Augmenting the Connectivity of Planar and Geometric Graphs*

Ignaz Rutter† Alexander Wolff‡

Technical Report 2008-3
Fakultät für Informatik, Universität Karlsruhe

Abstract

In this paper we study some 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 also 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.

*A preliminary version of this work has been published as I. Rutter and A. Wolff, Augmenting the Connectivity of Planar and Geometric Graphs, *Electronic Notes Discrete Math.*, vol. 31, pp. 53–56

†Fakultät für Informatik, Universität Karlsruhe(TH), KIT, P.O. Box 6980, D-76128 Karlsruhe, Germany. Supported by grant WO 758/4-3 of the German Science Foundation (DFG). WWW: `http://i11www.iti.uni-karlsruhe.de/en/members/Ignaz_Rutter/index`.

‡Lehrstuhl für Informatik I, Institut für Informatik, Universität Würzburg.

1 Introduction

Augmenting a given graph to increase its connectivity is important, e.g., for securing communication networks against node and link failures. The planar version of the problem, where the augmentation has to preserve planarity, also has applications in graph drawing [KB91]. 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 the added edges again. However, with each removal of an edge 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* 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. The unweighted cases can be solved in polynomial time, while the weighted versions are hard [ET76]. Frederickson and Ja'Ja' [FJ81] 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 [Hsu02] gave an $O(m + n)$-time algorithm for (unit-weight) 2-vertex connectivity augmentation.

Kant and Bodlaender [KB91] 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 [FM98]. Fialko and Mutzel [FM98] gave a 5/3-approximation for PVCA. Kant [Kan96] showed that PVCA and PECA can be solved in linear time for outerplanar graphs.

Provan and Burk [PB99] 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 $\overline{\mu(u)\mu(v)}$. We are exclusively interested in geometric graphs that are *plane*, that is, whose edges intersect at most in 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.

Rappaport [Rap89] has shown 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. [AGH$^+$08] have given worst-case bounds for geometric PVCA and PECA. For geometric PVCA they show that $n-2$ edges are sometimes needed and are always sufficient. For geometric PECA they prove that $2n/3$ edges are sometimes needed and $6n/7$ edges are always sufficient. In the special case of plane geometric trees they show that $n/2$ edges are sometimes needed and that $2n/3$ edges are always sufficient for PECA. Tóth [Tót08] improved the upper bounds to $\lfloor n/2 \rfloor$ for trees and $2n/3 + O(1)$ for arbitrary plane geometric graphs.

Our results. First we show that PECA is NP-complete, too. This answers an open question posed by Kant [Kan93].

Second, we sharpen the result of Rappaport [Rap89] 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 $(c-1)$-connected graph c-connected is also NP-hard for $c = 3, \ldots, 5$. (Any planar graph has a vertex of degree at most 5 and hence is at most 5-connected.) The gadgets in our construction are such that they establish hardness for both vertex and edge connectivity.

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 PVCA and PECA and their geometric variants.

problem	planar	outerplanar	geometric	convex
PVCA	NPC [KB91]	$O(n)$ [Kan96]	NPC	$O(n)$
PECA	NPC	$O(n)$ [Kan96]	NPC	$O(n)$
weighted PVCA	NPC	open	NPC	$O(n^2)$
weighted PECA	NPC	open	NPC	$O(n)$

Table 1: Complexity of PVCA and PECA.

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. [AGH$^+$08] 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 [KB91]. Kant proved that the minimum biconnectivity augmentation problem is NP-complete and gave 2-approximations for both problems [KB91]. We also strengthen the result of Rappaport [Rap89] 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

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 (with positive probability) a set E' 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 [Lic82]. 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 the x-axis and clauses correspond to non-crossing three-legged "combs" above or below the x-axis [KR92], see Figure 1.

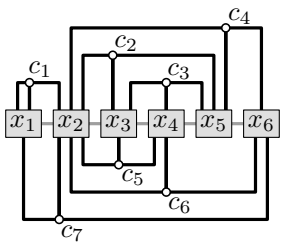
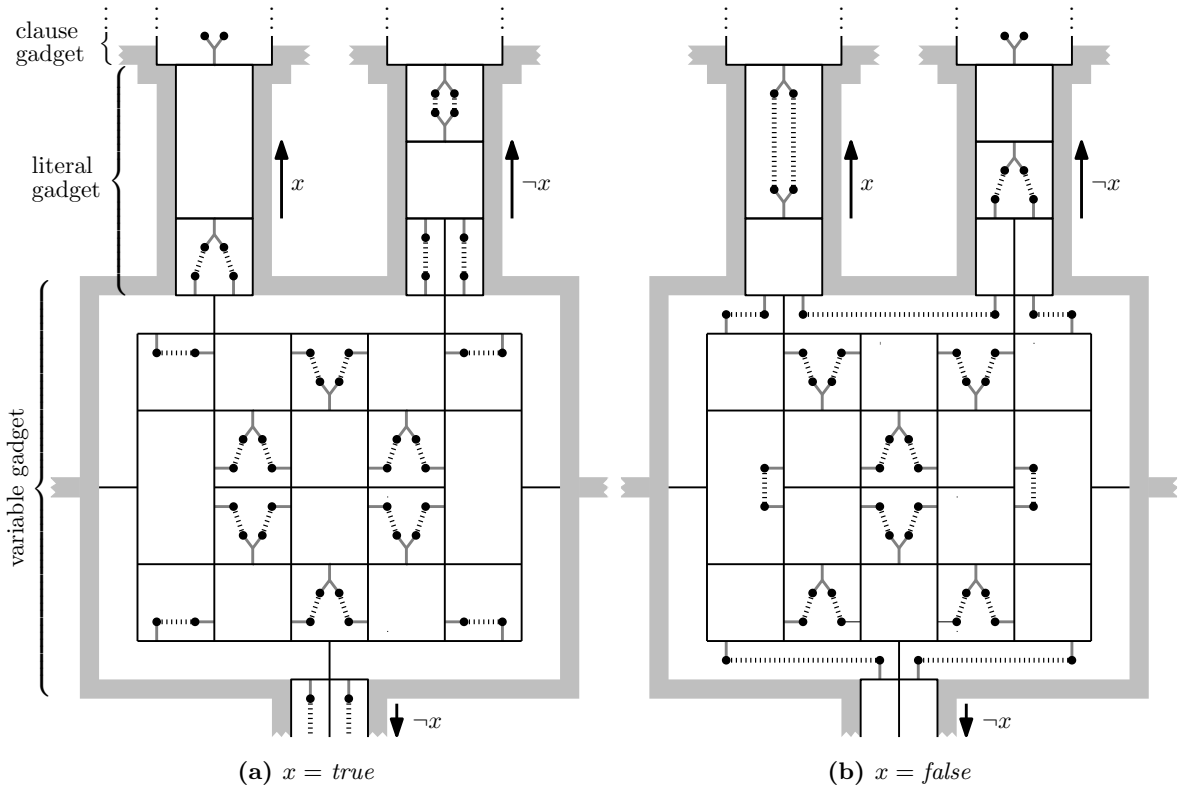


Fig. 1: Layout of the variable–clause graph corresponding to a planar 3-SAT formula with variables $x_1, \ldots, x_6$ and clauses $c_1, \ldots, c_7$.

Note that if a graph G has k leaves, at least $k/2$ edges are needed to bridge-connect the graph. We now construct a planar graph G_φ with k_φ leaves such that G_φ can be made bridge-connected by adding $k_\varphi/2$ edges if and only if φ is satisfiable. To achieve this, we construct so-called *gadgets*, that is, subgraphs of G_φ, that represent the variables, literals, and clauses of φ, see Figures 2 and 3.

In the two figures, leaves are highlighted by small black disks. All bends and junctions of line segments represent vertices of degree greater 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. We call a frame vertex incident to a shape the *base* of that shape.

The union of the gadgets is surrounded by a *wall*, that is, a 3-connected planar subgraph of G_φ that we depicted by a thick light-gray rectilinear polygon in Figures 2 and 3. The wall also connects gadgets of variables that are adjacent in the layout of the variable–clause graph



(a) $x = \mathit{true}$ **(b)** $x = \mathit{false}$

Fig. 2: Gadget for a variable x in its two states; the adjacent literal gadgets are also shown.

mentioned above. The task of the wall is to fix the planar embedding of G_φ, except for that of the I- and Y-shapes in the variable and literal gadgets. Those can be embedded in either of the two faces incident to their bases.

Consider a variable x that occurs in the given 3SAT formula φ. The part of the frame that belongs to the gadget of x is a grid that consists of 4×5 squares, except that the horizontal edges between the two center-most grid cells in the first and the last column are missing. This grid is connected via literal gadgets, which we describe below, to the gadgets of all clauses that contain x. The grid is connected to the literal gadgets by vertical edges, one for each literal. Further, a number of I- and Y-shapes are attached to the grid: (a) to each of the 4×4 vertical edges in the interior of the grid and somewhere near each of the four corner vertices of the grid, an I-shape is attached, and (b) to each of the three center-most horizontal edges between the first and the second and between the third and the forth row of grid cells, a Y-shape is attached. This finishes the description of our variable gadget. Note that it is symmetric to the vertical line through the center of the third column and to the horizontal line between the second row and the third row of grid cells.

We claim that there are exactly two ways how to augment the variable gadget such that 16 edges are used for the 32 leaves. In this claim, edges that connect variable-gadget to literal-gadget leaves count half. The two ways are depicted in Figures 2a and 2b. Note that the two leaves of a Y-shape cannot be matched with each other in such an augmentation; each of them must be matched to the leaf of an I-shape. This shows that the Y-shapes in the upper half

of the grid must alternatingly go up and down. Of course, the same holds for the Y-shapes in the lower half. The behaviors of the two halves are synchronized by the I-shapes incident to the non-separated grid cells in the first and the last column. The two former I-shapes are either matched to the Y-shapes in the second column (forcing them towards the horizontal symmetry axis of the gadget) or they are matched to each other (forcing the Y-shapes in the second column away from the horizontal axis of symmetry). Due to symmetry, this forces the I-shapes near the four corners of the grid either all into the grid or all out of the grid. This finishes the proof of our above claim. Now we identify the augmentations of the gadget of variable x that embed the corner I-shapes inside the grid with the value *true* of x, and we identify the only other possible augmentation with the value *false* of x.

The gadget of x is connected to the gadget of each clause in φ that contains x. Each connection is realized by a literal gadget, see Figure 2. A literal gadget consists of two or three cells depending on whether the corresponding literal is positive or negated. In both cases, a Y-shape is connected to each of the two horizontal edges that are closest to the clause gadget. In the gadget for positive literals, two I-shapes are connected to the horizontal edge that is closest to the variable gadget. In the gadget for negated literals, the cell closest to the variable gadget is split vertically into two halves, and an I-shape is connected to each of the four horizontal edges of the resulting *twin cell*. This finishes the description of how the literal gadget is constructed.

We now claim that a Y-shape of the literal gadget can only be embedded in the adjacent clause gadget if the value of the literal is *true*. In the case of a positive literal this is easy to see (see Figure 2a): the other Y-shape must be embedded in the cell adjacent to the variable gadget, which forces the two I-shapes into the same cell. This is only possible if the corner I-shapes of the variable gadgets are embedded inside the grid, that is, if the variable is *true*. In the case of a negated literal, we argue as follows. If a Y-shape is embedded in the adjacent clause gadget, the other Y-shape must be embedded into the middle cell, which in turn forces the I-shapes connected to the boundary of the twin cell to lie outside the twin cell. (Here we need the split!) This is only possible if the corner I-shapes of the variable gadget lie outside the grid, that is, if the variable is *false*.

Finally, our clause gadget is shown in Figure 3. It consists of a square box containing a Y-shape. The desired bound ($k/2$ augmentation edges for k leaves) can only be met if at least one literal gadgets embeds its last Y-shape into the box. (The case where two literals do this—see Figure 3c—illustrates why we need Y-shapes, not I-shapes in the clause box.) If and only if no Y-shape can be embedded into the box, the clause is *false*, and we need an additional edge as shown in Figure 3a.

We use a constant number of vertices and edges for each 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 this 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 of the corresponding problems for higher degrees of connectivity.

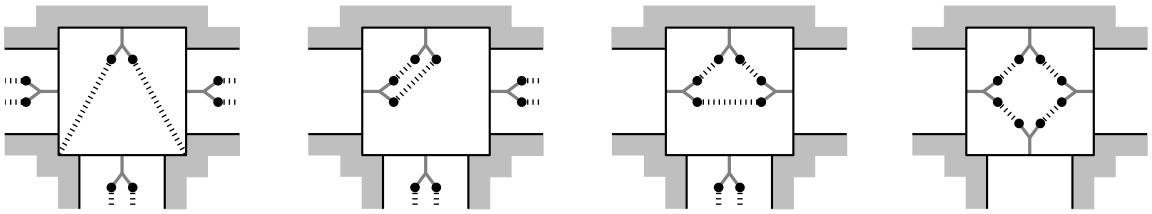


Fig. 3: Clause gadget with literal gadgets attached on the left, right, and bottom side. If all three literal gadgets transmit the value *false*, the two leaves in the clause gadget must be matched to non-leaves (a), otherwise they can be matched to leaves of the literal gadgets (b)–(d).

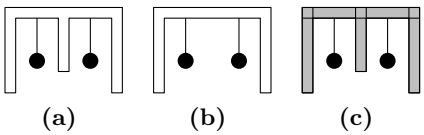


Fig. 4: Loopholes

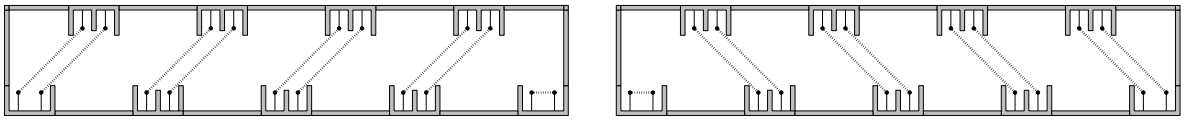
Theorem 2. *Let G be a plane geometric graph with k leaves. It is NP-complete to decide whether it is possible to augment G with $k/2$ edges such that G becomes bridge- or biconnected.*

Note 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.

Proof. The proof is again by reduction from Planar3SAT. We give gadgets for representing variables, pipes and clauses. One basic building block is what we call a *loophole*. The default loophole, depicted in Figure 4a, consists of a U-shaped cycle and two attached I-shapes such that the leaves of the I-shapes 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 4b), the two leaves do see each other. 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.

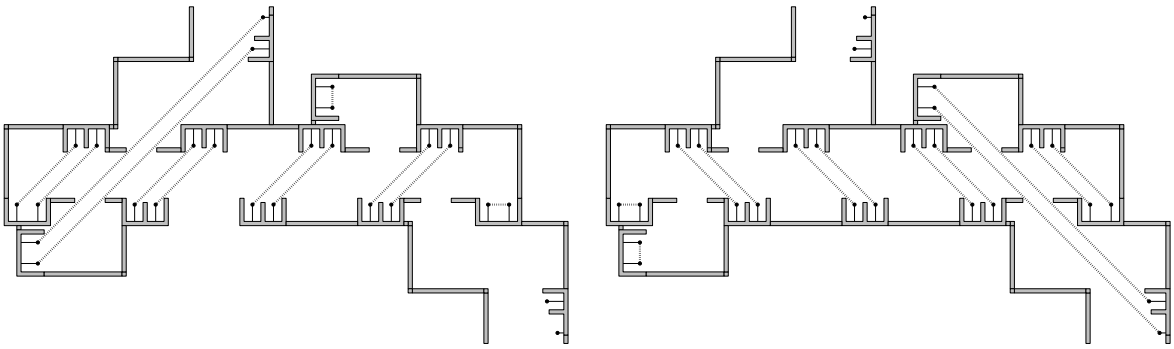
We are now ready to describe our gadget for representing a variable. It consists of two parallel lines of evenly spaced loopholes with the upper loopholes pointing downwards and the lower ones pointing upwards. The lower line contains one more loophole than the upper one and its two outermost loopholes are self-connecting. The lines are aligned such that every loophole (except the first and last of the lower line) lies horizontally between two opposing loopholes (its partners) on the other line. The distances between the loopholes and the two lines are chosen such that the I-shapes of each loophole can only be connected to the leaves of its two partners on the other line without producing a crossing.

For any minimum augmentation the two I-shapes of a loophole on the upper line must be connected to the two I-shapes of its left or right partner on the other line 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 line, otherwise crossings would occur. On the lower line, depending on the choice either the first or last loophole does not receive edges and its I-shapes must



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

Fig. 5: Variable gadget with its two states.



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

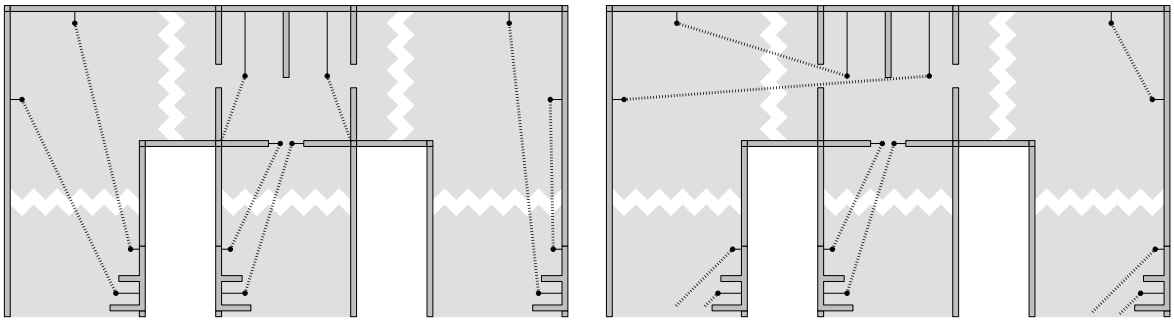
Fig. 6: Pipe gadgets attached to a variable. The left pipe transmits the logical value of the variable while the right one transmits its negation.

be connected. Hence a variable has exactly two different optimal augmentations. The two possible states are shown in Figure 5. We say that the variable is in state *true* if the edges connecting loophole partners have slope 1 and *false* if it is -1.

Next we show how the state of a variable can be copied and transmitted. The rough idea is to remove some of the walls of the variable and to attach a self-connecting loophole on one side and a skewed loophole (loophole that misses one wall, the reason for this will become clear later) on the other side such that they can be connected if and only if the augmentation in the variable has slope 1 (slope -1). Figure 6 shows a pipe that passes the state of the variable above the variable and a second pipe that transmits its negation below. In this way, if the variable has the right state, the leaves of the skewed loophole may be matched by the corresponding self-connecting loophole of the pipe if 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 will present next.

The clause gadget consists of a square that contains two I-shapes separated from each other by a wall. On three sides, the square is connected to corridors that lead to the variables that form the clause. Each corridor contains two I-shapes that are positioned such that they see (a) each other, (b) the two I-shapes inside the inner square of the gadget, and (c) the two I-shapes of the skewed loophole where the clause is attached to a variable. It is not hard to see that if any of the skewed loophole is already augmented by the variable then the two I-shapes in the corridor are free to connect to the two I-shapes in the square. Only if all skewed loopholes require the corresponding I-shapes in their corridor for augmentation the two I-shapes in the center square require an additional edge. The gadget is shown in Figure 7.

It is not hard to see that the graph resulting from our construction is connected if the planar variable–clause graph of the given formula is connected. □



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

Fig. 7: Gadget representing a clause. The left figure shows the state *false* which occurs if all pipes transmit the value *false*. The right figure shows one of the other seven cases where at least one pipe transmits the value *true*.

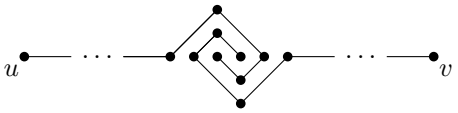


Fig. 8: Construction that removes a cycle, locally leaves only one possibility to augment, and does not interfere with the remainder of the graph with respect to augmentation.

By a simple extension we get the following result.

Corollary 1. *Let T be a plane geometric tree with k leaves. It is NP-complete to decide whether T 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 plane geometric graph with k' 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 8. Note that we can make the spiral in the center of the construction arbitrary small such that is does not prevent any connections in the remainder of the graph. We iterate this construction until there are no cycles left. Denote the resulting graph, which is in fact a tree, by T and the number of its leaves by k. It is clear that each of the new leaves has only one possibility to connect to another leaf. Namely the one that restores the previously removed cycle. Hence T can be augmented with $k/2$ vertices iff G can be augmented with $k'/2$ edges. Clearly the reduction can be performed in polynomial time. □

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.

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.

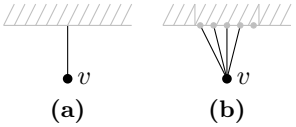


Fig. 9: Replacing an I-shape by a fat I-shape (here with $c = 4$).

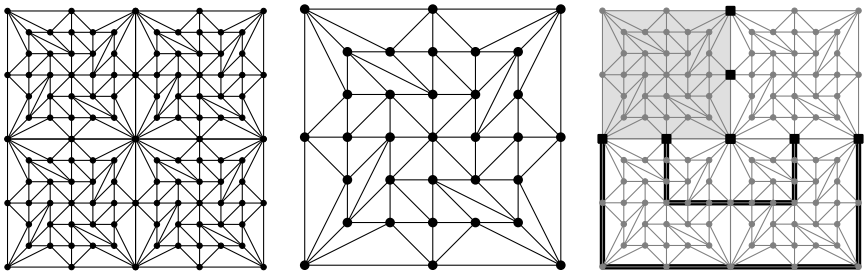


Fig. 10: A building block (a), a 5-connected sub-block with 33 vertices (b), proof that a building block is 5-vertex connected (c).

(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 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 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 remove all I-shapes from G_φ, subdivide the walls of the loopholes as in Figure 4c into gray rectangles, and replace each gray rectangle in Figures 6 and 7 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 I-shapes of G_φ. We connect each leaf of G_φ by $(c-1)$ edges to the boundary of G^1_φ such that no two leaves have a common neighbor, see Figure 9b. 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 $1 \leq c \leq 5$ 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 was 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 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 $1 \leq c \leq 5$ 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 convex geometric graphs, that is, its vertices are in convex position. We focus on augmenting a given convex geometric graph to bridge- and biconnectivity only since every convex geometric graph is outerplanar and hence contains a vertex of degree at most 2. This implies that the connectivity can never be raised further by adding edges only.

We first consider the problem of biconnecting a convex geometric graph. In Section 3.2 we give an algorithm that computes an edge set of minimum cardinality that bridge-connects a convex geometric graph. Finally in Section 3.3 we consider a weighted version of bridge-connectivity augmentation, where each edge that can be added is associated with a (positive) weight, for example its length. The goal is then to find a minimum weight augmentation. This problem is solvable in polynomial time as well in a convex geometric graph. We give an algorithm that finds a minimum weight augmentation in $O(n^2)$ time.

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

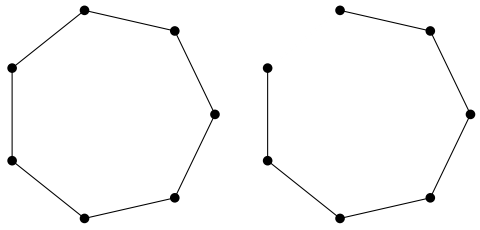


Fig. 11: A cycle (left) and a near-cycle (right).

3.1 Biconnecting Convex Geometric Graphs

Consider an arbitrary connected convex geometric graph G. Suppose that there are two consecutive points u, v on the convex hull that are not connected by an edge. Adding the missing outer edge would create a new face F. It's 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 [Mel87], this can be done in linear time.

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. However it can be bridge-connected by adding the single missing edge to form a cycle. The basic idea is to decompose an arbitrary convex geometric graph into cycles and near cycles and use this decomposition to give an edge set of minimum cardinality that bridge-connects the graph.

We now introduce two types of edges: An edge between two consecutive points of the convex hull is called an *outer edge*. An edge that is not an outer edge is called an *inner edge*. Note that if G is a connected convex geometric graph that does *not* contain an inner edge then G must be a cycle or a near-cycle.

An inner edge $e = uv$ can be used to decompose the graph into two subgraphs. The edge e divides the vertex set of G into two convex point sets P_{left} and P_{right}, where P_{left} and P_{right} are the sets of all points to the left resp. right of the edge e, respectively. The points u and v are included in both sets. The graphs G_{left} and G_{right} are the subgraphs of G induced by P_{left} and P_{right}, respectively. We call this operation *splitting G at e*. We now prove the existence of an inner edge that splits the graph in a certain way.

Lemma 1. *Let G be a convex geometric graph and f an outer edge of G. If G is not a cycle or a near-cycle then there exists an inner edge e such that one of the corresponding graphs G_{left} resp. G_{right} is a cycle or a near-cycle that does not contain f.*

Proof. If G does not have any inner edges then it must be either a cycle or a near-cycle. Suppose this is not the case and there is an inner edge e'. Now split at e' and obtain G_{left} and G_{right}. Witout loss of generality G_{left} does not contain f. If G_{left} is a (near-)cycle we're done using $e := e'$. Otherwise observe that e' is an outer edge in G_{left}. Hence by induction there exists an edge e that splits off a (near-)cycle of G_{left} that does not contain e. Since e is the only edge that G_{left} and G_{right} have in common, the edge e actually splits off the cycle from G. Further it does not contain f since f is not even in G_{left}. □

Now consider such an edge $e = uv$ and the corresponding subgraphs G_{left} and G_{right} where without loss of generality G_{left} is a (near-)cycle. Any edge e' that can be added to G such that $G + e'$ is planar must connect two points in the same subgraph G_{left} or G_{right} since it would cross e otherwise. If G_{left} is a near-cycle then we can add a single edge to make it bridge-connected. As the edge must be inside G_{left} this is optimal. So by now we're in the case that G_{left} is a cycle. We first note that any edge that connects a vertex of G_{right} to a vertex of G_{left} must connect to either u or v. Hence it is safe to remove all edges of the cycle except e and all of its vertices except u and v. We call the resulting graph G'.

The main problem is to decide whether we still need e for a minimum augmentation. We can decide this in a special case only: If one of its incident vertices, say u, has degree 1, we can remove e while preserving an optimal solution. This is the case since every edge that connects a vertex w to u and remains planar can also be connected to v while preserving planarity. The only difference concerning bridge-connectivity is that e will not lie on the induced cycle anymore. However this is not relevant since e already lies on another cycle. Hence in this case we may also remove e and hence u from G'. Otherwise we defer the decision and just mark e as removable. If we remove any edge e' later, we check if this creates another leaf incident to a bridge marked as removable. In that case we remove this bridge as well and continue this process until we cannot remove any more bridges. By applying induction to the above argument it can be seen that this does not change the size of an optimal solution.

We apply this process to G until it does not contain any further inner edges. This leaves us with three cases. Either the remaining graph is a cycle, a near-cycle or a single edge. In the first two cases we add at most one more edge. In the last case the remaining edge must be a bridge of G that is not marked as removable. Hence it is still a bridge in the current augmented graph. However since we worked optimally so far the optimal solution requires an additional edge as well. We simply add an additional edge to the augmented graph such that the last edge is no longer a bridge.

A straightforward implementation yields a running time of $O(n^2)$. We now show how the algorithm can be implemented to run in linear time. To achieve this, we interleave the augmentation and the search for a suitable split-edge.

The basic idea is that we do not have to compute the next split always starting with the whole graph. Instead we consider the component where we split off the last time and continue there locally. To implement this, we use the following data structures: A doubly linked list representing the convex hull of the point set. We also assume that the graph is given with the corresponding embedding, that is, the edges incident to a vertex are ordered in clockwise order and the first element is either an edge to the next point of the convex hull if it exists or the next edge in clockwise order otherwise. Further every edge has a removable-flag which is set to false initially.

We first pick an arbitrary outer edge f. Now we use inner edges to split the graph and always select the subgraph that does not contain f for processing first. When we split at an

inner edge e we first process the component that does not contain the edge f and mark e as the new edge that should not be used. When this is done the edge e is an outer edge (if it is not removed) and hence may be cut off now. Now we show the details of implementing each step.

First note that splitting at an edge e requires only the duplication of e and its two endpoints u and v and can hence be done in constant time.

The next step is to find a split-edge fast. If we cannot find a split, we are in one of the base cases. Note that in that case we have enough time to check each edge and to find the missing edge for the cycle if necessary, as this happens at most twice to every edge. Hence the overall running time of this part is linear.

We now show how to find an inner edge. In the beginning we pick a start-vertex incident to f. We now inspect the edges incident to this vertex. Using the list of our convex hull we can easily check whether a given edge is an outer edge in constant time. If the vertex is incident to an inner edge we use it to split the graph. Otherwise the vertex has at most two incident edges and we move on to the next vertex on the convex hull (in clockwise order). We do this until we either reach the start-vertex again, that is, the remaining graph is a (near-)cycle or until we find an inner edge. When splitting into two graphs, we use the vertex where we found the split as the new start-vertex in both components. This ensures that every vertex is used to split as long as it has incident inner edges. Hence all edges are considered only twice during the splitting-process. Once a vertex already used for splitting is considered again we know that the component is done. Hence this part of the Algorithm runs in linear time.

Obviously every edge can be removed at most once and hence the mark- and remove-part of the algorithm takes overall linear time, too. The results are summarized in Theorem 4.

Theorem 4. *Let $G = (V, E)$ be a convex geometric graph. If the convex hull of V and the corresponding embedding of G is given we can compute a set of edges A of minimum cardinality s.t. $G + A$ is bridge-connected in linear time and space.*

3.3 Minimum-Weight Augmentation

We now generalize the algorithm from the previous section to the case where every edge that can be added is also associated with a positive cost, e.g., its length, which we denote by $\ell(e)$. We then seek a plane minimum-cost augmentation of a given graph G such that G becomes bridge-connected. Obviously this problem is NP-complete in arbitrary geometric graphs because it contains TSP as a subproblem. For convex connected geometric graphs, however, the problem can be solved in polynomial time. Here we give an algorithm that finds a solution in $O(n^2)$ time.

The basic idea is again to cut off (near-)cycles. In the weighted case, however, it is more difficult to decide which solution should be taken in a near-cycle. There may be different solutions, some create a cycle that contains the edge used to cut off and some do not. If we do not put the edge on a cycle because it is cheaper to do so, this might change later, when we have to incoroporate the edge in another cycle in the remainder of the graph. Hence we cannot decide in advance but combine the approach with dynamic programming.

We give each edge e a weight $w(e)$. Now consider a near-cycle C. The edge-weights are chosen in such a way, that we can leave a given edge e to be a bridge (not contained in any cycle) at the cost of $w(e)$. In the beginning all weights are set to ∞. Now we cut off a

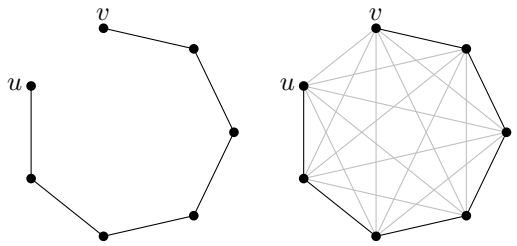


Fig. 12: Minimum weight augmentation of a near-cycle.

near-cycle. The situation is as follows: Given a near-cycle C that was cut off at an edge e find an edge-set E that minimizes

$$\sum_{e \in E} \ell(e) + \sum_{\substack{e \in C \\ e \text{ bridge in } C+E}} w(e).$$

For the moment assume we could solve such a problem. Then we could do so once with an edge-weight of 0 assigned to e and once with an edge-weight of ∞. The latter choice forces e on a cycle, whereas the first one might be cheaper. We store both solutions and use the difference as the new weight of e. We then remove all edges of C except e from G. When we process the edge e a second time with the other near-cycle that contains it, this weight is exactly the cost of switching the solution in the previous cycle such that we may leave it a bridge in the current cycle. When we encounter a cycle we simply set the weight of the cut-edge to 0. When we have processed the last (near-)cycle and G is empty, we use the stored information to compute a minimum weight planar augmentation of G.

It remains to show how a solution can be found for the case of a near-cycle C. Let u and v be the two leaves of C. It can be seen that a solution corresponds to a shortest path from u to v in the graph depicted in Figure 12, where the weight of each gray edge is its length. Note that every edge may only be used in direction from u to v, never backwards because this would result in a crossing. The shortest-path-problem can of course be solved in $O(n^2)$ time using Dijkstra's algorithm [CLRS01]. Since every edge is processed at most twice in this manner the overall running time is in $O(n^2)$.

We have proved the following theorem:

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

4 Path Augmentation

In this section we consider the following problems:

Planar k-path augmentation (k-PathAug)G

> iven 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 can be triangulated and thus be augmented to contain three vertex-disjoint paths between any two vertices.

4.1 Planar 2-Path Augmentation

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

Proof. We prove the edge-variant; the vertex-variant can be shown 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 13. Abellanas et al. [AGH+08] came up with this example to show that, for trees, geometric PECA sometimes requires $n/2$ edges.

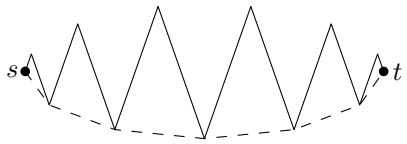


Fig. 13: Zig-zag path of n vertices that needs $n/2$ edges (dashed) to augment [AGH+08].

They conjectured that $n/2$ edges always suffice to augment a tree to bridge-connectivity. Recently, Tóth [Tót08] 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 of T except perhaps the outer face are triangles. This follows from the fact that every simple polygon has a triangulation [dBCvKO08].

Lemma 2. *Let $S \subset \mathbb{R}^2$ a finite set of points. Let $G = (S, E)$ and $G' = (S, E')$ be connected plane geometric graphs such that $E \subseteq E'$ and let $s, t \in S$. 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 it holds that $|N^i(v)| \geq 2i+1$ or $N^i(v) = S$. The proof is by induction on i. For $i = 0$ the statement is clearly true.

We now treat the case $i > 0$. We can assume that $N^i(v) \neq S$. By the induction hypothesis it holds that $|N^{i-1}(v)| \geq 2i-1$. Hence it remains to show that $|\partial N^i(v)| \geq 2$. This is obvious since otherwise T would not be 2-connected.

Let $k = \min\{i \mid N^i(s) \cap N^i(t) \neq \emptyset\}$. It is clear that T contains an s–t path of length at most $2k$. We now bound k from above. The disjointness of $N^{k-1}(s)$ and $N^{k-1}(t)$ and our observation above yield $m := |N^{k-1}(s) \cup N^{k-1}(t)| \geq 4k-2$. This implies $k \leq (m+2)/4$.

We now do a simple case analysis depending on m. In case $m \leq n-2$, we get $k \leq n/4$, which results in a path of length at most $2k = n/2$. The case $m = n-1$ is impossible since it would imply the existence of a cut vertex, namely the only vertex in $S \setminus (N^{k-1}(s) \cup N^{k-1}(t))$. The last case, $m = n$, implies that $N^k(s) \cap N^{k-1}(t) \neq \emptyset$. Hence T contains an s–t path of length at most $2k-1 = n/2$. □

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 augmenation 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 that 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 suppose $i > 1$. We apply the induction hypothesis to $N^{i-1}(v)$ and show that $|\partial N^i(v)| \geq 3$. The cases $|\partial N^i(v)| \in \{0, 1\}$ can be excluded as in the proof of Lemma 3. Thus suppose $|\partial N^i(v)| = 2$. 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 a contradiction to the fact that every fully triangulated graph is 3-connected.

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 reach the convex hull.

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 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 as shown in Lemma 2.

Concerning step 1, note that the boundary of the outer face of a plane geometric graph is generally 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 [AGH+08].

Therefore we can apply Melkman's linear-time algorithm [Mel87] 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 [Cha91].

As for step 2, a shortest s–t path π in the triangulation T can be found in linear time with BFS. It remains to show how to implement step 3 that is, how to find and 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 path from s to t in G. Now remove all bridges of G that are used by π'. Call the resulting graph G'. We number the connected components of G' as they occur along π' and label the vertices of each component accordingly. Throughout the algorithm we maintain the invariant that all vertices of π with label up to $j (j = 0, 1, \dots)$ lie in the same 2-edge connected component of G. This clearly holds for $j = 0$. The preservation of the invariant yields the correctness of the algorithm.

As in the proof of Lemma 2 we go through the edges of π in order, starting from s. We consider three cases. First, if the endpoints of the current edge $e = \{u, v\}$ have the same label, we do not need to add any edges and simply advance to the next edge of π. As u and v have the same label, e is not a bridge and thus the invariant is preserved.

Second, if u and v have different labels and e is not in G, we add e to G and set j to the label of v. This preserves the invariant: By induction we have that, before the addition of e, s and u belong to the same 2-edge connected component. Since u and v are connected in G, they are 2-edge connected in $G + e$. By transitivity we have that s and all vertices of π with label at most j are in the same 2-edge connected component.

Third, if u and v have different labels and e is already in G (that is, e is a bridge in G). In the triangulation T, the edge e forms a triangle $\{u, v, w\}$. Hence there are two candidates for adding an edge to G: either uw or vw. If u and w have different labels, we add the edge uw and set j to the label of w. If u and w have the same label, we add vw and set j to the label of v. This again preserves our invariant.

In summary, we have proved the following theorem.

Theorem 8. *Let $G = (S, E)$ be any plane connected geometric graph with n vertices, and let s and t be any two 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 $G = (S, E)$ 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* 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 14. Hence we have removed an s–t separating chord without introducing a new one. Inductively we obtain the desired triangulation T_G. □

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.

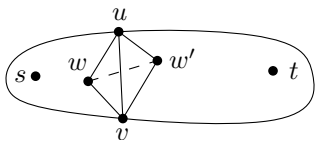


Fig. 14: Removing an s–t separating chord uv by flipping.

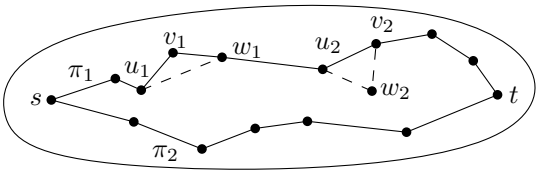


Fig. 15: Hand rail construction along path π_1.

Since every plane geometric graph can be triangulated we characterize the triangulations that contain three edge-disjoint s–t paths.

Theorem 10. *Let $T = (S, E)$ be any 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. Since T is biconnected, there exist two vertex-disjoint s–t paths π_1 and π_2 by 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. 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 belongs to π_1, we let $s^* = s_2$, otherwise we add 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 the vertex w lies next to v on π_1 in the direction of t, we add the edge uw to π_3, otherwise we add uw and uv, see Figure 15. In neither case 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. □

Acknowledgments

We thank Csaba Tóth for the idea behind Theorem 3.

References

[AGH+08] Manuel Abellanas, Alfredo García, Ferran Hurtado, Javier Tejel, and Jorge Urrutia. Augmenting the connectivity of geometric graphs. *Comput. Geom. Theory Appl.*, 40(3):220–230, 2008.

[Cha91] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(1):485–524, 1991.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 2nd edition, 2001.

[dBCvKO08] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, 3rd edition, 2008.

[ET76] Kapali P. Eswaran and R. Endre Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976.

[FJ81] Greg N. Frederickson and Joseph Ja'Ja'. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981.

[FM98] Sergej Fialko and Petra 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.

[Hsu02] Tsan-sheng Hsu. Simpler and faster biconnectivity augmentation. *J. Algorithms*, 45(1):55–71, 2002.

[Kan93] Goos Kant. *Algorithms for Drawing Planar Graphs.* PhD thesis, University of Utrecht, 1993.

[Kan96] Goos Kant. Augmenting outerplanar graphs. *J. Algorithms*, 21(1):1–25, 1996.

[KB91] Goos Kant and Hans L. Bodlaender. Planar graph augmentation problems. In Frank Dehne, Jörg-Rüdiger Sack, and Nicola Santoro, editors, *Proc. 2nd Workshop Algorithms and Data Structures (WADS'91)*, volume 519 of *Lecture Notes Comput. Sci.*, pages 286–298. Springer-Verlag, 1991.

[KR92] Donald E. Knuth and Arvind Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.

[Lic82] David Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[Mel87] Avraham A. Melkman. On-line construction of the convex hull of a simple polyline. *Inform. Process. Lett.*, 25(1):11–12, 1987.

[PB99] J. Scott Provan and Roger C. Burk. Two-connected augmentation problems in planar graphs. *J. Algorithms*, 32:87–107, 1999.

[Rap89] David Rappaport. Computing simple circuits from a set of line segments is NP-complete. *SIAM J. Comput.*, 18(6):1128–1139, 1989.

[Tót08] Csaba D. Tóth. Connectivity augmentation in plane straight line graphs. *Electronic Notes in Discrete Mathematics*, 31:49–52, 2008.