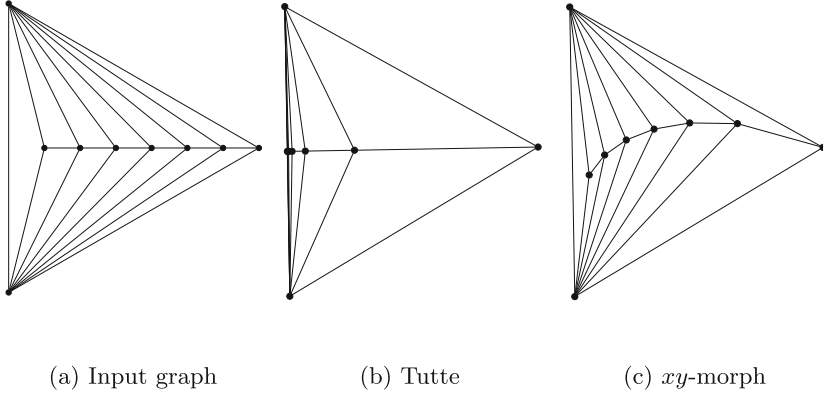


Fig. 1. Tutte drawings can have exponential area.

the set of equations amounts to finding a minimum-energy configuration for the springs given the pinned vertices of the outer face [4, 13].

One unfortunate drawback of Tutte’s algorithm is that it can produce drawings with exponential area or exponentially small edge lengths, depending on the normalization of coordinates. Indeed, Eades and Garvan [5] show that this undesirable result occurs even for the planar graphs formed by connecting two outer vertices to each vertex of a simple path and to each other, as shown in Figs. 1a and 1b. Intuitively, the idealized springs representing graph edges have equal stress, which, in turn, “pull” groups of springs into unsightly vertex clusters.

Hopcroft and Kahn [11] generalize Tutte’s algorithm to spring systems with different stress weights. In this framework, which we explore in this paper, we assign a stress weight, $w_{u,v}$, to each edge, (u, v) , of G .² We begin as in the Tutte framework by “pinning” the vertices of an outer face, f , to be the vertices of a convex polygon, and we then formulate two linear equations for each internal vertex, u , of G , as follows:

$$\sum_{(u,v) \in E} w_{u,v}(x_u - x_v) = 0, \quad \text{and} \quad \sum_{(u,v) \in E} w_{u,v}(y_u - y_v) = 0, \quad (1)$$

where $p_v = (x_v, y_v)$ is the point to which vertex v is assigned. Note that for a vertex, v , on the outer face, we pin $p_v = (x_v^*, y_v^*)$; hence, $x_v = x_v^*$ and $y_v = y_v^*$ are constants in our linear system. As Hopcroft and Kahn [11], as well as Floater [9], show, if the stresses, $w_{u,v}$, are all positive, except possibly for the edges of the outer face, then the resulting drawing is a planar straight-line drawing with each face being convex. In this paper, we experimentally explore the aesthetic improvements to a Tutte embedding that can be achieved by manipulating the stresses in such stress-graph drawings of 3-connected planar graphs.

Related Prior Results. We are not familiar with any prior work on the manipulation of the weights in stress-graph drawings strictly for the purpose of improv-

² Tutte’s approach can be viewed as being for the case when $w_{u,v} = 1$ for each edge.

ing the aesthetic qualities. Nevertheless, the general technique of manipulating stresses in stress-graph drawings is not without precedent. For example, Hopcroft and Kahn [11] and Eades and Garvan [5] give conditions for stresses so that the resulting drawing is the projection of a 3-dimensional convex polyhedron onto the plane. Chrobak, Goodrich, and Tamassia [3] further explore this approach, claiming to produce a 3-dimensional realization of a 3-connected planar graph as the 1-skeleton of a 3-dimensional convex polyhedron with vertex resolution $\Omega(1)$ and with linear volume.³ Indeed, their approach comes close to ours, in that they first compute weights for a weighted Tutte drawing with good vertex resolution (using a flow-based approach) and then apply the Maxwell–Cremona correspondence to lift this drawing to a convex polyhedron. Their method does not necessarily result in aesthetically pleasing drawings or convex polyhedra, despite the good spacing for the x -coordinates. Researchers have also explored interpolating between stress-graph drawings to morph from one layout to another. For example, Floater and Gotsman [10] use interpolation of the weights for two convex embeddings to morph between them, albeit with vertex movements that are represented implicitly. They also devise a method to obtain weights that will produce a given drawing. Erickson and Lin [6] morph between two convex via unidirectional morphs, where vertices move parallel to the direction of an edge. Kleist et al. [12] turn drawings of planar 3-connected graphs into strictly convex planar drawings with similar morphs.

Our Results. We propose several methods of weight manipulation. In the first, we simplify (and correct) the approach of Chrobak, Goodrich, and Tamassia [3] for finding drawings in which vertices have uniformly distributed coordinates. Instead of using iterated flows, we find suitable weights in linear time by counting certain paths in an st -orientation of the graph. Our implementation fixes the outer face as a regular polygon; in the full version on arXiv we show that an alternative choice allows all vertices, including outer face vertices, to have uniform x -coordinates. We experiment with a modified version of this method that produces two planar straight-line drawings that evenly spread the x -coordinates and the y -coordinates, respectively. We then construct a morph that averages the weights of the x - and y -spread drawings. The idea is that this morph will have fairly even spacing on both directions, e.g., as shown in Fig. 1c and Figs. 2a to 2d. We also explore a “kaleidoscope” version of this approach, where we rotate the coordinate axes to find the best spreads. In another approach, we weight edges based on depth in spanning trees rooted at the outer vertices. Edges closer to the outer vertices will have higher weight and thus more “pull”, spreading the internal vertices away from the center of the outer face in a manner that preserves the general structure. We explore two types of spanning trees: BFS and (for fully triangulated graphs) Schnyder woods [1, 8, 14].

³ However, their proof is only valid for polyhedra that have a triangle face.

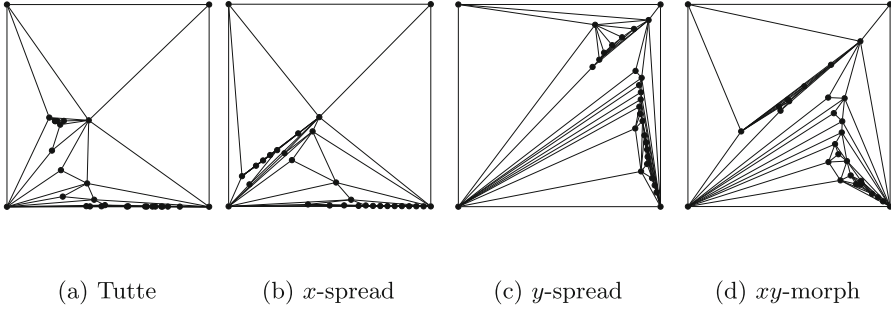


Fig. 2. Drawings of a planar graph with 30 vertices and 80 edges, $G(30, 80)$.

2 Algorithms

Weight Manipulation to Spread Vertices Uniformly. To find weights whose stress-graph embedding spreads vertices evenly, we first begin with an unweighted Tutte drawing, rotating it if necessary so no edge is vertical. We sort the vertices by x -coordinates in this drawing, and orient edges from left to right, producing an st -orientation: an acyclic orientation in which each vertex v_i with $1 < i < n$ has both incoming and outgoing edges. Next, we choose new x -coordinates x_i for the interior vertices that are as evenly spaced as possible under the constraint that they respect the sorted x -ordering of all the vertices. (The same constraint is also present in the flow-based method of Chrobak et al. [3]) We can choose new positive edge weights for the Tutte drawing to produce the chosen x -coordinates in linear time. Conceptually, we gradually increase weights along a sequence of paths in the graph, starting with all weights zero. For each edge e , we find a directed path from v_1 to v_n through e , and increase weights on the edges of this path.

Along a single path through consecutive vertices v_i, v_j, v_k , the spacing between the vertex placements should be in the proportion $x_j - x_i : x_k - x_j$, which can be achieved by giving edges $v_i v_j$ and $v_j v_k$ the weights $1/(x_j - x_i)$ and $1/(x_k - x_j)$ respectively. Because these weights do not depend on the other edges of the path, we can use this weight for each edge in all of the paths that it belongs to and preserve the x -equilibrium. In total, the weight of any edge $v_i v_j$ in the whole graph (summing its weights for each path it appears in) will be $n_{ij}/(x_j - x_i)$, where n_{ij} is the number of paths containing edge $v_i v_j$.

To calculate these numbers efficiently, we compute two spanning trees in the oriented graph: tree T_1 directed out of v_1 , and tree T_n directed into v_n (shortest-path trees via BFS were used for the implementation). For each edge $v_i v_j$, include a path that follows T_1 from v_1 to v_i , then edge $v_i v_j$, then follow T_n from v_j to v_n . We can count the number of these paths that use $v_i v_j$ as follows:

- There is one path defined in this way from $v_i v_j$.

- Let D_j be the set of descendants of v_j in T_1 (including v_j itself) and $d^+(v_k)$ be the number of outgoing edges from v_k . If $v_i v_j$ belongs to T_1 , then $\sum_{v_k \in D_j} d^+(v_k)$ paths pass through $v_i v_j$ in T_1 before crossing to T_n .
- Let A_i be the set of descendants of v_i in T_n and $d^-(v_k)$ be the number of incoming edges at v_k . If $v_i v_j$ belongs to T_n , then symmetrically $\sum_{v_k \in A_i} d^-(v_k)$ paths pass through $v_i v_j$ in T_n after crossing to T_n .

The sums of descendant out-degrees in T_1 , and of descendant in-degrees in T_n , can be computed in linear time by a simple bottom-up tree traversal, after which we can calculate the weight $n_{ij}/(x_j - x_i)$ of all edges in linear time. A weighted Tutte drawing with positive weights and convex outer face cannot introduce crossings, so we get a convex drawing with spread out x -coordinates using these new weights. To spread by a different direction, we can rotate the initial unweighted Tutte drawing before doing the spread. Indeed, as we explore experimentally, we consider a number of distinct rotation angles, producing drawings similar to the way a kaleidoscope produces patterns as it is turned.

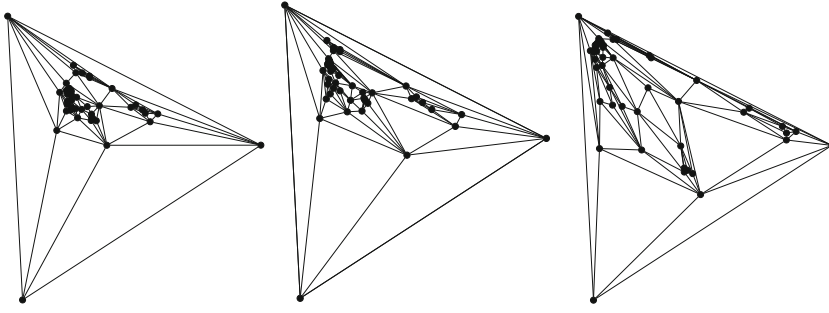
Moreover, we can produce an “ xy -morph” drawing of the input graph. Let a weighted Tutte drawing be represented by $\Gamma = (\Lambda, \mathcal{P})$, where Λ is the coefficient matrix containing the edge weights and $\mathcal{P}$ is the convex polygon chosen to be the outer face. One can morph between the x -coordinate spread drawing $\Gamma_0 = (\Lambda_0, \mathcal{P})$ and y -coordinate spread drawing $\Gamma_1 = (\Lambda_1, \mathcal{P})$ to obtain a more balanced graph drawing $\Gamma_{1/2}$. Intuitively, this is like stopping halfway in Floater and Gotsman’s morphing algorithm [10], where we construct $\Gamma_{1/2} = (\Lambda_{1/2}, \mathcal{P})$ where $\Lambda_{1/2} = \frac{1}{2} \cdot \Lambda_0 + \frac{1}{2} \cdot \Lambda_1$. (See Fig. 2.)

Weight Manipulation via Spanning Tree Depth. In our spanning-tree approach, we first do a Tutte drawing, then we find a set of edge-covering spanning trees, T , for the graph rooted at the outer vertices, such as BFS trees or Schnyder woods [1, 8, 14]. Next, we assign weights to the edges of each tree, T , in a top-down manner according to its depth in the spanning tree. With these new weights, we do another stress-graph drawing.

Let the *depth* of an edge in a tree be the number of edges from the root to the edge plus one (to include the edge itself). Then we assign an edge at depth i with weight a/r^i , where a is some initial constant and r is a scaling parameter. When using BFS to find the shortest-path tree T_v rooted at an outer vertex v , we assign weights to an edge according to its lowest depth from any of the outer vertices. To do this, we create a dummy “super”-vertex, v_s , connected to all the outer vertices and run BFS from v_s , which is akin to running BFS on all the outer vertices simultaneously. For the case when the outer face is a triangle, we also consider Schnyder woods, which form an edge-covering set of three spanning trees that have nice “flow” properties [1, 8, 14]. (See Fig. 3.)

3 Experiments

Our experimental setup modifies the Open Graph Drawing Framework (OGDF) C++ library [2]. One of our goals is to compare our weight manipulation methods against Tutte’s algorithm, which often produces exponentially small edge



(a) Tutte (b) Schnyder-spread, $r = 5$ (c) BFS-spread, $r = 5$

Fig. 3. Drawings of a pseudorandom graph, $G(50, 144)$.

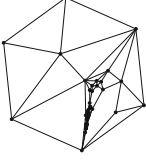
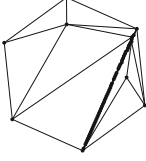
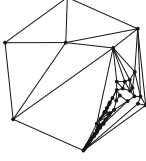
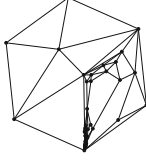
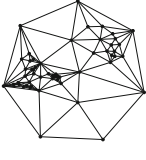
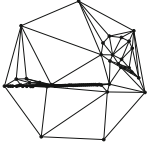
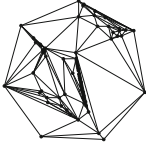
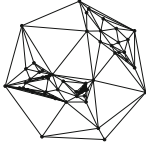
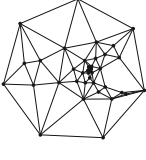
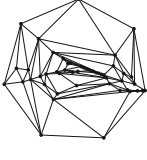
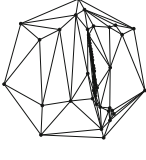
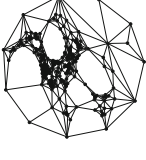
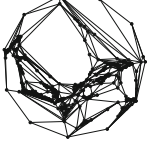
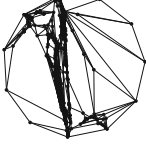
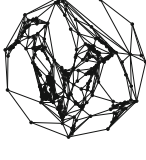
lengths. Thus, the main metric we use is the *edge-length ratio* $\rho(\Gamma)$ of drawing Γ , which is the longest edge length divided by the smallest edge length in the drawing. In Table 1, we compare the edge-length ratios of the Tutte embeddings of several pseudorandom planar graphs against the x -spread, the y -spread, the xy -morph between the previous two, and the BFS-spread. For the BFS-spread, we choose the parameter r to be the integer that minimizes the edge-length ratio $\rho(\Gamma)$. We do not show the results for the Schnyder-spread, as they were almost always worse than the BFS-spread.

Not surprisingly, our testing demonstrates that the x -spread and y -spread drawings achieve edge-length ratio close to the number of vertices, n , because of the uniform vertex spacing that they produce. Nevertheless, optimizing exclusively for edge-length ratio can result in vertices that cluster close to a straight line as can be seen in Table 1. In contrast, the xy -spread drawing often is more aesthetically pleasing, as it tends to have better symmetry visualization than either the x - or y -spread drawings without clustering. However, it usually results in higher edge-length ratio than either of the two drawings it morphs. It may even have a higher edge-length ratio than its corresponding Tutte drawing, as seen by the xy -morph for $G(50, 130)$ in Table 1.

The edge-length ratio of BFS-spread drawings tends to be smaller than Tutte embeddings, while still preserving those drawings' general structure and symmetry visualization.

We also experimented with a “kaleidoscope” drawing paradigm, where we rotate the x - and y -axes by small angular increments and compute an xy -morph for each angle. The edge-length ratios can vary dramatically in such drawings, so the minima offer good choices. We show an example plot of edge-length ratios in Fig. 4, with its worst and best rotations in Table 2. More figures can be found in the full version on arXiv: <https://arxiv.org/abs/2307.10527>.

Table 1. Drawing Gallery. $\rho(\Gamma)$ is the edge-length ratio, r is the scaling parameter.

	Tutte	x -spread	y -spread	xy -morph	BFS-spread
$G(60, 150)$					
	$\rho(\Gamma) = 52717$	$\rho(\Gamma) = 86$	$\rho(\Gamma) = 63$	$\rho(\Gamma) = 2074$	$\rho(\Gamma) = 16156, r = 3$
$G(100, 200)$					
	$\rho(\Gamma) = 3973$	$\rho(\Gamma) = 43$	$\rho(\Gamma) = 71$	$\rho(\Gamma) = 229$	$\rho(\Gamma) = 380, r = 3$
$G(70, 200)$					
	$\rho(\Gamma) = 1165$	$\rho(\Gamma) = 42$	$\rho(\Gamma) = 69$	$\rho(\Gamma) = 216$	$\rho(\Gamma) = 283, r = 3$
$G(50, 130)$					
	$\rho(\Gamma) = 766$	$\rho(\Gamma) = 31$	$\rho(\Gamma) = 31$	$\rho(\Gamma) = 1020$	$\rho(\Gamma) = 272, r = 3$
$G(400, 1100)$					
	$\rho(\Gamma) = 56162$	$\rho(\Gamma) = 538$	$\rho(\Gamma) = 414$	$\rho(\Gamma) = 5054$	$\rho(\Gamma) = 9092, r = 2$

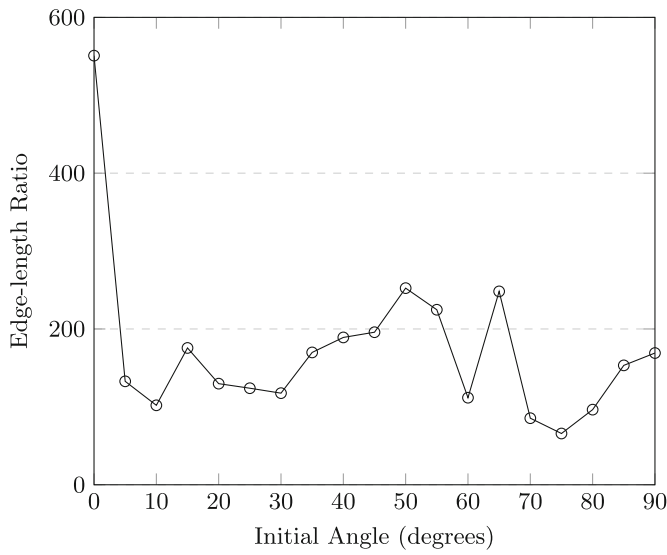
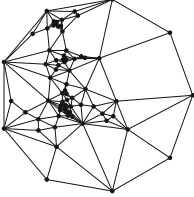
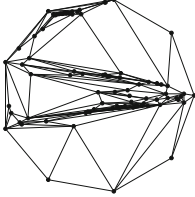
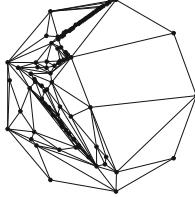
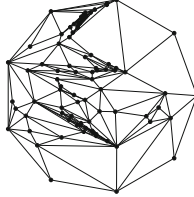
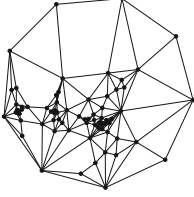
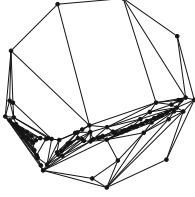
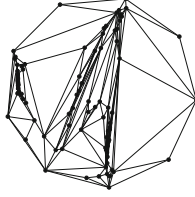
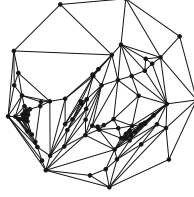


Fig. 4. Edge-length ratios for kaleidoscope xy -morphs for $G(90, 240)$, increments of 5°C .

Table 2. Worst and best rotations for the graph of Fig. 4.

	Tutte	x -spread	y -spread	xy -morph
$\alpha = 0^\circ$				
$\alpha = 75^\circ$				

Acknowledgements. This research was supported in part by NSF grant CCF-2212129.

References

1. Bonichon, N., Felsner, S., Mosbah, M.: Convex drawings of 3-connected plane graphs. *Algorithmica* **47**(4), 399–420 (2007)
2. Chimani, M., Gutwenger, C., Jünger, M., Klau, G., Klein, K., Mutzel, P.: The open graph drawing framework (OGDF). In: *Handbook of Graph Drawing and Visualization*, pp. 543–569. CRC Press (2013)
3. Chrobak, M., Goodrich, M.T., Tamassia, R.: Convex drawings of graphs in two and three dimensions. In: *12th Symposium on Computational Geometry (SoCG)*, pp. 319–328. New York, NY, USA (1996). <https://doi.org/10.1145/237218.237401>
4. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Hoboken (1999)
5. Eades, P., Garvan, P.: Drawing stressed planar graphs in three dimensions. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 212–223. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021805>
6. Erickson, J., Lin, P.: Planar and toroidal morphs made easier. In: Purchase, H.C., Rutter, I. (eds.) *GD 2021*. LNCS, vol. 12868, pp. 123–137. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92931-2_9
7. Fáry, I.: On straight-line representation of planar graphs. *Acta Scientiarum Mathematicarum* **11**(2), 229–233 (1948)
8. Felsner, S.: Lattice structures from planar graphs. *The Electronic Journal of Combinatorics*, pp. R15–R15 (2004)
9. Floater, M.S.: Parametric tilings and scattered data approximation. *Int. J. Shape Model.* **04**(03n04), 165–182 (1998). <https://doi.org/10.1142/S021865439800012X>
10. Floater, M.S., Gotsman, C.: How to morph tilings injectively. *J. Comput. Appl. Math.* **101**(1), 117–129 (1999). [https://doi.org/10.1016/S0377-0427\(98\)00202-7](https://doi.org/10.1016/S0377-0427(98)00202-7), <https://www.sciencedirect.com/science/article/pii/S0377042798002027>
11. Hopcroft, J.E., Kahn, P.J.: A paradigm for robust geometric algorithms. *Algorithmica* **7**(1–6), 339–380 (1992)
12. Kleist, L., Klemz, B., Lubiw, A., Schlipf, L., Staals, F., Strash, D.: Convexity-increasing morphs of planar graphs. *Comput. Geom.* **84**, 69–88 (2019)
13. Kobourov, S.G.: Spring embedders and force directed graph drawing algorithms. *arXiv preprint arXiv:1201.3011* (2012)
14. Schnyder, W.: Embedding planar graphs on the grid. In: *1st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 138–148 (1990)
15. Stein, S.K.: Convex maps. *Proc. Am. Math. Soc.* **2**(3), 464–466 (1951)
16. Tutte, W.T.: How to draw a graph. *Proc. Lond. Math. Soc.* **3**(1), 743–767 (1963)
17. Wagner, K.: Bemerkungen zum Vierfarbenproblem. *Jahresber. Deutsch. Math.-Verein.* **46**, 26–32 (1936)