Graph-Theoretic Solutions
to Computational Geometry Problems

David Eppstein

Computer Science Department, University of California, Irvine

Abstract. Many problems in computational geometry are not stated in graph-theoretic terms, but can be solved efficiently by constructing an auxiliary graph and performing a graph-theoretic algorithm on it. Often, the efficiency of the algorithm depends on the special properties of the graph constructed in this way. We survey the art gallery problem, partition into rectangles, minimum-diameter clustering, rectilinear cartogram construction, mesh stripification, angle optimization in tilings, and metric embedding from this perspective.

1 Introduction

Graph algorithms and computational geometry form separate communities with separate conferences such as the International Workshop on Graph-Theoretic Concepts in Computer Science and the ACM Symposium on Computational Geometry, respectively, but they also meet in broader algorithms conferences, and there has been much interplay between the research topics in the two areas.

Many classical graph algorithm problems have geometric analogues, algorithmic problems on graphs defined by a geometric input. In most cases, problems of this type can be solved directly by constructing the graph and then applying a general-purpose graph algorithm, but can be sped up by examining the graph algorithm's structure more closely and applying appropriate geometric data structures. A notable instance of this phenomenon is the Euclidean minimum spanning tree (the spanning tree of a complete graph in which the vertices represent points and edge lengths are Euclidean distances): by using a Delaunay triangulation in place of a complete graph, the quadratic time of a naive algorithm can be improved to $O(n \log n)$ [67]. Other work along these lines includes algorithms for Euclidean matching [1, 71] and bipartiteness testing [26].

Graph drawing, the visualization of graphs via geometric graph representations [19, 45, 59], forms another community represented by the annual International Symposium on Graph Drawing. Most work in the area concerns drawings in which a graph's vertices are represented as geometric points, disks, or polygons, and its edges are represented as line segments or curves. Researchers in this area seek algorithms that optimize mathematical stand-ins for their aesthetic quality and readability such as the number of crossings, the number of bends in non-straight edges, the angles formed by adjacent edges, the area of the drawing, or the amount of symmetry that the drawing displays; the interplay between these different measures provides much scope for research.

Geometric techniques have sometimes also been applied to solve problems that were originally defined in purely graph-theoretic terms. One instance concerns parametric

minimum spanning trees, minimum spanning trees of graphs whose edge weights are linear functions of a parameter. Dey proved [18] that $O(mn^{1/3})$ different trees may be formed in this way from a parametric graph with m edges and n vertices, using the *crossing number inequality* that s simple plane curves with n shared endpoints have $\Omega(s^3/n^2)$ crossings [4, 50], while the best known lower bound, $\Omega(m\alpha(n))$, involves a reduction from another geometric problem, lower envelopes of line segments [22].

In this paper we survey connections between computational geometry and graph algorithms of yet another type: problems in computational geometry that, although not initially defined graph-theoretically, may be solved by constructing an auxiliary graph from the input, applying a purely graph-theoretic algorithm to this auxiliary graph, and translating the output of this algorithm back into geometric terms. We discuss problems including the art gallery problem, partition into rectangles, minimum-diameter clustering, rectilinear cartogram construction, mesh stripification, angle optimization in tilings, and metric embedding. The ordering of the problems is roughly chronological, and the selection of topics is idiosyncratic and (especially for the later problems) largely drawn from the author's own research rather than being comprehensive.

2   Art Gallery Problems

Most computer scientists are familiar both with Chvátal's *art gallery theorem* [13, 61] that every n-vertex simple polygon has a set of $\lfloor n/3 \rfloor$ *guard points* from which the whole polygon may be seen, and with Fisk's elegant graph-theoretic proof [36]. One begins the proof by adding a maximal set of non-crossing diagonals to the polygon, partitioning it into triangles. Graph-theoretically, the vertices, sides, and added diagonals of the polygon form a maximal outerplanar graph; the *weak dual* of this graph (the adjacency graph of the triangles, omitting the outer face) is a tree (Figure 1, center). Every maximal outerplanar graph may be colored with three colors, as may be proven by induction: the result is clearly true when the graph is a triangle, and any larger maximal planar graph may be colored by removing a leaf from the dual tree, coloring the remaining graph, restoring the leaf, and observing that the restored vertex has only two neighbors and therefore has a free color available. Each triangle has one vertex of each color, so each color class forms a valid guard set, and the smallest of the color classes has at most $\lfloor n/3 \rfloor$ vertices. This proof technique translates into an efficient algorithm: polygons may be triangulated in linear time [10], and the induction proof leads to a linear-time algorithm for 3-coloring maximal planar graphs.

A less well-known variant of the art gallery problem concerns *simple orthogonal polygons*, simple polygons all of whose sides are parallel to the coordinate axes. Simple orthogonal polygons need only $\lfloor n/4 \rfloor$ guards, as can be shown again by graph coloring. Every simple orthogonal polygon can be partitioned by diagonals into convex quadrilaterals [46]; the resulting tree of quadrilaterals can be viewed as a special type of *squaregraph*, a planar graph in which every bounded face has four sides and every vertex either belongs to the unbounded face or has at least four incident edges [12, 6]. Adding the two diagonals of each quadrilateral to a squaregraph forms a *kinggraph* [12] (Figure 1, right). As with maximal outerplanar graphs, the kinggraphs derived from simple orthogonal polygons may be shown to be 4-chromatic by removing leaves of the
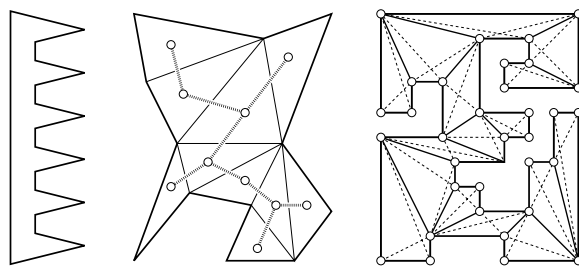
Fig. 1. The art gallery problem. Left: a comb polygon requiring $\lfloor n/3 \rfloor$ guards. Center: triangulating an input polygon produces a maximal outerplanar graph, the weak planar dual of which is a tree. Right: The kinggraph formed by adding diagonals (dashed) to a partition of an orthogonal polygon into convex quadrilaterals.

dual tree.[1] In any four-coloring, each quadrilateral has a vertex of each color, so each color class forms a guard set, and the smallest color class has at most $\lfloor n/4 \rfloor$ guards. This bound is tight: a comb-shaped orthogonal polygon requires $\lfloor n/4 \rfloor$ guards, just as a comb-shaped simple polygon requiring $\lfloor n/3 \rfloor$ guards shows that the bound for the standard art gallery problem is tight (Figure 1, left).

The biggest remaining open problem in art gallery theory concerns edge guards: how many edges must one select, from a simple polygon, so that every point of the polygon may be seen from some point on some selected edge? It is not clear, in this case, what graph one should define on the edges of the polygon to force the color classes of a coloring of the graph to form edge guard sets.

3 Partition into Rectangles

Many geometric algorithms take as input a complicated polygonal domain and cover or partition it using simpler shapes [65]; the partitions into triangles and quadrilaterals from the previous section are important examples. Another problem of this type concerns the partition into rectangles of an orthogonal polygon. The input polygon may have polygonal holes, unlike the simple orthogonal polygons of the previous section, and the rectangles are not required to meet edge-to-edge and vertex-to-vertex. The goal is to minimize the total number of rectangles in the partition (Figure 2).

Rectangular partitions have many applications. In VLSI design, it is necessary to decompose masks into the simpler shapes available in lithographic pattern generators [62], and similar mask decomposition problems also arise in DNA microarray design [39]. Rectangular partitions can simplify convolution operations in image processing [35] and can be used to compress bitmap images [11]. Closely related matrix decomposition problems have been applied to radiation therapy planning [21,47], and rectangular partitions have also been used to design robot self-assembly sequences [51].

Define a *good diagonal* to be an axis-parallel line segment interior to the input polygon that connects two concave vertices of the polygon. As several authors independently

---

[1] More generally, every kinggraph is 4-chromatic, but a proof is beyond the scope of this survey.
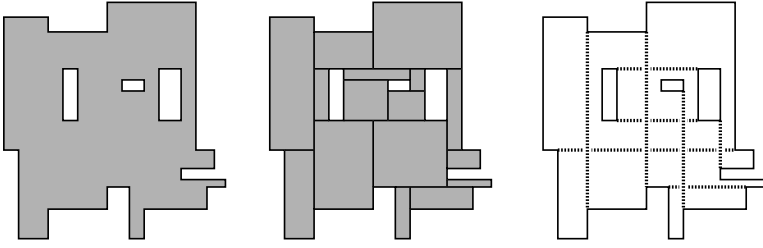
Fig. 2. Partitioning an orthogonal polygon (left) into the minimum number of rectangles (center). The right figure shows the axis-parallel diagonals that connect pairs of concave vertices; the rectangle partition problem may be solved by finding a maximum independent set in the bipartite intersection graph of these diagonals.

discovered [35, 55, 60], the minimum number of rectangles in a partition of a polygon with n vertices and h holes is $n/2 + h - g - 1$, where g is the maximum size of a set of disjoint good diagonals. To see this, consider the number of corners of rectangles in a partition, four times the number of rectangles. Let G be a maximal set of disjoint good diagonals that form a subset of the line segments in some partition of an orthogonal polygon, and define a *bad vertex* to be a nonconvex polygon vertex that is not an endpoint of G. Each polygon vertex forms at least one rectangle corner. Additionally, for each bad vertex v, let s be an interior line segment of the partition having v as an endpoint. Then either s crosses a segment in G, and the two corners formed by the crossing that are on the same side as v with respect to the crossed segment can be charged to v, or s ends at a non-vertex and the two corners formed at its other endpoint can be charged to v. This charging scheme shows that the number of rectangle corners in the partition is at least n plus twice the number of bad vertices. The number of nonconvex vertices in any orthogonal polygon with h holes is $n/2 + 2h - 2$, so the number of bad vertices is $n/2 + 2h - 2|G| - 2$, and a lower bound of $n/2 + h - |G| - 1$ rectangles follows. Conversely, if G is any set of disjoint good diagonals, a partition with exactly $n/2 + h - |G| - 1$ rectangles may be found by considering the bad vertices for G, one at a time, and extending a line segment from each bad vertex to the closest previously-added segment or polygon side. Thus, finding a partition into a minimum number of rectangles is equivalent to finding a maximum number of disjoint good diagonals.

The intersection graph of the good diagonals is bipartite: each horizontal diagonal intersects only vertical diagonals and vice versa. Therefore, finding the maximum number of disjoint good diagonals translates, in graph-theoretic terms, into finding a maximum independent set in a bipartite graph. By König's theorem [49], in any n-vertex bipartite graph the maximum independent set has size $n - M$, where M is the cardinality of a maximum matching; an independent set of this size may be found from a maximum matching by partitioning the vertices according to the lengths of the shortest alternating paths from an unmatched vertex to the given vertex, and including the vertices at even levels of this partition. Therefore, the maximum independent set of the intersection graph, the corresponding maximum set of disjoint good diagonals, and a partition into a minimum number of rectangles, may all be found in polynomial time.

A naive implementation of an algorithm that reduces the problem to a bipartite graph and then applies a general-purpose bipartite graph matching algorithm would solve the problem in time $O(n^{2.5})$, where n denotes the number of vertices of the input polygon: this is the time needed to apply the Hopcroft–Karp matching algorithm [41] to the intersection graph of the good diagonals, which may have $\Theta(n)$ vertices and $\Theta(n^2)$ edges. However, by using geometric range searching data structures to speed up the search for alternating paths within the matching algorithm, it is possible to improve the overall running time to $O(n^{3/2}\log n)$ [42, 53, 54].

4 Minimum Diameter Clustering

The *diameter* of a set of points is the maximum distance of any two of its points. The problem of finding low-diameter subsets of larger sets of input points [3, 27] may be formulated in several different ways: one may take as input a number k and produce as output the subset of k points with minimum diameter, one may take as input a number D and produce as output the largest subset with diameter at most D, or one may solve the *decision problem* in which D and k are both given and the task is to determine whether there exists a set of k points with diameter at most D. Since there are only $O(n^2)$ potential diameter values and n different values of k, an efficient algorithm for any one of these tasks leads to efficient algorithms for the other two tasks as well.

Finding the largest size k of a set with diameter at most a given value D has a direct translation to a graph theoretic problem, but the very directness of the translation means that it is unhelpful: it provides merely a restatement of the problem rather than conveying any new insight. We may scale the input point set so that $D = 2$; then what we seek is the largest clique in a *unit disk graph*, the intersection graph of unit disks centered at the points (Figure 3, left); note that a set of disks forming a clique need not have a common intersection point. Maximum cliques in unit disk graphs may be found in polynomial time, given a disk representation of the graph [15] but this is just a trivial restatement of the minimum diameter clustering problem; it is NP-hard to find a disk representation given only a graph-theoretic description of a unit disk graph [8].

However, just as in the rectangle partition problem, minimum diameter clustering may be reduced to a more basic graph-theoretic problem, maximum independent sets in bipartite graphs [3, 15]. Suppose that we know or can guess the two points p and q forming the diametral pair in a cluster. Then, all the other points of the cluster must be within the *lune* formed by intersecting the two circles with pq as radius, centered on p and q. The largest possible cluster within the lune is the maximum clique of the unit disk graph with disk centers inside this lune, or equivalently the maximum independent set of the complement G of this restricted unit disk graph. But G is bipartite: if the lune is bisected by line pq, then a point in G can be connected by an edge only to other points on the other side of the bisection line (Figure 3, right). Therefore, its maximum independent set may be found in polynomial time, as discussed in the previous section.

Based on this idea, one could test all $\Theta(n^2)$ pairs pq, find the bipartite graph derived from each pair and apply a bipartite matching algorithm to it, and return the best cluster found from all of these separate tests. However it is more efficient to use dynamic
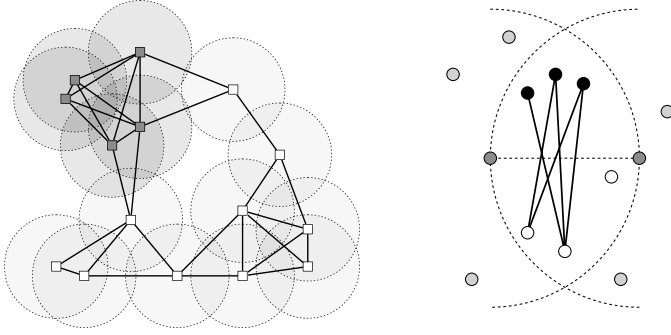
Fig. 3. Minimum diameter clustering. Left: a unit disk graph and its largest clique (the five darker circles in the upper left). Right: for any two specified points, the largest cluster having those two points as its diametral pair may be found as a maximum independent set of a bipartite graph in which the two sides of the bipartition are two halves of a lune defined by the two specified points.

graph algorithms to share work between multiple different matching problems [27]. Suppose we seek the size of the largest cluster for a given D. For each point p that could be an endpoint of a diametral pair, consider the lune defined by a segment of length D with p as one endpoint. If the defining segment rotates through an angle of 2π around p, the lune rotates with it, and the set of input points inside the lune undergoes a sequence of $O(n)$ discrete changes in which some point joins or leaves the set. After each change, one may update the maximum matching of the bipartite graph defined from the lune by a single alternating path search. Thus, the overall algorithm loops through all n possible choices of p, and performs a nested loop through the $O(n)$ set insertion and deletion operations defined by rotating a size-D lune around p. For each set update operation the algorithm performs an alternating path search to update a maximum matching and the maximum independent set in the bipartite graph defined from the lune, and when the nested loops terminate the algorithm returns the largest cluster found in each of these searches. As Aggarwal et al. [3] describe, each step of an alternating path search may be performed in logarithmic time with the aid of the circular hull data structure of Hershberger and Suri [40]. Therefore, each alternating path search takes time $O(n \log n)$, the sequence of $O(n)$ alternating path searches for a single choice of p takes time $O(n^2 \log n)$, and the overall clustering algorithm takes time $O(n^3 \log n)$ [27].

If k rather than D is given as input, the problem may be solved by a binary search among the $O(n^2)$ different distances defined by the input points, that checks for each distance whether it is the diameter of some k-point cluster. For this variant, the time may be further improved to $O(n \log n + k^2 n \log^2 k)$ by using a k-nearest-neighbor calculation to reduce the problem to $O(n/k)$ subproblems with $O(k)$ points per subproblem [27].

It appears to be open whether there exist polynomial time algorithms for solving the analogous minimum diameter clustering problems in higher dimensions (equivalently, finding maximum cliques in intersection graphs of unit balls in higher dimensions) or for finding a maximum clique in a unit disk graph when a geometric representation of the graph is not given.
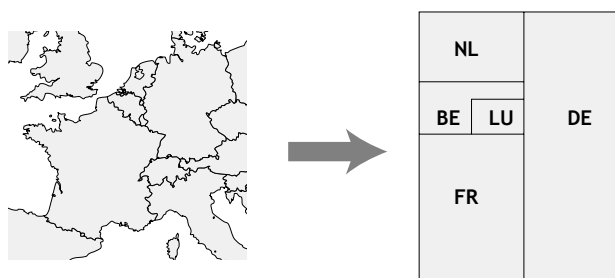
Fig. 4. A rectilinear cartogram of five countries in northwestern Europe. Map based on a CC-BY-SA licensed image by Brianski, Canuckguy, Zaparojdik, Madman2001, Roke, and Ssolbergj, online at http://commons.wikimedia.org/wiki/File:Blank_Map_of_Europe_-w_boundaries.svg.

5   Bend Minimization

A rectilinear cartogram [64, 72, 58] is a diagram in which geographic regions have been replaced by orthogonal polygons, with approximately the same shapes and in approximately the same positions with respect to each other as they hold geographically, but in which the areas of the regions have been distorted to represent numerical data about the regions unrelated to their physical areas. In introducing these diagrams in 1934, Raisz [64] wrote, "it should be emphasized that the statistical cartogram is not a map," and the stylization inherent in using orthogonal polygons helps perform this emphasis.

We have studied algorithms for constructing cartograms that can accomodate arbitrary area assignments [31] and in which the adjacencies between regions have desired orientations [30] but a more basic problem, constructing a cartogram in which the region shapes are as simple as possible, had already been solved in a different context, that of graph drawing [69]. Simple shapes such as rectangles aid the viewer in measuring and comparing areas and hence in understanding the data represented by a cartogram. We would like to find a cartogram that represents a given map with a minimum number of *bends* (corners interior to the boundary between two adjacent regions); for instance, in Figure 4 there is one bend, between Belgium and Luxembourg; this is optimal, as Luxembourg is surrounded by only three countries and hence must have at least one bend on its boundary. As Tamassia [69] shows, this problem of bend minimization can be solved in polynomial time by translating it into a network flow problem.

To form a flow problem that represents the bends of some given cartogram, construct a graph that has a single "circulation" vertex, a vertex for each region of the cartogram (including the exterior of the diagram as one of the regions), and a vertex for each junction where three or four regions of the cartogram meet (four regions at a junction may be needed to model places like the Four Corners in the southwest of the U.S. where four states meet, but five or more regions cannot meet at a single junction in an orthogonal cartogram). Include edges from the circulation vertex to each other vertex, between each two vertices that represent adjacent regions, and between two vertices that represent an incident junction-region pair. Label this graph with flow amounts, as follows: each junction vertex has four incoming units of flow from the circulation

vertex, and sends one unit of flow to the vertices representing the regions in the four quadrants incident to the junction. For each bend in the cartogram, send one unit of flow from the region that has a convex vertex at the bend to the region that has a concave vertex. For each interior region having k junctions on its boundary, send $4 - 2k$ units of flow to the circulation vertex (or equivalently, if $4 - 2k < 0$, send $2k - 4$ units from the circulation vertex to the region vertex). And for an exterior region with k junctions on its boundary, send $2k + 4$ units of flow to the circulation vertex. The result can be shown to be a valid circulation: that is, for each vertex of the flow graph, the numbers of incoming and outgoing flow units are equal. For the junction vertices, this is clear because the four incoming units are balanced by the four quadrants into which a unit of flow is sent. An interior region with k junctions and no bends must form a rectangle, as a junction cannot form a concave corner; thus, it has four incident junction vertices that send a single unit of flow, and the remaining $k - 4$ incident junctions send two units of flow into the region, for a total of $2k - 4$ incoming units, balancing the flow to the circulation vertex. Each additional cave corner at a bend causes one unit of incoming flow from another region, but must be balanced by an additional convex corner; if that convex corner belongs to a bend, it provides a unit of outgoing flow, and if it belongs to a junction then that junction sends one fewer unit of incoming flow, in either case leading to the same total flow balance. A similar argument shows that the flow into and out of the exterior region vertex is also balanced, from which it follows that the flow must be balanced at the single remaining vertex, the circulation vertex.

Conversely, as Tamassia shows, one can assign costs and flow capacities to this network in order to force a minimum cost circulation to correspond to a minimum-bend drawing. Capacity constraints are needed to force the incoming flow to each junction vertex to be exactly four units; the edges between adjacent region vertices are assigned unit cost, and all other edges have zero cost. With these constraints and costs, the flow described above has cost equal to its number of bends. Conversely, any integer solution to the capacitated circulation problem can be translated into a drawing in which the total number of bends is equal to the cost of the circulation, so a cartogram with the minimum number of bends can be found from a minimum-cost circulation, which can be constructed in polynomial time. The flow graph is an *apex graph*: if one vertex, the circulation vertex, is removed, the rest is planar. Therefore, specialized techniques for finding flows in planar graphs may be used to speed up this algorithm [38, 43, 57].

6 Mesh Stripification

Stripification refers to the problem in computer graphics of partitioning a triangulated surface model of a three-dimensional object into *triangle strips*, sequences of triangles meeting edge-to-edge [5, 29, 34, 73]. Such a partition allows for fast rendering by requiring the coordinates of only one vertex per triangle to be transmitted to the graphics hardware; the other two vertex locations may be found from the previous triangle in the strip. Triangle strips aid in data compression of geometric models: a predicted location that aids in compressing the coordinates of each successive vertex may be found by extrapolation from the previous triangle in a strip [17]. Additionally, if a mesh can be
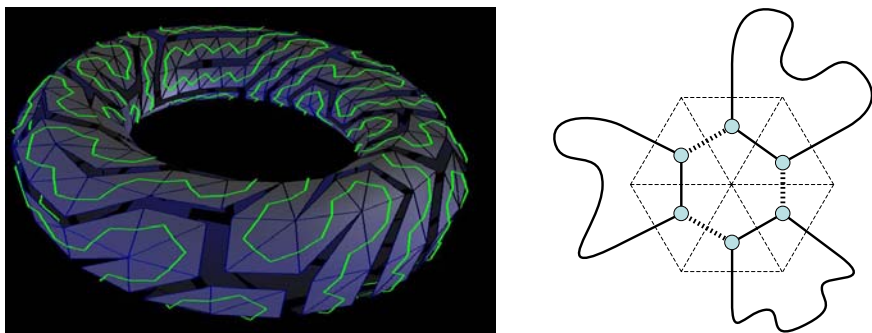
Fig. 5. Left: a single strip in a triangulated torus, rendered by M. Gopi, from [25]. Right: a local move at a vertex allows three cycles to be merged into a single cycle.

represented as a single cyclic triangle strip, as in Figure 5 (left), the structure of the strip may be used to cover the surface of the model with a space-filling curve [29], and contractions of the boundary edges of the strip can be used for topology-preserving simplifications of the model [20].

Finding such a strip may be viewed graph-theoretically, as finding a Hamiltonian cycle in the 3-regular dual graph of the triangulation, a graph with one vertex per triangle and one edge for each pair of adjacent triangles. However, this is problematic for two reasons. First, the Hamiltonian cycle problem is NP-complete and the known exponential-time algorithms for the problem [25, 44] are only capable of solving it within a reasonable time for models of at most a few hundred triangles. Second, and more importantly, not all triangulated models, even with spherical topology, have single triangle strips of this type. Tutte's counterexample to Tait's conjecture that 3-regular polyhedra are Hamiltonian [68, 70] dualizes to become a triangulated mesh that cannot be represented as a single cycle of triangles.

However, it is important to realize that the dual graph does not uniquely represent the geometry: we may change the triangulation, and hence change the graph, without changing the model's shape. The dual graph of any triangulated model is both 3-regular and bridgeless; therefore, by a theorem of Petersen [63], it has a perfect matching, which may be found efficiently [7]. The set of edges complementary to the matching forms a partition of the triangulation into a collection of disjoint cycles [29]; however, there may be more than one of these cycles. As we observed [29], in many cases a local move at a vertex of the triangulation, that swaps matched and unmatched edges connecting the triangles sharing that vertex, may reduce the number of cycles (Figure 5, right). If no such move is available, two adjacent triangles from two different cycles may be bisected across their shared edge, leading to a new triangulation of the same surface with the property that a cycle-reducing local move is available at the newly created vertex. By repeating this subdivision process, one eventually reaches a triangulation that covers a surface identical to the input model, but one that has a different dual graph than the input and that can be represented as a single strip. Although theoretically this could increase the total number of triangles by as much as a factor of $3/2$, in practice we saw at most a 2% increase.
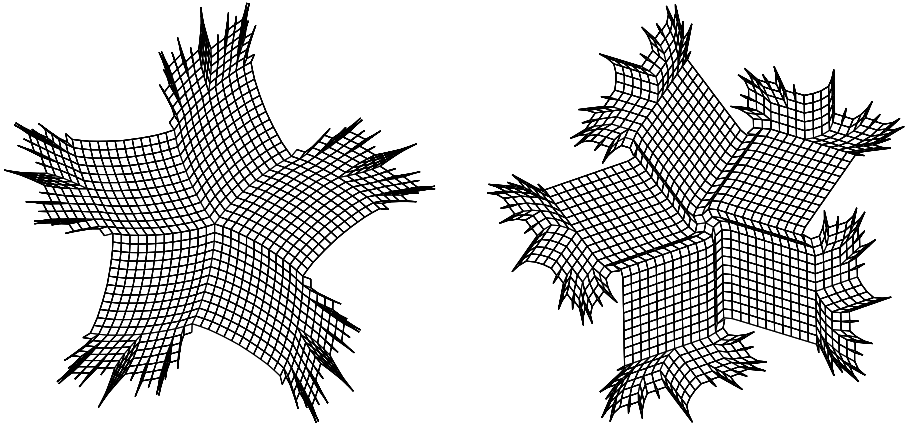
Fig. 6. Two drawings of the squaregraph dual to a 5-chromatic triangle-free circle graph described by Ageev [2], from [6]

7   Angle Optimization of Tilings

The planar dual of any arrangement of lines in the plane may be represented as a *tiling* of a convex subset of the plane by centrally symmetric convex polygons; this tiling has one polygon for each crossing point of the arrangement, which may be chosen to have unit-length edges perpendicular to the lines crossing at that point. De Bruijn [16] used this construction to form aperiodic tilings (including the rhombic Penrose tiling) from overlaid families of evenly-spaced parallel lines. More generally the dual of a *weak pseudoline arrangement* (a set of curves that are the image of a line under a homeomorphism of the plane, that have at most one point of intersection per pair of curves, and that cross each other at their intersection points), may be represented as a tiling of a nonconvex simple polygon by centrally-symmetric polygons, and every such tiling arises in this way [24, 28]; this provides a convenient method for visualizing squaregraphs, which are the duals of triangle-free hyperbolic line arrangements [6].

However, the tilings obtained directly from this construction may be hard to read due to having polygons with very sharp angles, as is true for instance for the tiling in Figure 6 (left). In unpublished work with Kevin Wortman [32], we considered the problem of finding a combinatorially equivalent tiling that maximizes the minimum angle in the tiling (the so-called angular resolution [9, 56]), such as the one in Figure 6 (right); we showed that this optimal tiling could be found by a parametric shortest path computation in an auxiliary graph derived from the input.

Our algorithm constructs a graph in which the edges have weights that are linear functions of a parameter λ that represents the angular resolution of the drawing. In a tiling of this type, one can define an equivalence relation on the sides of tiles in which opposite pairs of sides on the same side are equivalent; the equivalence classes form *zones* of line segments that are required to be parallel and to have the same length. Our graph has one vertex per zone, and an additional start vertex that has a zero-length edge to each other vertex; the distance from the start vertex to a zone's vertex represents the

adjustment in angle for the segments in the zone between an initial tiling and the optimal tiling. The constraint that the angle between two sides of a polygon be at least λ can then be expressed by the existence of an edge between the vertices representing the zones containing the two sides, with length $\theta_i - \theta_j - \lambda$, where θ_i and θ_j are the angles formed by the two zones in the initial tiling. Similarly, the constraint that an interior angle be convex can be expressed by an edge with length $\pi + \theta_i - \theta_j$. With these vertices and edge lengths, it can be shown that there exists a tiling with angular resolution at least λ if and only if the result of substituting that value into each edge weight function is a graph with no negative cycles. Therefore, the optimal angular resolution is the largest value of λ giving no negative cycles. Due to the special form of the weights in the parametric graph (each weight is either constant or a constant minus λ) this parametric negative cycle detection problem can be solved in $O(n^3)$ time by an algorithm of Karp and Orlin [48]. The translation from the tiling angular resolution problem to the parametric negative cycle detection problem and back can be performed within the same time bound.

8   Metric Embedding into Stars

There has been a large amount of interest recently within the theoretical computer science community in problems of embedding unstructured metric spaces (which may be specified, for instance, as a distance matrix) into simpler and more highly constrained metrics, with low distortion [52]. Such embeddings may be used, for instance, in approximation algorithms: one can design an approximation algorithm for the constrained class of metrics, and apply it to arbitrary metrics, incurring the distortion of the embedding as a penalty factor in the approximation ratio of the algorithm. The construction of graph spanners [23] may also be seen as an instance of this type of problem: one wishes to approximate a metric describing the shortest paths in an arbitrary graph, by a more highly constrained metric of shortest paths in a sparse graph. Most work in this area has concentrated on finding embeddings that guarantee low but non-optimal distortion; however there has also been some work on finding the best possible embedding [33,37]. In particular, we describe here our work on finding optimal embeddings into star metrics, which (as in the angle optimization of the previous section) involves translating the problem into one of parametric negative cycle detection in an auxiliary graph.

Following [33], we define a *star metric* to be a metric space in which there exists a distinguished point h (the *hub*) of the metric) such that, for every two points p and q, h lies on a shortest path from p to q. Expressed algebraically: $d(p,q) = d(p,h) + d(h,q)$. Thus, in contrast to arbitrary n-point finite metric spaces which have $n(n-1)/2$ degrees of freedom, a star metric has only $n-1$ degrees of freedom. We wish to find a map from an arbitrary finite metric space (represented as a distance matrix D) to a star metric space (represented as a vector H of distances from each point to the hub) that minimizes the distortion of the embedding; we do not require that the hub be the image of any point in the input space. By scaling H, if necessary, we can assume without loss of generality that the map does not decrease any distance; thus, what we seek is a vector H satisfying the constraint that, for all p and q, $D[p,q] \leq H[p] + H[q]$, and minimizing the *dilation*

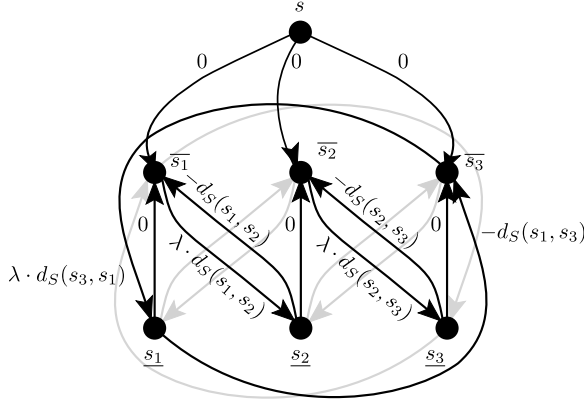$$\delta = \max_{p,q} \frac{H[p] + H[q]}{D[p,q]}.$$

Fig. 7. The parametric network into which an optimal star embedding problem is translated. From [33].

This is a linear program: it may be rephrased as the problem of finding H and δ subject to the linear nondecreasing-distance constraints and the linear constraints that, for each p and q, $D[p,q]\delta - H[p] - H[q] \geq 0$, and, among all tuples of H and δ that satisfy the constraints, finding the tuple minimizing δ. However, it may be solved more efficiently, in strongly polynomial time, by transforming it into a parametric negative cycle detection problem.

Specifically, as our paper shows, we may find the optimal star embedding using an auxiliary graph defined from the input, having two vertices $\overline{p}$ and $\underline{p}$ for each metric space point p, as well as a special start vertex s. There is an edge with length zero from s to each vertex $\overline{p}$ and from each vertex $\underline{p}$ to the corresponding $\overline{p}$. In addition, for each pair of points p and q, there is an edge of length $-D[p,q]$ from $\underline{p}$ to $\overline{q}$ and another edge of length $\lambda D[p,q]$ from $\overline{p}$ to $\underline{q}$. As we show, the optimal dilation δ is the smallest value that can be assigned to λ such that the resulting graph has no negative cycles, and for this value of λ the distance $D[p]$ of p from the hub in the optimal embedding may be computed as half of the difference between the two distances from s to $\overline{p}$ and $\underline{p}$.

Thus, the star metric embedding problem may be solved optimally by a single parametric negative cycle detection calculation similar to that in the previous section. However, in this case the edge weight functions of the parametrized network no longer have the special form needed to apply the Karp–Orlin algorithm; instead, we developed a more general algorithm that solves any parametric negative cycle detection problem in strongly polynomial time. The basic idea of the algorithm is to use a matrix squaring method of Savage [66] to calculate the parametrized function representing the weights of paths between each pair of vertices; after i iterations of squaring a matrix whose entries represent these functions, they will correctly describe the lengths of the shortest path between each pair of vertices among paths constrained to have at most 2^i hops. After each matrix-squaring step, the entries of the matrix will not be linear functions but rather more general piecewise linear concave functions; we perform a binary search among the set of breakpoints of these functions to narrow down the range of values of λ within which the optimal value must lie, allowing us to simplify the functions to be

linear once again. Each binary search step involves applying the Bellman-Ford algorithm to determine whether a specific value of λ causes the parametrized graph to have a negative cycle. The negative cycle detection algorithm resulting from this parametric search procedure has total running time $O(n^3 \log^2 n)$, which is therefore also the time for using this approach to solve the optimal star metric embedding problem.

9 Conclusions

As we have seen, a graph-theoretic point of view can be useful in many algorithmic problems that do not, initially, seem to have much to do with graphs. The graph problems occurring in the solutions of the geometry problems discussed here are mostly well-known and classical: they include the maximum independent set and maximum clique problems, maximum cardinality matching, shortest paths, and minimum cost flow. However, sometimes these problems occur in a somewhat different form than they have been studied elsewhere in the literature: the star metric embedding problem, for instance, required parametric negative cycle detection at a level of generality that had previously not been considered.

Special classes of graphs, and specialized algorithms that take advantage of these special classes, seem to be ubiquitous in this area. In the examples here, we have seen natural applications of trees, maximal outerplanar graphs, squaregraphs and kinggraphs, bipartite graphs, unit disk graphs, apex graphs, and 3-regular bridgeless graphs. The simple coloring algorithm for maximal outerplanar graphs described in the section on the art gallery theorem generalizes to chordal graphs (maximal outerplanar graphs are chordal) and to perfectly orderable graphs [14].

There are undoubtedly many more applications of graph-theoretic concepts in computational geometry waiting to be discovered, and it is important that the lines of communication remain open between the graph algorithm and computational geometry communities, so that computational geometers will know where to find the algorithms they need and so that graph algorithms researchers can focus their efforts on the problems and graph classes with the greatest benefit in geometric applications.

References

1. Agarwal, P.K., Efrat, A., Sharir, M.: Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. SIAM J. Comput. 29(3), 912–953 (2000)
2. Ageev, A.A.: A triangle-free circle graph with chromatic number 5. Discrete Mathematics 152, 295–298 (1996)
3. Aggarwal, A., Imai, H., Katoh, N., Suri, S.: Finding k points with minimum diameter and related problems. J. Algorithms 12(1), 38–56 (1991)
4. Ajtai, M., Chvátal, V., Newborn, M., Szemerédi, E.: Crossing-free subgraphs. In: Theory and Practice of Combinatorics. North-Holland Mathematics Studies, vol. 60, pp. 9–12 (1982)
5. Arkin, E.M., Held, M., Mitchell, J.S.B., Skiena, S.S.: Hamiltonian triangulations for fast rendering. The Visual Computer 12(9), 429–444 (1996)

6. Bandelt, H.-J., Chepoi, V., Eppstein, D.: Combinatorics and geometry of finite and infinite squaregraphs. Electronic preprint arxiv:0905.4537 (2009)
7. Biedl, T.C., Bose, P., Demaine, E.D., Lubiw, A.: Efficient algorithms for Petersen's matching theorem. J. Algorithms 38, 110–134 (2001)
8. Breu, H., Kirkpatrick, D.G.: Unit disk graph recognition is NP-hard. Computational Geometry Theory and Applications 9(1–2), 3–24 (1998)
9. Carlson, J., Eppstein, D.: Trees with convex faces and optimal angles. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 77–88. Springer, Heidelberg (2007)
10. Chazelle, B.: Triangulating a simple polygon in linear time. Discrete & Computational Geometry 6(1), 485–524 (1991)
11. Cheng, Y., Iyengar, S.S., Kashyap, R.L.: A new method of image compression using irreducible covers of maximal rectangles. IEEE Trans. Software Engineering 14(5), 651–658 (1988)
12. Chepoi, V., Dragan, F., Vaxès, Y.: Center and diameter problem in planar quadrangulations and triangulations. In: Proc. 13th Annu. ACM–SIAM Symp. on Discrete Algorithms (SODA 2002), pp. 346–355. ACM Press, New York (2002)
13. Chvátal, V.: A combinatorial theorem in plane geometry. Journal of Combinatorial Theory, Series B 18, 39–41 (1975)
14. Chvátal, V.: Perfectly orderable graphs. In: Berge, C., Chvátal, V. (eds.) Topics in Perfect Graphs. Annals of Discrete Mathematics, vol. 21, pp. 63–68. North-Holland, Amsterdam (1984)
15. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. Discrete Mathematics 86, 165–177 (1990)
16. de Bruijn, N.G.: Algebraic theory of Penrose's non-periodic tilings of the plane. Indagationes Mathematicae 43, 38–66 (1981)
17. Deering, M.: Geometry compression. In: Proc. 22nd Conf. Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 13–20 (1995)
18. Dey, T.K.: Improved bounds for planar k-sets and related problems. Discrete & Computational Geometry 19(3), 373–382 (1998)
19. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall, Englewood Cliffs (1998)
20. Díaz-Gutiérrez, P.: Using graph algorithms for geometry processing on surfaces. PhD thesis, Univ. of California, Irvine (2009)
21. Engel, K.: Optimal matrix-segmentation by rectangles. Discrete Applied Mathematics 157(9), 2015–2030 (2009)
22. Eppstein, D.: Geometric lower bounds for parametric matroid optimization. Discrete & Computational Geometry 20, 463–476 (1998)
23. Eppstein, D.: Spanning trees and spanners. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of Computational Geometry, ch. 9, pp. 425–461. Elsevier, Amsterdam (2000)
24. Eppstein, D.: Algorithms for drawing media. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 173–183. Springer, Heidelberg (2005)
25. Eppstein, D.: The traveling salesman problem for cubic graphs. J. Graph Algorithms and Applications 11(1), 61–81 (2007)
26. Eppstein, D.: Testing bipartiteness of geometric intersection graphs. ACM Trans. Algorithms 5(2), 15 (2009)
27. Eppstein, D., Erickson, J.: Iterated nearest neighbors and finding minimal polytopes. Discrete & Computational Geometry 11(3), 321–350 (1994)
28. Eppstein, D., Falmagne, J.-C., Ovchinnikov, S.: Media Theory. Springer, Heidelberg (2007)
29. Eppstein, D., Gopi, M.: Single-strip triangulation of manifolds with arbitrary topology. Eurographics Forum 23(3), 371–379 (2004); Proc. 25th Conf. Eur. Assoc. for Computer Graphics (EuroGraphics 2004)

30. Eppstein, D., Mumford, E.: Orientation-constrained rectangular layouts. In: Dehne, F., et al. (eds.) WADS 2009. LNCS, vol. 5664, pp. 266–277. Springer, Heidelberg (2009)

31. Eppstein, D., Mumford, E., Speckmann, B., Verbeek, K.A.B.: Area-universal rectangular layouts. In: Proc. 25th ACM Symp. Computational Geometry, pp. 267–276 (2009)

32. Eppstein, D., Wortman, K.: Optimal angular resolution for face-symmetric drawings. Electronic preprint arxiv:0907.5474 (2009)

33. Eppstein, D., Wortman, K.: Optimal embedding into star metrics. In: Dehne, F., et al. (eds.) WADS 2009. LNCS, vol. 5664, pp. 290–301. Springer, Heidelberg (2009)

34. Evans, F., Skiena, S.S., Varshney, A.: Optimizing triangle strips for fast rendering. In: Proc. 7th IEEE Conf. Visualization, pp. 319–326 (1996)

35. Ferrari, L., Sankar, P.V., Sklansky, J.: Minimal rectangular partitions of digitized blobs. Computer Vision, Graphics, and Image Processing 28(1), 58–71 (1984)

36. Fisk, S.: A short proof of Chvátal's watchman theorem. Journal of Combinatorial Theory, Series B 24(3), 374 (1978)

37. Fomin, F., Lokshtanov, D., Saurabh, S.: An exact algorithm for minimum distortion embedding. In: Paul, C., Habib, M. (eds.) WG 2009. LNCS, vol. 5911. Springer, Heidelberg (2009)

38. Garg, A., Tamassia, R.: A new minimum cost flow algorithm with applications to graph drawing. In: North, S.C. (ed.) GD 1996. LNCS, vol. 1190, pp. 201–216. Springer, Heidelberg (1997)

39. Hannenhalli, S., Hubbell, E., Lipshutz, R., Pevzner, P.A.: Combinatorial algorithms for design of DNA arrays. In: Chip Technology. Advances in Biochemical Engineering/Biotechnology, vol. 77, pp. 1–19. Springer, Heidelberg (2002)

40. Hershberger, J., Suri, S.: Finding tailored partitions. J. Algorithms 12(3), 431–463 (1991)

41. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. 2(4), 225–231 (1973)

42. Imai, H., Asano, T.: Efficient algorithms for geometric graph search problems. SIAM J. Comput. 15, 478–494 (1986)

43. Imai, H., Iwano, K.: Efficient sequential and parallel algorithms for planar minimum cost flow. In: Asano, T., Imai, H., Ibaraki, T., Nishizeki, T. (eds.) SIGAL 1990. LNCS, vol. 450, pp. 21–30. Springer, Heidelberg (1990)

44. Iwama, K., Nakashima, T.: An improved exact algorithm for cubic graph TSP. In: Lin, G. (ed.) COCOON 2007. LNCS, vol. 4598, pp. 108–117. Springer, Heidelberg (2007)

45. Junger, M., Mutzel, P.: Graph Drawing Software. Springer, Heidelberg (2004)

46. Kahn, J., Klawe, M., Kleitman, D.: Traditional galleries require fewer watchmen. SIAM Journal on Algebraic and Discrete Methods 4(2), 194–206 (1983)

47. Kalinowski, T.: A dual of the rectangle-segmentation problem for binary matrices. Electronic J. Combinatorics 16(1), R89 (2009)

48. Karp, R.M., Orlin, J.B.: Parametric Shortest Path Algorithms with an Application to Cyclic Staffing. Technical Report OR 103-80, MIT Operations Research Center (1980)

49. Kőnig, D.: Gráfok és mátrixok. Matematikai és Fizikai Lapok 38, 116–119 (1931)

50. Leighton, T.: Complexity Issues in VLSI. Foundations of Computing Series. MIT Press, Cambridge (1983)

51. Li, G., Zhang, H.: A rectangular partition algorithm for planar self-assembly. In: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, pp. 3213–3218 (2005)

52. Linial, N.: Finite metric spaces–combinatorics, geometry and algorithms. In: Proc. International Congress of Mathematicians, Beijing, vol. 3, pp. 573–586 (2002)

53. Lipski Jr., W.: Finding a Manhattan path and related problems. Networks 13, 399–409 (1983)

54. Lipski Jr., W.: An $O(n \log n)$ Manhattan path algorithm. Information Processing Letters 19, 99–102 (1984)

55. Lipski Jr., W., Lodi, E., Luccio, F., Mugnai, C., Pagli, L.: On two-dimensional data organization II. Fundamenta Informaticae 2, 245–260 (1979)

56. Malitz, S., Papakostas, A.: On the angular resolution of planar graphs. SIAM J. Discrete Mathematics 7, 172–183 (1994)
57. Miller, G.: Flow in planar graphs with multiple sources and sinks. SIAM J. Comput. 24(5), 1002–1017 (1995)
58. Mumford, E.: Drawing Graphs for Cartographic Applications. PhD thesis, Technische Universiteit Eindhoven (2008)
59. Nishizeki, T., Rahman, M.S.: Planar Graph Drawing. World Scientific, Singapore (2004)
60. Ohtsuki, T.: Minimum dissection of rectilinear regions. In: Proc. IEEE Int. Symp. Circuits and Systems, pp. 1210–1213 (1982)
61. O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, Oxford (1987)
62. Patel, K.: Computer-aided decomposition of geometric contours into standardized areas. Computer-Aided Design 9(3), 199–203 (1977)
63. Petersen, J.P.C.: Die theorie der regularen graphs. Acta Mathematica 15, 193–220 (1891)
64. Raisz, E.: The rectangular statistical cartogram. Geographical Review 24(2), 292–296 (1934)
65. Sack, J.-R., Urrutia, J.: Polygon decomposition. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 491–518. Elsevier, Amsterdam (1999)
66. Savage, C.: Parallel Algorithms for Graph Theoretic Problems. PhD thesis, University of Illinois, Urbana-Champaign (1977)
67. Shamos, M.I., Hoey, D.: Closest-point problems. In: Proc. 16th IEEE Symp. Foundations of Computer Science, pp. 151–162 (1975)
68. Tait, P.G.: Listing's Topologie. Philosophical Magazine (5th ser.) 17, 30–46 (1884)
69. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM J. Comput. 16(3), 421–444 (1987)
70. Tutte, W.T.: On Hamiltonian circuits. Journal of the London Mathematical Society (2nd ser.) 21(2), 98–101 (1946); Reprinted in Scientific Papers, vol. II, pp. 85–98
71. Vaidya, P.M.: Geometry helps in matching. SIAM J. Comput. 18(6), 1201–1225 (1989)
72. van Kreveld, M., Speckmann, B.: On rectangular cartograms. Computational Geometry Theory and Applications 37(3), 175–187 (2007)
73. Xiang, X., Held, M., Mitchell, J.S.B.: Fast and effective stripification of polygonal surface models. In: Proc. Symp. Interactive 3D Graphics, pp. 71–78 (1999)