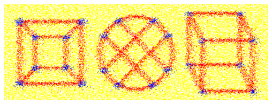

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 16, no. 2, pp. 599–628 (2012)
DOI: 10.7155/jgaa.00275

Augmenting the Connectivity of Planar and Geometric Graphs

Ignaz Rutter[1] *Alexander Wolff*[2]

[1]Faculty of Informatics, Karlsruhe Institute of Technology, Germany
[2]Lehrstuhl für Informatik I, Universität Würzburg, Germany
http://www1.informatik.uni-wuerzburg.de/en/staff/wolff_alexander

Abstract

In this paper we study connectivity augmentation problems. Given a connected graph G with some desirable property, we want to make G 2-vertex connected (or 2-edge connected) by adding edges such that the resulting graph keeps the property. The aim is to add as few edges as possible. The property that we consider is planarity, both in an abstract graph-theoretic and in a geometric setting, where vertices correspond to points in the plane and edges to straight-line segments.

We show that it is NP-hard to find a minimum-cardinality augmentation that makes a planar graph 2-edge connected. For making a planar graph 2-vertex connected this was known. We further show that both problems are hard in the geometric setting, even when restricted to trees. The problems remain hard for higher degrees of connectivity. On the other hand we give polynomial-time algorithms for the special case of convex geometric graphs.

We also study the following related problem. Given a planar (plane geometric) graph G, two vertices s and t of G, and an integer c, how many edges have to be added to G such that G is still planar (plane geometric) and contains c edge- (or vertex-) disjoint s–t paths? For the planar case we give a linear-time algorithm for $c = 2$. For the plane geometric case we give optimal worst-case bounds for $c = 2$; for $c = 3$ we characterize the cases that have a solution.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
July 2010	August 2012	August 2012	August 2012	August 2012

Published: September 2012

Article type: Regular Paper

Communicated by: M. T. Goodrich

Supported by grant WO 758/4-3 of the German Science Foundation (DFG).
E-mail address: rutter@kit.edu (Ignaz Rutter)

1 Introduction

Augmenting a given graph to increase its connectivity is important, for example, for making communication networks resistant against node and link failures. The planar version of the problem, where the augmentation has to preserve planarity, also has applications in graph drawing [12]. Many graph-drawing algorithms guarantee nice properties (such as convex faces) for graphs with high connectivity. To apply such an algorithm to a less highly connected graph, one adds edges until one reaches the required level of connectivity, uses the algorithm to produce the drawing, and finally removes edges that were added before. With each removal of an edge, however, one might loose some of the nice properties (such as the convexity of a face). Hence, it is natural to look for an augmentation that uses as few edges as possible. Recall that a graph is *c-vertex connected* (or simply *c-connected*) if the removal of any subset of $c-1$ vertices does not disconnect the graph. Analogously, a graph is *c-edge connected* if the removal of any subset of $c-1$ edges does not disconnect the graph. It is common to use the term *biconnected* for 2-vertex connected and the term *bridge-connected* for 2-edge connected.

In this paper, we consider the following two problems.

Planar 2-Vertex Connectivity Augmentation (PVCA):
Given a connected planar graph $G = (V, E)$, find a smallest set E' of vertex pairs such that the graph $G' = (V, E \cup E')$ is planar and biconnected.

Planar 2-Edge Connectivity Augmentation (PECA)
is defined as PVCA, but with *biconnected* replaced by *bridge-connected*.

The corresponding problems without the planarity constraints have a long history, both for directed and undirected graphs. Eswaran and Tarjan [6] showed that the unweighted cases can be solved in polynomial time, whereas the weighted versions are hard. Frederickson and Ja'Ja' [8] gave $O(n^2)$-time factor-2 approximations and showed that augmenting a directed acyclic graph to be strongly connected, and augmenting a tree to be bridge- or biconnected, is NP-complete—even if weights are restricted to the set $\{1, 2\}$. Hsu [9] gave an $O(m + n)$-time sequential algorithm for (unit-weight) 2-vertex connectivity augmentation that can be parallelized well.

Kant and Bodlaender [12] showed that PVCA is NP-complete and gave 2-approximations for both PVCA and PECA that run in $O(n \log n)$ time. Their 1.5-approximation for PVCA turned out to be wrong [7]. Fialko and Mutzel [7] gave a 5/3-approximation for PVCA. Kant [11] showed that PVCA and PECA can be solved in linear time for outerplanar graphs.

Provan and Burk [17] considered related problems. Given a planar graph $G = (V, E_G)$ and a planar biconnected (bridge-connected) graph $H = (V, E_H)$ with $E_G \subseteq E_H$, find a smallest set $E' \subseteq E_H$ such that $G' = (V, E_G \cup E')$ is planar and biconnected (bridge-connected). They show that both problems are NP-hard if G is not necessarily connected and give $O(n^4)$-time algorithms for the connected cases.

We also consider a geometric version of the above problems. Recall that a *geometric* graph is a graph where each vertex v corresponds to a point $\mu(v)$ in the plane and where each edge uv corresponds to the straight-line segment $\overline{\mu(u)\mu(v)}$ connecting u and v. We are exclusively interested in *plane* geometric graphs, that is, geometric graphs whose edges neither cross each other nor contain vertices other than their endpoints. Therefore, in this paper by geometric graph we always mean a plane geometric graph. Given a geometric graph G we again want to find a (small) set of vertex pairs such that adding the corresponding edges to G leaves G plane and augments its connectivity.

In this context, Rappaport [18] showed that it is NP-complete to decide whether a set of line segments can be connected to a simple polygon, that is, geometric PVCA and PECA are NP-complete. Abellanas et al. [1] gave worst-case bounds for geometric PVCA and PECA. For geometric PVCA, they showed that $n - 2$ edges are sometimes needed and are always sufficient. For geometric PECA, they proved that $2n/3$ edges are sometimes needed and $6n/7$ edges are always sufficient for graphs with n vertices. In the special case of plane geometric trees (with n vertices) they show that $n/2$ edges are sometimes needed and that $2n/3$ edges are always sufficient for PECA. Tóth [19] lowered the upper bounds to $\lfloor n/2 \rfloor$ for n-vertex trees and $2n/3 + O(1)$ for arbitrary n-vertex plane geometric graphs. Al-Jubeh et al. [2] characterized plane geometric graph that can be augmented to be 3-vertex or 3-edge

problem	planar	outerplanar	geometric	convex
PVCA	NPC [12]	$O(n)$ [11]	NPC [Thm. 2]	$O(n)$ [Obs. 2]
PECA	NPC [Thm. 1]	$O(n)$ [11]	NPC [Thm. 2]	$O(n)$ [Thm. 4]
w-PVCA	see above	open	see above	$O(n)$ [Obs. 2]
w-PECA		open		$O(n^2)$ [Thm. 5]

Table 1: Complexity of various versions of PVCA and PECA. NPC stands for NP-complete; the prefix "w" indicates the weighted versions of the problems.

connected. They showed that if a plane geometric graph with n vertices can be augmented to a 3-edge-connectivity, then at most $2n-2$ new edges are always sufficient and sometimes necessary. Their augmentation algorithm runs in $O(n \log^2 n)$ time. They further prove that, if the input graph is already 2-edge-connected, then $n-2$ new edges are always sufficient and sometimes necessary for the augmentation to 3-edge-connectivity. In this case, their algorithm runs in $O(n\alpha(n))$ time, where $\alpha(n)$ is the inverse of the Ackermann function.

Our results. First we show that PECA is NP-complete, too. This answers an open question posed by Kant [10].

Second, we sharpen the result of Rappaport [18] by showing that geometric PVCA and PECA are NP-complete even if restricted to trees. Not unexpectedly, the problems remain hard for higher degrees of connectivity: finding a minimum-cardinality augmentation that makes a plane geometric $(c-1)$-vertex connected graph c-vertex connected is also NP-hard for $c = 3, \ldots, 5$. The gadgets in our construction are such that they establish hardness for both vertex and edge connectivity. Recall that any planar graph has a vertex of degree at most 5 and hence is at most 5-connected.

Third, we give algorithms that solve geometric PVCA and PECA in polynomial time for *convex* geometric graphs, that is, graphs whose vertex sets correspond to point sets in convex position.

Table 1 gives an overview about what is currently known about the complexity of the problems PVCA and PECA and their geometric variants.

Fourth, we consider a related problem, the geometric s–t path augmentation problem. Given a plane geometric graph G, two vertices s and t of G, and an integer $c > 0$, is it possible to augment G such that it contains c edge-disjoint (c vertex-disjoint) s–t paths? We restrict ourselves to $c \in \{2, 3\}$. For $c = 2$ we show that edge-disjoint s–t path augmentation can always be done and needs at most $n/2$ edges, where n is the number of vertices in the graph G. We give an algorithm that computes such an augmentation in linear time. The tree that yields the above-mentioned lower-bound of Abellanas et al. [1] also shows that our bound is tight. For $c = 3$ we show that edge-disjoint s–t path augmentation is always possible, and we give an $O(n^2)$-time algorithm that decides whether a given graph has a vertex-disjoint s–t path augmentation.

In this paper we use the term *leaf* for a degree-1 vertex in any graph, not only in a tree.

2 Complexity

In this section, we show that PECA is NP-complete. This settles an open problem posed by Kant [12]. Kant proved that the minimum biconnectivity augmentation problem is NP-complete and gave 2-approximations for both problems [12]. We also strengthen the result of Rappaport [18] and show that geometric PECA and geometric PVCA are NP-complete even in the case of trees. We also show hardness for the corresponding problems with higher degrees of connectivity.

2.1 Complexity of PECA

We start by settling the complexity of PECA. For our proof, recall that an *embedding* of a planar graph is given by a circular ordering of the incident edges around each vertex.

Theorem 1 PECA *is NP-complete.*

Proof: PECA is in $\mathcal{NP}$ since, given a planar graph G and an integer $k > 0$, we can guess a set $E' \subseteq V \times V$ of at most k non-edges of G and then test efficiently whether $G + E'$ is planar.

We prove the NP-hardness of PECA by reducing from the problem Planar3SAT, which is known to be NP-hard [15]. An instance of Planar3SAT is a 3SAT formula φ whose variable–clause graph is planar. Such a graph can be laid out (in polynomial time) such that variables correspond to pairwise disjoint axis-parallel rectangles intersecting a horizontal line and clauses correspond to non-crossing three-legged "combs" above or below that line [13], see Figure 1.

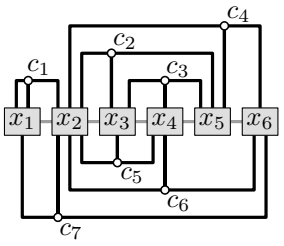


Figure 1: Layout of the variable–clause graph corresponding to a planar 3-SAT formula with variables $x_1, \ldots, x_6$ and clauses $c_1, \ldots, c_7$.

Note that if a graph G has k leaves, at least $k/2$ edges need to be added to bridge-connect the graph. In case $k/2$ edges suffice, each of these edges connects two leaves and no two edges are incident to the same leaf. In other words, the edges form a perfect matching of the leaves. We now construct a planar graph G_φ that can be augmented with a perfect leaf matching if and only if φ is satisfiable. The graph G_φ consists of so-called *gadgets*, that is, subgraphs that represent the variables, literals, and clauses of φ, see Figures 2. The rough structure of G_φ follows the layout of the variable-clause graph depicted in Figure 1. For each gadget, we will argue that there are only a few ways to embed and augment it with a perfect leaf matching. Note that our construction connects variable gadgets corresponding to neighboring variables in the layout of the variable–clause graph of φ. Hence G_φ is always connected. Additionally, we identify the left boundary of the leftmost variable gadget with the right boundary of the rightmost variable gadget.

In the figure, leaves are highlighted by small black disks. All bends and junctions of line segments represent vertices of degree greater than 1. The (black and dark gray) solid line segments between adjacent vertices represent the edges of G_φ; the thick dotted line segments represent non-edges of G_φ that are candidates for an augmentation of G_φ. The set of black solid edges forms a subgraph of G_φ that we call the *frame*. The dark gray solid edges form what we call *I-shapes* and *Y-shapes*, which connect singles leaves and pairs of leaves to the frame, respectively. In Figure 2, we marked examples of I- and Y-shapes.

Consider the graph G'_φ that we obtain from the frame by contracting all vertices of degree 2. We claim that G'_φ is 3-vertex connected. This is true since (a) the subgraph of G'_φ induced by the variable gadgets is 3-connected and (b) each subgraph induced by a clause gadget and the three corresponding literal gadgets is also 3-connected and is attached to the former (variable gadget) subgraph in six vertices.

Recall a classical result of Whitney's [20], which says that a 3-connected planar graph has a unique planar embedding. Hence, by the arguments above, G'_φ has a unique embedding. The same holds for the frame of G_φ as it is a subdivision of G'_φ. In other words, the embedding of G_φ is fixed up to the embedding of the I- and Y-shapes.

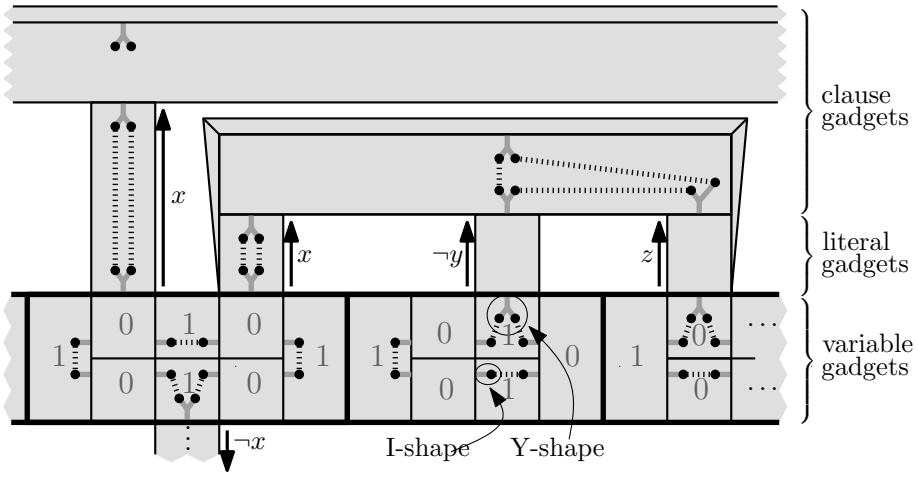


Figure 2: Part of the graph G_φ for a 3SAT formula φ that contains the clause $(x \wedge \neg y \wedge z)$. The augmentation (dotted edges) corresponds to the assignment $x = y = \mathit{false}$ and $z = \mathit{true}$.

We say that an augmentation of a gadget or of G_φ is *tight* if the new edges form a leaf matching and the resulting graph G' is bridge-connected. It is easy to see that if G_φ has a tight augmentation, then G_φ has an embedding such that the following two properties hold.

(P1) Each face contains an even number of leaves.

(P2) Each face that contains a Y-shape contains at least four leaves.

Note that in a tight augmentation, the two leaves of a Y-shape cannot be matched to each other since the edge of the Y-shape that is not incident to a leaf would be a bridge.

Our variable gadget consists of two rows of square faces where the horizontal edge between the two leftmost faces and the horizontal edge between the two rightmost faces is missing. Effectively, the faces of a variable gadget form a cycle. Starting from the leftmost (rectangular) face, we call the faces *odd* and *even*. To every interior vertical edge an I-shape is attached. Due to (P1), the I-shapes can be matched in exactly two ways; either in the odd or in the even faces. If the matching is in the even faces, then the corresponding variable is true, and vice versa.

A literal gadget consists of a square face that lies immediately above or below the variable gadget. A positive literal (such as the ones labeled with x in Figure 2) is attached to an even face, a negated literal (such as the one labeled with $\neg y$ in Figure 2) is attached to an odd face. A literal gadget contains two Y-shapes, one attached to each of its two horizontal edges. Due to (P2) these Y-shapes are embedded either both inside or both outside the literal gadget. Again due to (P2) the Y-shapes must be embedded inside the literal gadget if no I-shapes are embedded into the adjacent face of the variable gadget. In this case, the literal has the value *false*. If two I-shapes are embedded into the adjacent face of the variable gadget, the Y-shapes of the literal gadget can (but don't have to) be embedded to the outside (see the literal $\neg y$).

Finally, each clause gadget consists of a single rectangular face that contains a Y-shape. If G_φ has a tight augmentation, then, due to (P2), at least two other leaves are embedded into every clause gadget face. This means that for each clause gadget, the Y-shapes of at least one adjacent literal gadget are embedded to the outside. In other words, at least one of the literals that make the clause is *true*. Hence, φ has a satisfying truth assignment.

Conversely, it is easy to see that if φ has a satisfying truth assignment, then all gadgets have a tight augmentation and hence, so does G_φ.

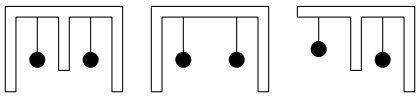


Figure 3: Various variants of loopholes.

We use a constant number of vertices and edges for each literal and clause gadget, thus our reduction—including the computation of the embedding of the variable–clause graph—is polynomial. □

Note that the graph constructed in the proof is 2-edge connected if and only if it is biconnected. Hence, our proof also shows that PVCA is NP-complete.

2.2 Geometric PVCA and Geometric PECA

Next we show that geometric PVCA and geometric PECA are NP-complete as well. With a simple modification, it follows that the problems are even NP-complete if the input is restricted to plane geometric trees. With another modification, we show hardness for the corresponding problems for higher degrees of connectivity.

Note that we cannot recycle our proof of Theorem 1 to show hardness for the geometric variants of the problems: there, we exploited that certain parts of the graph (the I- and Y-shapes) could be embedded in different (but adjacent) faces. Here, we are given embedded graphs; we cannot even move vertices or edges. To show hardness, we exploit this rigidity. Our proof is again by reduction from Planar3SAT. Although the graph that we are about to construct looks very different from the one we constructed in the proof of Theorem 1, similar functional units (as the I- and Y-shapes) will play a role.

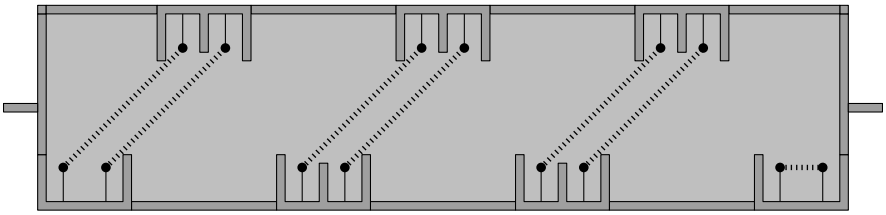
Theorem 2 *Let G be a connected plane geometric graph with k leaves. It is NP-complete to decide whether it is possible to augment G with $k/2$ edges such that G becomes bridge- or biconnected.*

Proof: For membership in $\mathcal{NP}$ we argue as in the proof of Theorem 1. To show the NP-hardness of the two problems, we again reduce from Planar3SAT and construct a connected plane geometric graph G consisting of gadgets that represent the variables, literals, and clauses of the given planar 3SAT formula. Recall once more that we need to add at least $k/2$ edges in order to make G bridge- or biconnected since every leaf must lie on a cycle afterwards and must hence be incident to one of the added edges.

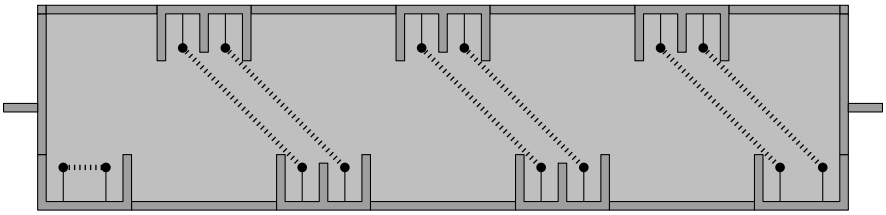
The basic building block of our gadgets is what we call a *loophole*. The default loophole, depicted in Figure 3a, consists of an E-shaped cycle and two attached I-shapes that are placed such that their leaves cannot see each other. (Recall that an I-shape is a leaf with its incident edge.) In contrast, in a *self-connecting* loophole (see Figure 3b), the two leaves do see each other. A *skewed* loophole is a loophole that misses one of the boundaries (Figure 3c). In the terminology of the proof of Theorem 1, a default loophole corresponds to a Y-shape, and a self-connecting loophole corresponds to a pair of I-shapes in the same face. Skewed loopholes are similar to default loopholes; their shape differs to allow for certain connections without crossings.

Again, all our gadgets are surrounded by *walls*, that is, by biconnected subgraphs that ensure that the whole construction without the leaves is biconnected. In the figures, walls are indicated by gray rectilinear polygons.

We are now ready to describe our variable gadget, see Figure 4. It consists of two parallel rows of evenly spaced loopholes with the upper loopholes pointing downwards and the lower ones pointing upwards. The lower row contains one loophole more than the upper one and its two outermost loopholes are self-connecting. The rows are aligned so that every loophole



(a) *true*



(b) *false*

Figure 4: Variable gadget without any adjacent literal gadgets; the two minimum augmentations correspond to the values of the variable.

(except the first and last of the lower row) lies horizontally between two opposing loopholes—which we call its partners—on the other row. The distances between the loopholes and the two rows are chosen such that the I-shapes of each loophole can only be connected to the leaves of its two partners on the other row without producing a crossing. As in the proof of Theorem 1, we connect neighboring variable gadgets to ensure that the resulting graph is connected.

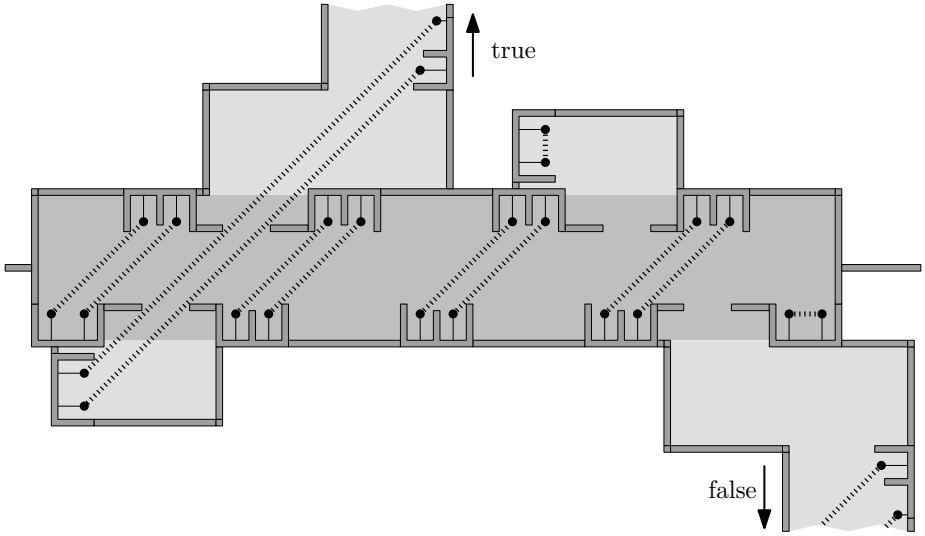
For any minimum augmentation, the two I-shapes of a loophole on the upper row must be connected to the two I-shapes of its left or right partner on the other row and the edges in the augmentation have slope 1 or -1. By construction this choice has to be the same for each loophole on the upper row, otherwise crossings would occur. On the lower row, depending on the choice either the first or last loophole does not receive new edges and its I-shapes must be connected. Hence, a variable has exactly two different minimum augmentations. The two possible states are shown in Figure 4. We say that the variable is in state *true* if the edges connecting loophole partners have slope 1 and *false* if they have slope -1.

Next we show how the state of a variable is transmitted to the gadgets that represent the clauses in which the variable occurs. This is the job of the literal gadgets. Roughly speaking, for each literal gadget we remove two wall pieces of the variable gadget, and attach a self-connecting loophole on one side and a skewed loophole on the other side, see Figure 5.

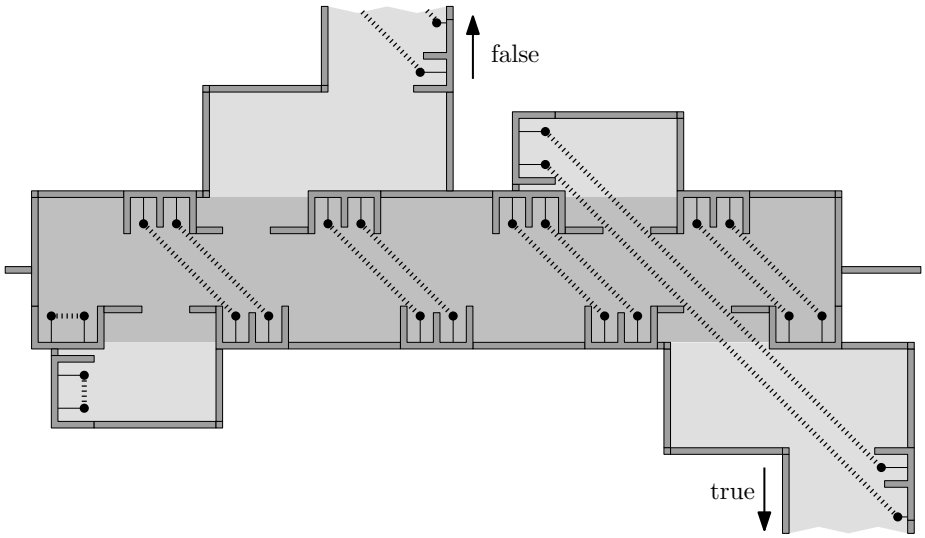
If the literal is positive (as in the case of the upper left literal gadget in Figure 5), a leaf in one of the two loopholes can be connected to a leaf in the other loophole if and only if the new edges in the variable gadget all have slope 1. If the literal is negated (as in the case of the lower right literal gadget in Figure 5), leaves of the two loopholes can only be interconnected if the new edges in the variable gadget all have slope -1.

In this way, the leaves of the skewed loophole can be matched to the leaves of the corresponding self-connecting loophole if and only if the value of the variable satisfies the corresponding literal. Otherwise, leaves of the skewed loophole have to be connected to a vertex inside the clause gadget, which we present next.

The clause gadget consists of a square that contains a loophole and two L-shaped wall parts that occupy the corners opposite of the loophole, see Figure 6. On three sides, the square is connected via literal gadgets to the gadgets of the three variables that form the clause in the given planar 3SAT formula. Each literal gadget contains two I-shapes that are positioned such

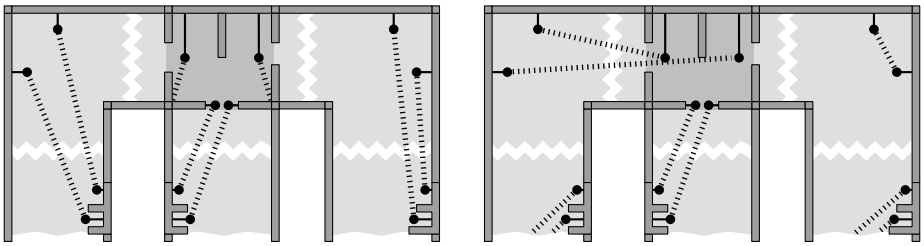


(a) state *true*



(b) state *false*

Figure 5: Variable gadget (middle gray) with two adjacent literal gadgets (light gray). The upper left literal gadget transmits the logical value of the variable, whereas the lower right literal gadget transmits the negation of that value.



(a) state *false* (b) state *true*

Figure 6: Clause gadget (middle gray) with the three adjacent literal gadgets (light gray). If all literals are *false*, the leaves in the clause gadget (or some other leaves) must be matched to non-leaves, or a leaf receives more than one new edge (a). If at least one literal is *true*, there is an augmentation that matches all leaves to other leaves. The right figure (b) depicts the situation where the literals corresponding to the left and the right gadget are true and the literal corresponding to the middle gadget is false.

that they see (a) each other, (b) the two I-shapes inside the square, and (c) the two I-shapes of the skewed loophole where the literal gadget is attached to a variable. It is not hard to see that if any of the skewed loopholes is matched to leaves in the variable gadget, then the two I-shapes in the literal gadget are free to connect to the two I-shapes in the square. Only if all three skewed loopholes are matched to I-shapes in their literal gadget, then the two I-shapes in the center square require an additional edge.

As in the proof of Theorem 1 we use a constant number of vertices and edges for each literal and clause gadget, thus our reduction—including the computation of the embedding of the variable–clause graph—is polynomial. □

By a simple trick we can slightly strengthen the result of Theorem 2.

Corollary 1 *It is NP-complete to decide whether a plane geometric tree with k leaves can be augmented to be bridge- or biconnected with $k/2$ edges.*

Proof: The proof is by reduction from the previous case. Let G be a connected plane geometric graph with k_G leaves. We now show how to remove cycles from G. Since the construction leaves G connected, the resulting graph is a tree.

To reduce the number of cycles we replace an arbitrary edge that lies on a cycle by the construction shown in Figure 7. Note that we can make the spiral in the center of the construction so small that is does not prevent any connections in the remainder of the graph. We iterate this construction until there are no cycles left. The resulting graph is a tree T. Let k_T be the number of leaves of T. It is clear that for each of the new leaves (in the spiral centers) there is only one way to connect to another leaf, namely to the one that restores the cycle we removed before. Hence, T can be augmented with $k_T/2$ edges if and only if G can be augmented with $k_G/2$ edges.

The reduction can be performed in polynomial time since we introduce at most one spiral per edge of G, each consisting of a constant number of edges. The length of the shortest new edge of T is roughly proportional to the smallest distance among the vertices of G. □

We now generalize the proof of Theorem 2 to show that for any $2 \leq c \leq 5$, it is NP-hard to augment a plane geometric graph to be c-connected by adding a given number of edges. Note that any planar graph has a vertex of degree at most 5, so planar graphs are at most 5-connected. To show that our construction has the desired properties, let us make some simple observations.

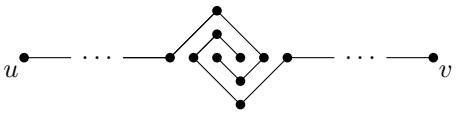


Figure 7: Construction that removes a cycle, locally leaves only one possibility to augment, and does not interfere with the remainder of the graph with respect to augmentation.

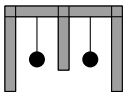


Figure 8: Subdivision of a loophole into rectangular blocks.

Figure 9: Adding edges to leaves in the second step of the construction of G_φ (here with $c = 5$).

Observation 1 *Let G be a graph with vertices u and v. Then the following properties hold:*

(i) If $G - u$ is c-connected and u has degree at least c, then G is c-connected as well.

(ii) If $G-\{u, v\}$ is c-connected and vertices u and v have degree at least $c-1$ but no common neighbor, then $G + uv$ is c-connected.

Proof: For showing property (i), suppose that S is a separator of G with $|S| < k$. Since S is no separator in $G - v$, S splits off only v from G. Hence, S contains all neighbors of v, and thus $|S| \geq k$. Contradiction.

Similarly, for property (ii), suppose that S is a separator of $G + uv$ with $|S| < k$. Since S is not a separator of G, S splits off u or v from $G + uv$. This, however, is not possible since both u and v have degree at least c in $G + uv$ and since their neighborhoods are disjoint. □

To generalize the proof of Theorem 2 to higher degrees of connectivity, we make the graph G_φ $(c - 1)$-connected in two steps.

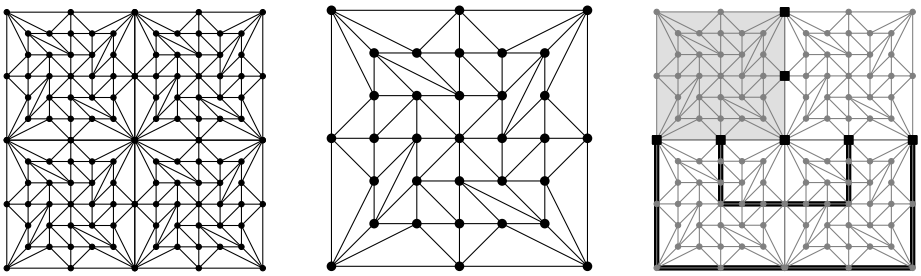
First, we temporarily remove all I-shapes from G_φ, subdivide the walls of the loopholes as in Figure 8 into gray rectangles, and replace each gray rectangle in Figures 5 and 6 by a copy of the graph depicted in Figure 10a. We stick two building blocks together by identifying the five vertices on the edge of one block to the five corresponding vertices on the edge of the other block. Call the resulting graph G^1_φ.

Second, we treat the former I-shapes of G_φ. We connect each leaf of G_φ by $(c - 2)$ additional edges to the boundary of G^1_φ such that no two leaves have a common neighbor, see Figure 9. We call the resulting graph G^2_φ. We now show that G^2_φ does the job.

Theorem 3 *It is NP-hard to decide the following question: given integers $2 \leq c \leq 5$ and $k \geq 1$ and a $(c-1)$-connected plane geometric graph G, can G be augmented to being c-connected by adding at most k edges?*

Proof: We again reduce from Planar3SAT, along the lines of the proof of Theorem 2. We first show that G^2_φ is $(c - 1)$-connected.

In order to see this, we claim that G^1_φ is 5-connected. The walls of G^1_φ are made from copies of our basic building blocks. Such a block is 5-connected for two reasons; (a) it consists of four copies of the smaller 5-connected graph, a *sub-block*, depicted in Figure 10b, whose 5-connectivity we have verified by a computer program and (b) two neighboring sub-blocks lie in the same 5-connected component. To see (b), consider the five *portals* that we define on the



(a) (b) (c)

Figure 10: A building block (a), a 5-connected sub-block with 33 vertices (b), proof that a building block is 5-vertex connected (c).

boundary of each sub-block, see the black squares in Figure 10c. Each vertex in a sub-block has five vertex-disjoint paths to its portals, which are connected to the corresponding portals of the neighboring sub-block via (possibly trivial) pairwise vertex-disjoint paths. Observations (a) and (b) plus symmetry show our claim.

Given that G^1_φ is 5-connected and $c \leq 5$, property (i) of Observation 1 yields that G^2_φ is $(c-1)$-connected. Note that the leaves of G_φ and the degree-$(c-1)$ vertices of G^2_φ are in one-to-one correspondence. Let K be their number. Clearly, in order to make G^2_φ c-connected, we need at least $K/2$ new edges. We claim that the graph G^2_φ that we have constructed above can be made c-connected by adding $K/2$ edges if and only if G_φ can be made biconnected by adding $K/2$ edges.

If G^2_φ can be made c-connected by adding $K/2$ edges, then these edges form a matching of the vertices of degree $c-1$. This matching can also be added (in a plane fashion) to G_φ.

Now we turn to the other direction. If G_φ can be made biconnected by adding $K/2$ edges, we add the corresponding edges to G^2_φ. We have shown above that G^1_φ is 5- and thus c-connected. Now property (ii) of Observation 1 yields that each of the remaining vertices lies in the same c-connected component as G^1_φ. This finishes the proof of our claim.

Clearly, our reduction is polynomial. □

Using the same graph G^2_φ in the reduction, we can prove the statement for edge connectivity, too.

Corollary 2 *Given integers $2 \leq c \leq 5$ and $k \geq 1$ and a $(c-1)$-edge connected plane geometric graph G, it is NP-hard to decide whether G can be augmented to being c-edge connected by adding at most k edges.*

3 Convex Geometric Graphs

In this section we show that geometric PVCA and geometric PECA can be solved in polynomial time for connected *convex* geometric graphs, that is, for graphs whose vertices are in convex position. We focus on augmenting a given connected convex geometric graph to bridge- and biconnectivity. Note that every convex geometric graph is outerplanar and hence contains a vertex of degree at most 2, which prevents higher connectivity.

We first consider the very simple problem of biconnecting a convex geometric graph, see Section 3.1. Then we give an algorithm that computes an edge set of minimum cardinality that bridge-connects a convex geometric graph, see Section 3.2. Finally, in Section 3.3 we consider a weighted version of bridge-connectivity augmentation. We give an algorithm that computes a minimum-weight augmentation in a connected n-vertex convex geometric graph in $O(n^2)$ time.

We assume that for a geometric graph the edges incident to a vertex are ordered clockwise. If this information is not provided, we can easily compute it in $O(n \log n)$ time.

3.1 Biconnecting Convex Geometric Graphs

Consider an arbitrary connected convex geometric graph G. Suppose that there are two consecutive vertices u and v on the convex hull that are not connected by an edge. Since G is connected, adding the edge uv creates a new face F. It is not hard to see that every vertex of $F - \{u, v\}$ disconnects G. Hence, in a biconnected convex graph all edges of the convex hull must be present.

On the other hand if all edges of the convex hull are present, then the graph is biconnected. Hence, it suffices to add all edges of the convex hull that are not already in G to make G biconnected. This is also the minimum number of edges that must be added. As the convex hull of the point set can be computed in linear time if G is connected [16], convex geometric graphs can be augmented to biconnectivity in linear time. We summarize this brief discussion.

Observation 2 *Let S be a set of n points in the plane, and let $G = (S, E)$ be a connected convex geometric graph. There is an efficient algorithm that computes a minimum-weight set E' of edges such that $G + E'$ is biconnected. If the embedding of G is given, the algorithm runs in linear time and uses linear space.*

3.2 Bridge-Connecting Convex Geometric Graphs

In this section we consider the problem of bridge-connecting a convex geometric graph $G = (V, E)$. We start by considering two basic graphs that are especially easy to bridge-connect, the *cycle* and the *near-cycle* shown in Figure 11. While the cycle is already bridge-connected, the near-cycle is not. It can, however, be bridge-connected by adding the single missing edge to form a cycle.

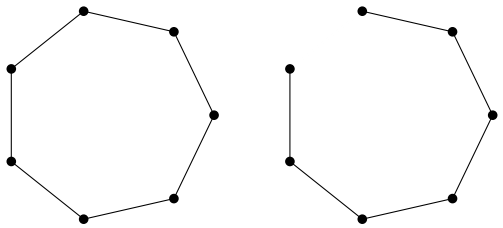


Figure 11: A cycle (left) and a near-cycle (right).

The basic idea is to decompose an arbitrary convex geometric graph into cycles and near-cycles and to use this decomposition to compute in a greedy fashion an edge set of minimum cardinality that bridge-connects the graph.

We differentiate between two types of edges. If an edge connects two consecutive vertices of the convex hull, we call it an *outer* edge, otherwise an *inner* edge. Note that if G is a connected convex geometric graph that does *not* contain an inner edge, then G is a cycle or a near-cycle.

Otherwise, an inner edge $e = uv$ can be used to split G into two subgraphs that can be augmented almost independently. The line defined by e splits the vertex set of G into two convex point sets P_1 and P_2. We then define, for $i = 1, 2$, the graph G_i as the subgraph of G induced by $P_i \cup \{u, v\}$. The interplay between augmentations of these two graphs is very limited since adding any edge between two vertices that are distinct from u and v and that do not belong to the same subgraph would introduce a crossing with e and is hence forbidden. On the other hand, the two augmentations are not completely independent as it suffices for e to be in *one* cycle. Hence, we store, for each edge e of G, a flag indicating whether e is already part of a cycle in the current partial augmentation. Initially all these flags are set to false.

Splitting G recursively along all inner edges defines a tree $\mathcal{T}$ whose nodes correspond to subgraphs of G. Two nodes of $\mathcal{T}$ are adjacent if and only if the corresponding subgraphs share an edge of G. The leaves of this tree correspond to components that are cycles or near-cycles. Starting from $E' = \emptyset$, we compute a minimum augmentation E' of G by iteratively augmenting a component C that corresponds to a leaf of $\mathcal{T}$. We do this as follows.

Let $e = uv$ be the edge that is shared by C and its parent in $\mathcal{T}$, and let $C' = C \setminus \{u, v\}$. We distinguish three different types of components. If C is a cycle, we mark all edges of C and remove C' from G. If C is a near-cycle that contains at least one edge except e that is not yet marked, we add to E' the unique edge that completes the cycle, mark e as lying on a cycle and remove C' from G. Finally, if C is a near-cycle and each edge except possibly e has been marked, we do not add any edge to E' and remove all vertices of C' from G. See Figure 12 for an example. Note that component 5 does not require an edge although it is not a cycle.

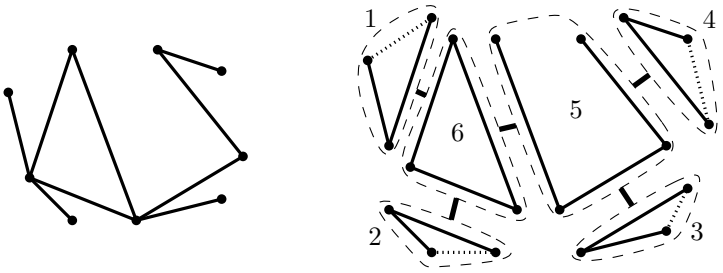


Figure 12: A convex geometric graph (left) and its decomposition along interior edges (right). The dashed edges form a bridge-connectivity augmentation of minimal size, the numbers indicate the processing order of the components.

Once we have processed the last component of G, the set E' is an augmentation of G since we only remove edges from G that are marked as lying on cycles in $G + E'$. The minimality of E' follows from the fact that in each component we need to add at most one edge and we only add an edge to a component if it is strictly required.

The algorithm can be implemented to run in linear time. The initial computation of $\mathcal{T}$ takes linear time, maintaining a list of leaves of the decomposition tree can be done in constant time per step and processing a component C takes time linear in the size of C.

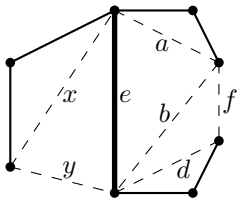
We summarize our result.

Theorem 4 *Let S be a set of n points in the plane, and let $G = (S, E)$ be a connected convex geometric graph. There is an efficient algorithm that computes a minimum-cardinality set E' of edges such that $G + E'$ is bridge-connected. If the convex hull of S and the corresponding planar embedding of G is given, the algorithm runs in linear time and uses linear space.*

3.3 Minimum-Weight Augmentation

We now generalize the algorithm from the previous section to the case where every potentially new edge e is associated with a positive cost $c(e)$. We seek a minimum-cost augmentation of a given plane graph G such that G becomes bridge-connected (while remaining plane). For a set of edges E' we define the cost of E' as $c(E') = \sum_{e \in E'} c(e)$. Given a connected convex geometric graph with n vertices, we can solve the problem in $O(n^2)$ time.

The basic idea is again to use a decomposition into (near-)cycles. The main difference from the previous problem is that in a near-cycle it is not always the best solution to add the unique edge that completes the cycle. Consider the graph given by the solid edges in Figure 13. The costs of the vertex pairs that are connected by dashed line segments are given in the table next to the drawing or will be specified later; all other non-adjacent vertex pairs



edge	a	b	d	f
cost	2	4	5	10

Figure 13: Example of a minimum-weight augmentation.

have a very high cost. We first focus on the component to the right of and including the bold split edge e. Adding the unique edge f that completes the cycle would incur a cost of 10, whereas adding the edge set $E_1 = \{b, d\}$ would incur a cost of only 9. Adding the edge set $E_2 = \{a, d\}$ would be even cheaper, namely 7. This solution, however, has the disadvantage that e does not lie on a cycle in the component to the right of e; hence e is forced to lie on a cycle in the component to the left of e. Which option yields the better solution globally depends on the costs of edges x and y. If $c(y) - c(x)$ is greater than $c(E_1) - c(E_2)$, the optimal global solution is $E_1 \cup \{x\}$, otherwise $E_2 \cup \{y\}$ is optimal. Hence, we cannot make the decision between E_1 and E_2 in advance. Instead, we store both costs for the component to the right of e, the cost $w^+(e)$ of a cheapest augmentation that puts e on a cycle and the cost $w^-(e)$ of a cheapest augmentation that does not necessarily put e on a cycle. Note that $w^+(s) \geq w^-(s)$ for any split edge s.

Initially, we set $w^+(e) = \infty$ and $w^-(e) = 0$ for each outer edge e of G. We then compute the decomposition tree $\mathcal{T}$ of G and process its components starting from the leaves as in the algorithm described in the previous section. Other than there, we need to use dynamic programming to find a global minimum-cost solution. Let C be a component and let $e = uv$ be the edge that is shared by C and its parent in $\mathcal{T}$. We assume that, for all edges e' of C that are distinct from e, we already have computed $w^+(e')$ and $w^-(e')$. For any set E' of vertex pairs of C, we denote the set of bridges of $C + E'$ by $\mathrm{br}_C(E')$, and we define the cost of E' with respect to e as

$$\mathrm{cost}_e(E') = \sum_{e' \in \mathrm{br}_C(E') \setminus \{e\}} (w^+(e') - w^-(e')) + \sum_{e' \in E'} c(e').$$

The first term of the cost function describes the increase of augmentation cost stemming from the fact that e is not on a cycle in $C + E'$ and hence must be part of a cycle in a previously processed component. The second term is the cost for the edges in E'. We set

$$w^-(e) = \min_{E'} \mathrm{cost}_e(E')$$

and

$$w^+(e) = \min_{E',\, e \notin \mathrm{br}_C(E')} \mathrm{cost}_e(E').$$

We now show how to compute these values efficiently. If C is a cycle, then $w^-(e) = w^+(e) = \mathrm{cost}(\emptyset) = 0$. If C is a near-cycle, we can reduce the computation of $w^-(e)$ and $w^+(e)$ to a shortest-path problem as follows.

We say that an augmentation E' of C is *(inclusion) minimal* if, for any proper subset $E'' \subset E'$, we have that $\mathrm{br}_C(E')$ is a proper subset of $\mathrm{br}_C(E'')$, that is, any smaller set covers fewer edges. The following lemma shows that any minimal plane augmentation of C has a certain path structure.

Lemma 1 *Let E' be a minimal plane augmentation of C, and let $u_1, \ldots, u_k$ be the vertices of C as they occur along C. Then $P = E' \cup \mathrm{br}_C(E')$ forms a path from u_1 to u_k. The subset of vertices that is visited by P occur along P in the same order as in C.*

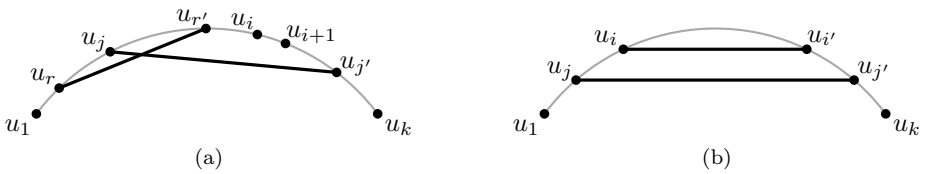


Figure 14: Illustration for the proof of Lemma 1.

Proof: We first show that all vertices of P have degree at most 2, except for u_1 and u_k, which have degree at most 1. Suppose that a vertex u_i of P is incident to two distinct vertices $u_j, u_{j'}$ with $i < j < j'$. Then edge u_iu_j is not a bridge since vertices $u_i, u_j, \ldots, u_{j'}$ form a cycle containing this edge. Therefore, u_iu_j lies in E'. As $j' > j$, we have that $\mathrm{br}_C(E' \setminus \{u_iu_j\}) = \mathrm{br}_C(E')$. Analogously, u_i can have at most one neighbor u_j with $j < i$ in P.

Next, we show that P connects u_1 and u_k. Note that u_1 is not a singleton, as it either is incident to an edge of E' or u_1u_2 lies in $\mathrm{br}_C(E')$. Let u_i be the vertex with the largest index that belongs to the connected component of u_1 in P. Note that $i > 1$ by the previous observation. Now suppose that $i < k$. The choice of u_i implies that u_i is not adjacent to any vertex u_j with $j > i$ in P. In particular, the edge u_iu_{i+1} does *not* lie in $\mathrm{br}_C(E')$, which, in turn, implies the existence of an edge $u_ju_{j'}$ with $j < i < j'$, see Figure 14a. Since $u_{j'}$ is not in the same connected component as u_1 (this would contradict the choice of u_i), we have that u_j, too, belongs to another connected component. Hence, the path from u_1 to u_i must contain an edge $u_ru_{r'}$ with $r < j$ and $j < r' \leq i < j'$. Such an edge would, however, cross the edge $u_ju_{j'}$, contradicting the planarity of E'. Therefore, we have that $i = k$ and that P connects u_1 and u_k as claimed.

It remains to show that P is connected. Suppose that P contains an edge $u_iu_{i'}$ that is not connected to u_1. Then, due to the planarity of E', the path from u_1 to u_k in P contains an edge $u_ju_{j'}$ with $j < i < i' < j'$. But then $u_iu_{i'} \notin \mathrm{br}_C(E')$ and $\mathrm{br}_C(E') = \mathrm{br}_C(E' \setminus \{u_iu_{i'}\})$, see Figure 14b. This contradicts $u_iu_{i'}$ lying in P.

The planarity of E' implies that P contains its vertices in the same order as along C. □

Lemma 1 shows that we can compute $w^+(e)$ by finding a shortest u_1–u_k path in the directed, weighted graph $\vec{C} = (V_C, \vec{E}_C; \ell)$ with vertex set $V_C = \{u_1, \ldots, u_k\}$, edge set $\vec{E}_C = \{u_iu_j \mid 1 \leq i < j \leq k, u_iu_j \neq e\}$, and weight

$$\ell(u_iu_j) = \begin{cases} w^+(u_iu_j) - w^-(u_iu_j), & j = i+1 \\ c(u_iu_j), & j > i+1 \end{cases}$$

for each edge u_iu_j in $\vec{E}_C$. Analogously, we can compute $w^-(e)$ by adding e to $\vec{C}$ with a weight of 0. Since $\vec{C}$ is a directed acyclic graph, a shortest path can be computed in time $O(|\vec{C}|) = O(|C|^2)$ [4]. This yields an overall running time of $O(n^2)$. We have proved the following theorem.

Theorem 5 *Let G be a connected convex geometric graph with n vertices. Then we can find, in $O(n^2)$ time, a minimum-weight set E' of vertex pairs such that $G + E'$ is bridge-connected.*

4 Path Augmentation

In this section we consider the following two problems.

Planar k-path augmentation (k-PathAug):
Given a planar graph G, two vertices s and t of G, and an integer $k > 1$, find a smallest set E' of vertex pairs sucht that $G + E'$ is planar and contains k edge-disjoint s–t paths.

Plane geometric k-path augmentation (geometric k-PathAug) is defined as above with "planar" replaced by "plane geometric". Note that for $k = 2$ the geometric case (geometric 2-PathAug) is a relaxed version of PECA. Both problems have a variant where the aim is to find vertex-disjoint paths. We refer to this as the *vertex variant* of (geometric) k-PathAug.

In the following we give a polynomial-time algorithm for planar 2-PathAug. We then turn to the geometric version of the problem. We show that in the worst case $n/2$ edges are needed for geometric 2-PathAug. For $k > 2$ geometric k-PathAug does not always have a solution. We give necessary and sufficient conditions for geometric 3-PathAug. We do not consider the non-geometric variant 3-PathAug, because every planar graph with at least four vertices can be triangulated, and hence, can be augmented to contain three vertex-disjoint paths between any two vertices [14].

4.1 Planar 2-Path Augmentation

Theorem 6 *2-PathAug and its vertex variant can be solved in linear time.*

Proof: We only consider the edge variant; the vertex variant can be solved analogously. Let $G = (V, E)$ be a planar graph, let s and t be two vertices of G, and let $C_1, \ldots, C_r$ be the 2-edge connected components of G. We first consider a special case of the problem. We assume that the 2-edge connected components of G form a path $C_1, \ldots, C_r$ and that $s \in C_1$ and $t \in C_r$.

For each component C_j with $2 \leq j \leq r-1$ consider the two vertices u_j and v_j of C_j that are incident to bridges. We say that C_j is a *pearl*, if $C_j + u_j v_j$ is planar. This is the case if and only if C_j has an embedding such that u_j and v_j lie on the outer face. If C_j is not a pearl, we say that C_j is a *ring*.

Let $i < k$ and let w_i and w_k be vertices of C_i and C_k, respectively, such that $G + w_i w_k$ is planar. Now assume there is a component C_j with $i < j < k$ that is a ring. Then the graph that results from contracting $C_1 \cup \cdots \cup C_{j-1}$ to u_j and $C_{j+1} \cup \cdots \cup C_r$ to v_j is $C_j + u_j v_j$. Contractions do not violate planarity, thus $C_j + u_j v_j$ is planar. This, however, violates the assumption that C_j is a ring. Hence no edge in a planar augmentation of G can "bypass" a ring. In other words, an optimal augmentation contains an edge between C_1 and the first ring, between the first and the second ring, etc., and between the last ring and C_r. If there are no rings, the optimal augmentation consists of an edge connecting C_1 and C_r, for example, between the corresponding cut vertices.

Now we consider the general case, that is, the 2-edge connected components form a tree $\mathcal{T}$. In $\mathcal{T}$, the components that contain s and t are connected by a path. This is the special case we have treated above. Obviously, any planar augmentation of the subgraph induced by the components on the path is also a planar augmentation of G. Since no ring on the path can be bypassed, there is no planar augmentation of G that uses fewer edges.

The tree of the 2-edge connected components can be computed in linear time. Finding the ring components on the path between s and t also takes linear time. Hence the whole algorithm runs in linear time. □

4.2 Geometric 2-Path Augmentation

Although geometric 2-PathAug appears to be a simplification of geometric PECA, it is not obvious how to take advantage of this. Therefore we consider the worst-case problem: how many edges are needed for geometric 2-PathAug in the worst case. For a zig-zag path with end vertices s and t whose vertices are in convex position $n/2$ edges are needed in order to establish two edge-disjoint s–t paths, see Figure 15. Abellanas et al. [1] came up with this example to show that, for trees, geometric PECA sometimes requires $n/2$ edges. They conjectured that $n/2$ edges always suffice to augment a tree to bridge-connectivity. Recently, Tóth [19] confirmed this. This shows that $n/2$ edges always suffice for geometric 2-PathAug in *trees*.

We show that *any* plane geometric graph has, for any two vertices s and t, an s–t 2-path augmentation with at most $n/2$ edges. We also give a simple algorithm that finds such an augmentation in linear time. We use the fact that every geometric graph $G = (S, E)$ has a *geometric triangulation*, that is, there is a graph $T = (S, E')$ with $E \subseteq E'$ such that all faces

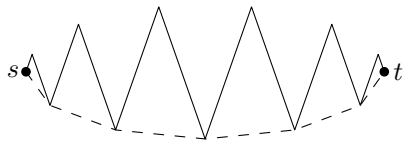


Figure 15: Zig-zag path of n vertices that needs $n/2$ edges (dashed) to augment [1].

of T except perhaps the outer face are triangles. This follows from the fact that every simple polygon has a triangulation [5].

Lemma 2 *Let S be a finite set of points in the plane, and let $s, t \in S$. Let $G = (S, E)$ and $G' = (S, E')$ be connected plane geometric graphs such that $E \subseteq E'$. If G' contains a path of length L between s and t, then there exists an s–t 2-path augmentation of G with at most L edges.*

Proof: We can assume that G' is a triangulation of S since this does not increase the length of a shortest s–t path in G'.

Let π be a path of length L between s and t in G'. We denote its vertices by $s = v_0, \ldots, v_L = t$. We use induction on L to show that we can augment G with L edges. We start with the case $L = 1$, that is, G' contains the edge $e = \{s, t\}$. If s and t lie in the same 2-edge connected component of G, G already contains two edge-disjoint s–t paths and we're done. Otherwise we consider two cases.

If e is not in G, then s and t lie in the same 2-edge connected component of $G + e$ since G is connected. If e is already in G then e is a bridge (otherwise s and t would be in the same 2-edge connected component). Removing e from G yields two connected subgraphs G_1 and G_2 of G. Since G' is a triangulation of S that contains all edges of G, there exists an edge $e' = vw$ in G' with $v \in G_1$ and $w \in G_2$ such that e' is different from e and $G + e'$ is plane. In $G + e'$ the vertices s and t lie in the same 2-edge connected components.

We now consider $L > 1$. Given a path π of length L we first apply the induction hypothesis to the path $\pi' = v_0, \ldots, v_{L-1}$. Then we use the same argument as above to show that it suffices to add at most one edge to G to make sure that v_{L-1} and v_L are in the same 2-edge connected component. The augmented graph is plane since it is a subgraph of G'. □

For the main result of this section, it remains to show that triangulations have small diameter. We need the following notation. Given a triangulation T and vertices s and t in T, we denote by $d(s, t)$ the length of a shortest s–t path in T. For a vertex v of T we denote by $N^i(v) = \{u \in T \mid d(v, u) \leq i\}$ the set of vertices of T at distance at most i from v and by $\partial N^i(v) = \{u \in T \mid d(v, u) = i\}$ the set of all vertices at distance exactly i from v. Note that $N^{i+1}(v) = N^i(v) \cup \partial N^{i+1}(v)$ for any vertex v in T and any integer $i \geq 0$.

Lemma 3 *Let S be a set of n points in the plane, and let $T = (S, E)$ be a triangulation of S. Then T contains a path of length at most $n/2$ between any pair of points in S.*

Proof: We first show that for any vertex v of T and for any integer $i \geq 0$ it holds that $|N^i(v)| \geq 2i + 1$ or $N^i(v) = S$. For $i = 0$ the statement clearly holds. For $i > 0$, we show that either $|\partial N^i(v)| \geq 2$ or $N^i(v) = S$. Clearly, $\partial N^i(v) = \emptyset$ implies $N^i(v) = S$ since T is connected. If $\partial N^i(v) = \{x\}$ and $N^i(v) \neq S$, then there exists a vertex y in $S \setminus N^i(v)$. In this case, however, the fact that every path from s to y must contain x implies that x is a cut vertex, which contradicts the fact that T is biconnected. This proves our lower bound on $|N^i(v)|$.

Now consider any pair of vertices s and t in S and set $k = \lfloor n/2 \rfloor$. By the previous inequality we have that $N^k(s) \geq 2\lfloor n/2 \rfloor + 1 \geq n$ and hence $N^k(s) = S$. Hence, t lies in $N^k(s)$ and, by the definition of $N^k(s)$, there exists a path of length at most $n/2$ from s to t. □

Together, Lemmas 2 and 3 yield the following theorem.

Theorem 7 *Let S be a set of n points in the plane, let $G = (S, E)$ be a plane geometric graph, and let s and t be two vertices of G. Then there is an s–t 2-path augmentation of G that uses at most $n/2$ edges.*

We now improve this bound for the case that the convex hull $\mathrm{CH}(S)$ of S does not contain too many points. The basic idea is to simultaneously grow neighborhoods around s and t; once $N^i(s)$ and $N^i(t)$ both contain vertices of $\mathrm{CH}(S)$ for some $i \geq 0$, there is a relatively short path connecting them.

Lemma 4 *Given a set S of n points in the plane, a geometric triangulation of S has diameter at most $2(n+3)/5 + h/2$, where $h = |\mathrm{CH}(S)|$.*

Proof: Let T be a triangulation of S and let v be a vertex of T. We claim the following. If $N^i(v) \cap \mathrm{CH}(S) = \emptyset$ then $|N^i(v)| \geq 3i + 1$.

We show this by induction on i. Clearly, the claim holds for $i = 0$. Now let $i \geq 1$. We apply the induction hypothesis to $N^{i-1}(v)$ and show that $|\partial N^i(v)| \geq 3$ if $N^i(v) \cap \mathrm{CH}(S) = \emptyset$. Assume, for the sake of contradiction, that $|\partial N^i(v)| \leq 2$. That means that all paths going from $N^{i-1}(v)$ to $S \setminus N^i(v)$ must pass through one of the two vertices in $\partial N^i(v)$. Hence, $\partial N^i(v)$ is a separator of cardinality 2. Let T' be the plane graph that results from T by triangulating the outer face of T (using non-straight-line edges). Since all edges in $T' - T$ connect points on the convex hull of S, which is disjoint from $N^i(v)$, it holds that $\partial N^i(v)$ is a separator of cardinality 2 of T'. This is a contradiction to the fact that every fully triangulated graph is 3-connected. Hence, our assumption is wrong, and the case $|\partial N^i(v)| \leq 2$ is ruled out. In other words, $|\partial N^i(v)| \geq 3$ for all $i \geq 1$ with $N^i(v) \cap \mathrm{CH}(S) = \emptyset$. This proves our claim.

Now let

$$k = \min\{i \mid N^i(s) \cap N^i(t) \neq \emptyset \text{ or both } N^i(s) \cap \mathrm{CH}(S) \neq \emptyset \text{ and } N^i(t) \cap \mathrm{CH}(S) \neq \emptyset\}.$$

be the first iteration where the iterated neighborhoods either meet or both have reached the convex hull of S.

Clearly there exists a path of length $2k + h/2$ between s and t. The neighborhoods give a path from s to the convex hull and from t to the convex hull and any two points on the convex hull are connect by a path of length at most $h/2$.

We now bound k in a similar fashion as before. We have $|N^{k-1}(s) \cup N^{k-1}(t)| \geq 5(k-1) + 2 = 5k - 3$ since the neighborhoods of s and t grow by at least two vertices as shown in the proof of Lemma 3 and one of them grows by at least three vertices by the claim above. On the other hand $|N^{k-1}(s) \cup N^{k-1}(t)| \leq n$. From this we get $k \leq (n+3)/5$.

Hence there exists a path from s to t with length at most $2k + h/2 \leq 2(n+3)/5 + h/2$. □

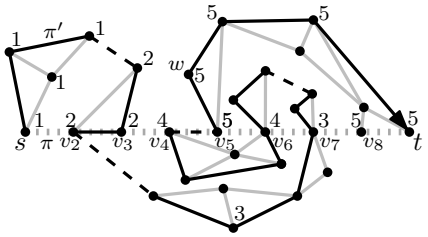
Note that the bound in Lemma 4 is strictly better than the bound in Lemma 3 if $h < (n-12)/5$.

We now turn to the corresponding algorithmic problem. In the remainder of this subsection we show how to compute a solution to geometric 2-PathAug of size at most $n/2$ in linear time. Given a graph G, our algorithm consists of the following three steps.

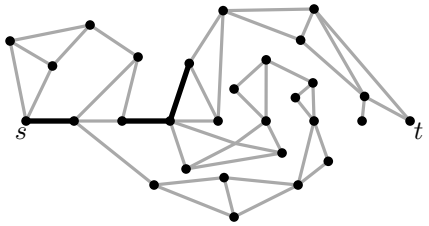
1. Find any triangulation T of G.
2. Compute a shortest path π from s to t in T.
3. Construct an s–t 2-augmentation from π (whose existence follows from Lemma 2).

Concerning step 1, note that the boundary of the outer face of a plane geometric graph is, in general, not a simple polygon. It is however *weakly simple* in the sense that segments that have a common point in the interior are actually the same segment. Algorithmically, weakly simple polygons can be handled just like simple polygons [1].

Therefore, we can apply Melkman's linear-time algorithm [16] for computing the convex hull of a polygonal chain to compute the convex hull of our geometric graph. We then add edges between neighboring points on the convex hull. Now all interior faces of our graph are weakly simple polygons and can hence be triangulated in linear time [3]. Let T be the resulting triangulation of G.



(a) A geometric graph G (solid gray, solid black and dashed black edges) with an s–t path π' (black; bridges are dashed). The s–t path π (dotted gray straight line) belongs to the triangulation T, which is not shown.



(b) The same graph G (gray edges) and the augmentation (black edges) computed by our algorithm.

Figure 16: Example for the linear-time 2-path-augmentation algorithm.

Concerning step 2, a shortest s–t path π in T can be found in linear time using breadth-first search.

Finally, concerning step 3, we want to show how to find an augmentation of G in linear time. We first compute a data structure that allows us to measure how we proceed along the path π by adding edges. Let π' be a simple s–t path in G. We remove all bridges of G that are used by π'. We call the resulting graph G'. We number the connected components of G' in the order in which they occur along π' and label the vertices of each component accordingly, starting with 1. Note that, by construction, all vertices on π' that have the same label lie in the same 2-edge connected component (with respect to the graph G). We denote the label of a vertex u by $\ell(u)$. See Figure 16a for an example.

As in the proof of Lemma 2, we go through the edges of π in order, starting from the edge leaving s. We consider the edges of π directed towards t. Initially, we set $j = 1$. Throughout the algorithm we maintain the following two invariants.

(I1) It holds that $j \geq \ell(v)$ for each vertex v of π whose incoming edge has already been processed.

(I2) All vertices of π' with label up to j lie in the same 2-edge connected component of G.

Both invariants clearly hold for $j = 1$. Together, the invariants yield the correctness of the algorithm since π ends in t and hence, according to invariant (I1), we have $j = \ell(t)$ after the last step and thus, by invariant (I2), s and t belong to the same 2-edge connected component.

We now describe the algorithm and show that it preserves the two invariants. We distinguish two main cases based on the values of j and of the label $\ell(v)$ of the endpoint of the current edge $e = uv$ of π.

If $\ell(v) \leq j$ (as for $e = v_5v_6$ in Fig. 16a), we simply advance to the next edge of π. Clearly, this preserves both invariants.

If $\ell(v) > j$, we add a suitable edge to G as follows. If e is not in G (as for $e = v_3v_4$ in Fig. 16a), we simply add e to G and set j to $\ell(v)$. Otherwise (as for $e = v_4v_5$ in Fig. 16a), e is a bridge of G lying on the path π' and we have $\ell(v) = \ell(u) + 1$. In the triangulation T, the edge $e = uv$ bounds at least one triangle. Let uwv be such a triangle. Hence, there are two possibilities for adding an edge to G; either uw or wv. If $\ell(w) > \ell(u)$ (as for $u = v_4$ and $v = v_5$ in Fig. 16a), we add the edge uw (edge v_4w in Fig. 16a) and set j to $\ell(w)$ (to 5 in Fig. 16a). Otherwise, we add the edge wv and set j to $\ell(v)$.

Clearly, the number of edges we add is bounded by the length of the path π. See Fig. 16b for the edges that are added in the case of the graph from Fig. 16a.

We now argue that the algorithm preserves our invariants. By our choice of j, it is clear that invariant (I1) holds. To prove (I2), let a and b be the two endpoints of the newly added edge. Let a' be a vertex of π' closest to a in G' (in terms of graph distance) and let π_a

be a shortest a–a' path. Let b' and π_b be defined analogously. Since π_a and π_b lie in G', $\ell(a) = \ell(a')$ and $\ell(b) = \ell(b')$. As $\ell(a) \neq \ell(b)$, π_a and π_b are disjoint; they "live" in different 2-edge-connected components of G. The newly added edge ab together with π_a and π_b and the subpath of π' that connects a' and b' form a simple cycle. This shows that, after adding the new edge ab, vertices a' and b' belong to the same 2-edge connected component of G. By invariant (I2) we have that a' lies in the same 2-edge connected component as s. Now we use transitivity and the fact that, after the addition of ab, variable j is set to $\ell(b) = \ell(b')$. This yields that indeed, after adding ab, all vertices of π' with label at most j lie in the same 2-edge connected component. In summary, we have proved the following theorem.

Theorem 8 *Let S be a set of n points in the plane, let $G = (S, E)$ be a plane geometric graph, and let s and t be vertices of G. Then there exists a set E' of at most $n/2$ vertex pairs such that $G + E'$ is a plane geometric graph that contains two edge-disjoint s–t paths. Such a set of vertex pairs can be computed in $O(n)$ time.*

4.3 Geometric 3-Path Augmentation

In this section we consider the problem of augmenting geometric graphs to contain more than two disjoint s–t paths while staying plane. The planar case obviously always has a solution, because every planar graph can be triangulated and a planar triangulation is always 3-connected. Hence we focus on the plane geometric cases in this section. In the following we give necessary and sufficient conditions for when plane geometric s–t 3-augmentation has a solution.

We first consider the vertex version of the problem, that is, given a geometric graph $G = (S, E)$ and two vertices s and t of G, add edges to G such that G contains three vertex-disjoint s–t paths.

4.3.1 The Vertex-Disjoint Case

Let $T = (S, E)$ be any plane geometric triangulation, and let s and t be any two vertices of T. An edge between two vertices of the convex hull that does not belong to the convex hull itself is called a *chord*. A chord $e = \{u, v\}$ is *s–t separating* if s and t lie in different connected components of $T \setminus \{u, v\}$.

Obviously there exist three vertex-disjoint s–t paths in T if and only if T does not contain an s–t separating chord. Hence we can rephrase our original question in the following form: Let G be any plane geometric graph. Can we triangulate G such that the resulting triangulation T_G contains no s–t separating chord? The following theorem states that this question can be answered in the affirmative.

Theorem 9 *Let S be a finite set of points in the plane, let $G = (S, E)$ be a plane geometric graph, and let s and t be any two vertices of G. If G contains no s–t separating chord, we can compute a triangulation T_G that contains three vertex-disjoint s–t paths.*

Proof: In the first step we add all edges of the convex hull to G and compute any triangulation of the interior. We can give an total ordering to the s–t separating chords of the triangulation by their facial distance from s. Let uv be the chord that is closest to s. Let uvw be the triangle that is on the same side as s with respect to uv and let uvw' be the other triangle bounded by uv. As u and v lie on the convex hull, we can *flip* the chord uv, that is, replace uv by the edge ww' without destroying planarity. If the new edge ww' was an s–t separating chord, then one of the edges uw and vw would have to be an s–t separating chord as well, contradicting the choice of uv, see Figure 17. Hence we have removed an s–t separating chord without introducing a new one. Inductively we obtain the desired triangulation T_G. □

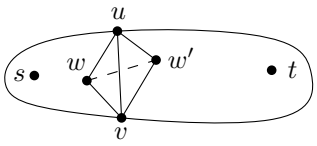


Figure 17: Removing an s–t separating chord uv by flipping.

4.3.2 The Edge-Disjoint Case

In this section we consider the problem of adding edges to a given plane geometric graph G such that for two fixed vertices s and t of G there exist three edge disjoint s–t paths. Since every plane geometric graph can be triangulated, we characterize the triangulations that contain three edge-disjoint s–t paths.

Theorem 10 *Let S be a finite set of points in the plane, let $T = (S, E)$ be a geometric triangulation of S, and let s and t be any two vertices of T. Then T contains three edge-disjoint s–t paths if and only if s and t have degree at least 3.*

Proof: Clearly the degree conditions are necessary for the existence of three edge-disjoint paths in T. We show that they are also sufficient. We use Menger's Theorem, which says that a graph is k-connected if and only if it contains k vertex-disjoint paths between any pair of vertices. Hence, since T is biconnected, there exist two vertex-disjoint s–t paths π_1 and π_2. We now show that we can find a third s–t path π_3 that is edge-disjoint from π_1 and π_2.

We start our construction of π_3 by constructing a path to a vertex s^* on $(\pi_1 \cup \pi_2) \setminus \{s\}$. Let e_1 and e_2 be the first edges of π_1 and π_2, respectively. Since s has degree at least 3 and T is triangulated, there exists a triangle incident to s whose boundary contains exactly one of the edges e_1 and e_2. Let s, s_1, and s_2 be the vertices of this triangle. We assume without loss of generality that $ss_1 = e_1$. We start π_3 with the edge ss_2, which neither belongs to π_1 nor to π_2. If s_2 lies on π_1 (see Figure 18a), we let $s^* = s_2$. Otherwise (see Figure 18b) we append the edge s_2s_1 to π_3 and let $s^* = s_1$. Note that s_2s_1 neither belongs to π_1 nor to π_2.

We now show that given the vertex s^* on π_1, we can continue the construction of π_3 by using π_1 as a "hand rail". Let u be a vertex on π_1 (initially $u = s^*$), and let v be the vertex next to u on π_1 in the direction of t. Then the edge uv bounds a triangle on at least one side. If $v = t$ then, due to $\deg(t) \geq 3$, there exists a triangle whose boundary contains ut and two other edges that do not belong to π_1 and π_2. Hence, we can use these to connect π_3 to t. If $v \neq t$, we consider the triangle $\{u, v, w\}$ whose boundary contains uv. If w is incident to v on π_1 in the direction of t, then we append the edge uw to π_3, see Figure 18a. Otherwise we append edges uw and wv to π_3, see Figure 18b. In neither of the two cases do we use edges that belong to π_1 or π_2.

Note that the path we construct in this way is not necessarily simple. This can be corrected by removing cycles. □

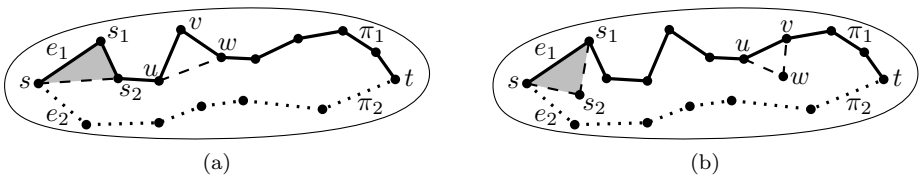


Figure 18: Hand rail construction along path π_1.

5 Conclusions

We have studied the complexity of several connectivity augmentation problems. We have showed that PECA, PVCA, and their geometric variants are all NP-hard. On the positive side, we have given efficient algorithms for 2-PathAug and its vertex variant on planar graphs. Further, we have studied worst-case bounds for geometric 2-PathAug, and we have fully characterized geometric graphs that can be augmented to contain three (vertex-)disjoint s–t paths.

We conclude with two open questions. Does geometric PECA admit a constant-factor approximation? Can geometric 2-PathAug be solved efficiently?

Acknowledgements

We thank Csaba Tóth for the idea behind Theorem 3 and Joachim Spoerhase for finding and helping us correct a number of mistakes in our manuscript.

References

[1] M. Abellanas, A. García, F. Hurtado, J. Tejel, and J. Urrutia. Augmenting the connectivity of geometric graphs. *Comput. Geom. Theory Appl.*, 40(3):220–230, 2008.

[2] M. Al-Jubeh, M. Ishaque, K. Rédei, D. L. Souvaine, C. D. Tóth, and P. Valtr. Augmenting the edge connectivity of planar straight line graphs to three. *Algorithmica*, 61(4):971–999, 2011.

[3] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(1):485–524, 1991.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition, 2001.

[5] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 3rd edition, 2008.

[6] K. P. Eswaran and R. E. Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976.

[7] S. Fialko and P. Mutzel. A new approximation algorithm for the planar augmentation problem. In *Proc. 9th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA'98)*, pages 260–269, 1998.

[8] G. N. Frederickson and J. Ja'Ja'. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981.

[9] T. Hsu. Simpler and faster biconnectivity augmentation. *J. Algorithms*, 45(1):55–71, 2002.

[10] G. Kant. *Algorithms for Drawing Planar Graphs*. PhD thesis, University of Utrecht, 1993.

[11] G. Kant. Augmenting outerplanar graphs. *J. Algorithms*, 21(1):1–25, 1996.

[12] G. Kant and H. L. Bodlaender. Planar graph augmentation problems. In F. Dehne, J.-R. Sack, and N. Santoro, editors, *Proc. 2nd Workshop Algorithms and Data Structures (WADS'91)*, volume 519 of *Lecture Notes Comput. Sci.*, pages 286–298. Springer-Verlag, 1991.

[13] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.

[14] S. O. Krumke and H. Noltemeier. *Graphentheoretische Konzepte und Algorithmen*. Vieweg+Teubner, 2nd edition, 2009.

[15] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[16] A. A. Melkman. On-line construction of the convex hull of a simple polyline. *Inform. Process. Lett.*, 25(1):11–12, 1987.

[17] J. S. Provan and R. C. Burk. Two-connected augmentation problems in planar graphs. *J. Algorithms*, 32:87–107, 1999.

[18] D. Rappaport. Computing simple circuits from a set of line segments is NP-complete. *SIAM J. Comput.*, 18(6):1128–1139, 1989.

[19] C. D. Tóth. Connectivity augmentation in planar straight line graphs. *Europ. J. Comb.*, 33(3):408–425, 2012.

[20] H. Whitney. Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, 43:150–168, 1932.