Connecting Obstacles in Vertex-Disjoint Paths

Marwan Al-Jubeh* Gill Barequet†* Mashhood Ishaque*
Diane L. Souvaine* Csaba D. Tóth‡* Andrew Winslow*

Abstract

Given a set of k disjoint convex polygonal obstacles inside a triangular container, we add straight-line noncrossing edges such that each obstacle has three vertex-disjoint paths to the container. We prove combinatorial bounds on the minimum number of edges that are always sufficient and sometimes necessary.

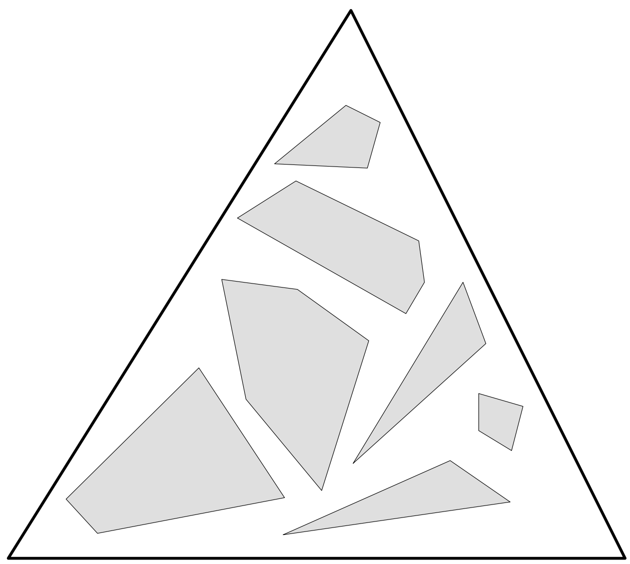

Figure 1: A triangular container with disjoint convex obstacles.

1 Introduction

A given graph is said to be k-connected if it remains connected upon deleting any $k-1$ vertices along with the incident edges. A k-connected graph has k vertex-disjoint paths between any two nodes. An important area of research in graph theory and computational geometry is the problem of connectivity augmentation. The k-connectivity augmentation problem asks for the minimum number of edges needed to augment a graph to make it k-connected. Edge-connectivity augmentation is defined analogously.

In abstract graphs, the connectivity augmentation problem can be solved in linear time for $k = 2$ [ET76, Ple76], and in polynomial time for any fixed k [JJ05]. For a given planar graph, the augmentation that has

*Dept. of Computer Science, Tufts University, Medford, MA. This material is based upon work supported by the National Science Foundation under Grant No. 0830734. {maljub01,barequet,mishaque,dls,awinsl02}@cs.tufts.edu

†Dept. of Computer Science, Technion, Haifa, Israel

‡Dept. of Mathematics, University of Calgary, AB, Canada. cdtoth@ucalgary.ca

to preserve graph planarity, is called *planarity-preserving* augmentation. Unfortunately, the problem is NP-hard even for $k = 2$ [BKB91]. For a given planar graph that has already been embedded in the plane, if the augmentation has to respect the given embedding, the augmentation is said to be *embedding preserving*. For a planar straight-line graph, the minimum embedding-preserving augmentation using noncrossing straight-line edges is NP-Hard for any $2 \leq k \leq 5$ [RW08].

There are two possible approaches to get around the NP-Hardness of the augmentation problem: (*i*) approximation algorithms (e.g., there is a 2-approximation algorithm for planarity-preserving connectivity augmentation for $k = 2$, which runs in $O(n \log n)$ time [BKB91]); and (*ii*) proving combinatorial bounds on the number of new edges in terms of the number of vertices (e.g., Al-Jubeh *et al.* [AJIR$^+$09] show that $2n - 2$ new edges are always sufficient and sometimes necessary for the embedding-preserving 3-edge-connectivity augmentation of a planar straight line graph with n vertices if augmentation is possible). Tóth and Valtr [TV09] characterized the planar straight line graphs that can be augmented to 3-connectivity. These graphs are called *3-augmentable*. It remains an open problem what is the minimum number of new edges that are sufficient for the 3-connectivity augmentation of every 3-augmentable planar straight line graphs with n vertices.

In this paper we consider a special type of augmentation problem (see the formulation below) and provide combinatorial bounds on the minimum number of necessary and sufficient new edges.

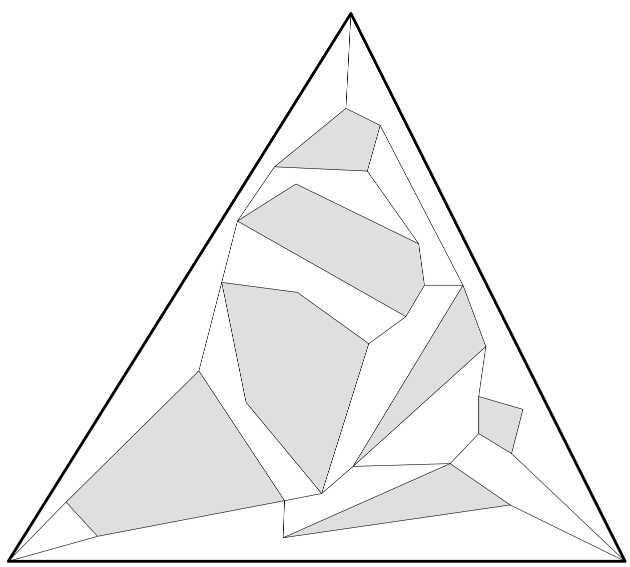

Figure 2: Adding noncrossing straight-line edges so as to make each obstacle connected by three vertex-disjoint paths to the triangular container.

1.1 Problem Definition

Given a set of k disjoint convex polygonal obstacles inside a triangular container, add straight-line non-crossing edges such that each obstacle has 3 vertex-disjoint paths to the three vertices of the container. The three paths should start at distinct vertices of the obstacle and end at distinct vertices of the container. They can use the edges of the obstacles arbitrarily.

1.2 When is Augmentation Possible?

If the obstacles are not convex, it might not be possible at all to add edges such that each obstacle has three vertex-disjoint paths to the container. In Figure 3 the inner-most obstacle "sees" only three other vertices, all of which belong to the same obstacle. Since it is not possible to route three vertex disjoint paths along the same obstacle without adding edges in the interior of the obstacle, this example is not augmentable.

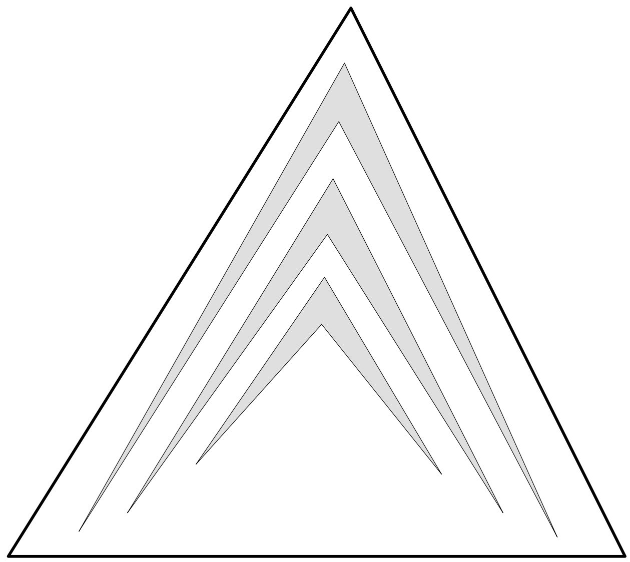

Figure 3: Non-convex obstacles may not be connected to the boundary by three vertex-disjoint paths.

For a set of disjoint convex obstacles inside the triangular container, every triangulation of the free space around the obstacles is a 3-connected graph [TV09]. It is easy to see that there are three vertex-disjoint paths from every obstacle to the container along the edges of a triangulation. For any particular obstacle, add a new internal node p and connect it to the boundary of the obstacle at three distinct vertices. Similarly, add a node q outside the triangular container and add the three edges connecting q to the corners of the container. It can be easily verified that the new graph is still 3-connected, which implies that there are three vertex-disjoint paths from p to q. Hence, there are three vertex-disjoint paths that start at distinct vertices of the obstacle and end at distinct vertices of the container. These three paths can be determined using any max-flow algorithm [AMO93].

Although a triangulation contains the desired augmentation as a subgraph, it may contain too many edges. In this paper we show how to perform this augmentation by using much fewer edges.

1.3 Our Results

- For k convex obstacles, where k can be arbitrarily large, $3k - 1$ edges are sometimes necessary (Section 2).
- For k convex obstacles, where k can be arbitrarily large but each obstacle has at most s sides, $3k - \frac{k-1}{s-1}$ edges are sometimes necessary (Section 2).
- For k convex obstacles, $3k$ edges are always enough (Section 3).

Once each obstacle has three vertex-disjoint paths to the container, we can get a 3-connected planar graph by adding an edge for each degree-2 vertex [AJIR$^+$09].

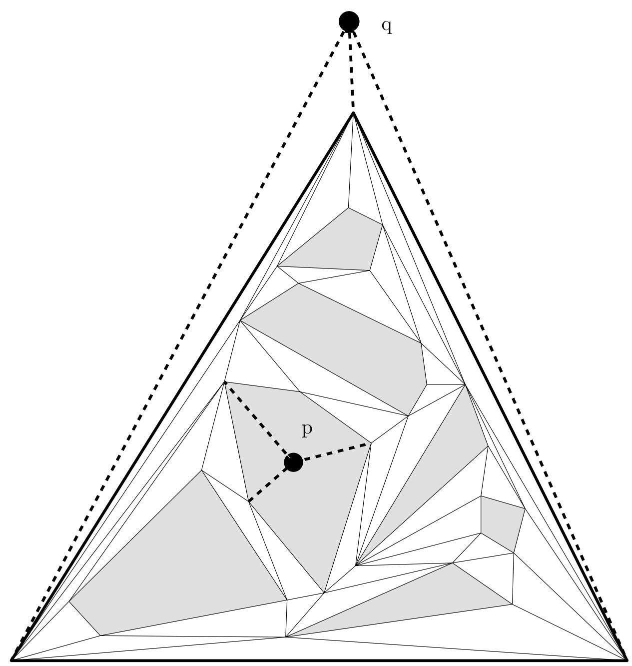

Figure 4: A triangulation of the free space around convex obstacles in a triangular container is a 3-connected graph, and it contains the desired augmentation as a subgraph.

2 Lower Bound Constructions

When there is only one convex obstacle, three edges are obviously required (and sufficient) for connecting it to the container. However, for k (an arbitrarily-large k) convex obstacles, at least $3k - 1$ edges are necessary in the worst case. Our lower bound construction is depicted in Figure 5. It includes one large convex obstacle which hides one small obstacle behind each side (except the base), such that each small obstacle can "see" only three different vertices (the top vertex of the container and two adjacent vertices of the large obstacle). Thus, we need three edges for each small obstacle and only two edges for the larger obstacle, connecting its two bottom vertices to the two endpoints of the base of the container.

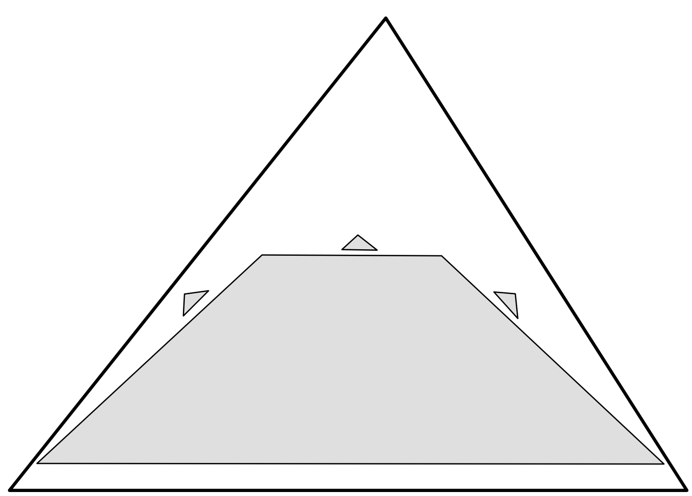

Figure 5: k convex obstacles, edges needed: $3k - 1$.

The large obstacle in the above construction is a convex k-gon, and so the lower bound $3k-1$ does not hold if the every obstacle has at most s sides, for some fixed $3 \leq s < k$. In that case we use a similar construction, in which a big s-sided obstacle hides $s-1$ smaller obstacles behind all its sides except one, and the construction is repeated recursively. This construction corresponds to a complete tree with branching factor $s-1$, in which the smaller obstacles are the children of a larger obstacle. For a fixed value of s, we set h as the height of the complete $(s-1)$-ary tree. Thus, the number of obstacles,

$$k = \frac{(s-1)^h - 1}{s-2}, \tag{1}$$

can be as high as we desire. The number of leaves in the tree is $(s-1)^{h-1}$. A simple manipulation of Equation 1 shows that this number equals $k - \frac{k-1}{s-1}$. Hence, the number of internal nodes in the tree is $\frac{k-1}{s-1}$. For the 3-vertex-disjoint path augmentation, each leaf obstacle needs three edges and each non-leaf obstacle needs two edges. The total number of edges required is, thus,

$$3\left(k - \frac{k-1}{s-1}\right) + 2\left(\frac{k-1}{s-1}\right) = 3k - \frac{k-1}{s-1},$$

which ranges from $\frac{5}{2}k + \frac{1}{2}$ to $3k-1$ for $3 \leq s \leq k$.

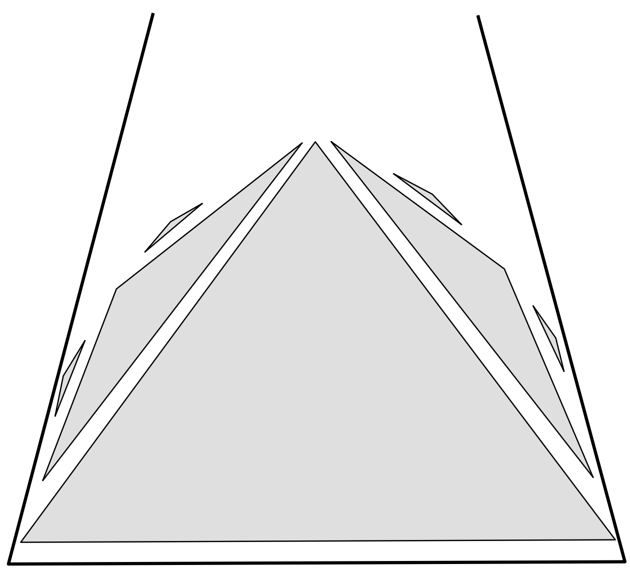

Figure 6: Construction for triangular obstacles.

3 The Upper Bound

We now prove that $3k$ edges are always sufficient for making the given set of obstacles O 3-vertex-connected to the triangular container C. Initially, there exists a triangulation T of the free space inside C that is the 3-connected, which is not always true for non-convex containers. The algorithm recurses such that each subproblem is on a polygonal container P with 3-connected triangulation (Lemmas 1 and 2).

We designate the three corners of the C with the colors red (v_R), green (v_G), and blue (v_B). Each obstacle is charged up to three times, once for each color. An obstacle is marked to indicate its connection to a particular colored corner of the container. If a path to a designated vertex goes through another obstacle, then the latter obstacle is charged for one of the edges. For each edge at least one obstacle is charged, and no obstacle is charged more than thrice, which implies that the entire process adds at most $3k$ edges. The procedure AUGMENT implements this process, which is invoked by a call AUGMENT(C, v_R, v_G, v_B).

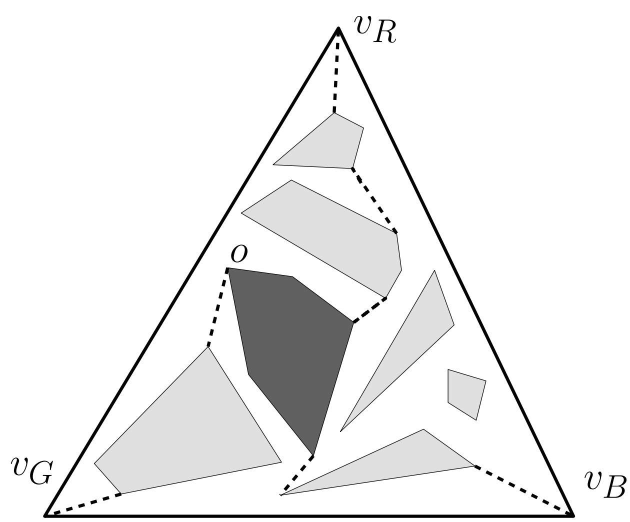

Figure 7: Vertex-disjoint paths from the obstacle o.

Algorithm 1 AUGMENT(P, v_R, v_G, v_B)

```
Pick an arbitrary obstacle o inside P.
Find three vertex-disjoint paths π_R, π_G, and π_B to the vertices v_R, v_G, and v_B, respectively.
for all paths π_i, where i ϵ {R, G, B} do
   π_i = SHORTENPATH(π_i)
   for all edges e along the path π_i from o to v_i do
      Mark the obstacle incident to e for v_i
      if e is a part of some obstacle boundary then
         Go to next edge.
      else if e in incident to the boundary of P then
         Add the edge e and exit loop.
      else if e in incident to the vertex v_i then
         Add the edge e and exit loop.
      else if e is incident to an marked obstacle then
         Add the edge e and exit loop.
      else
         Add the edge e.
      end if
   end for
end for
HANDLESUBPROBLEM(P, o, π_R, π_G)
HANDLESUBPROBLEM(P, o, π_R, π_B)
HANDLESUBPROBLEM(P, o, π_B, π_G)
```

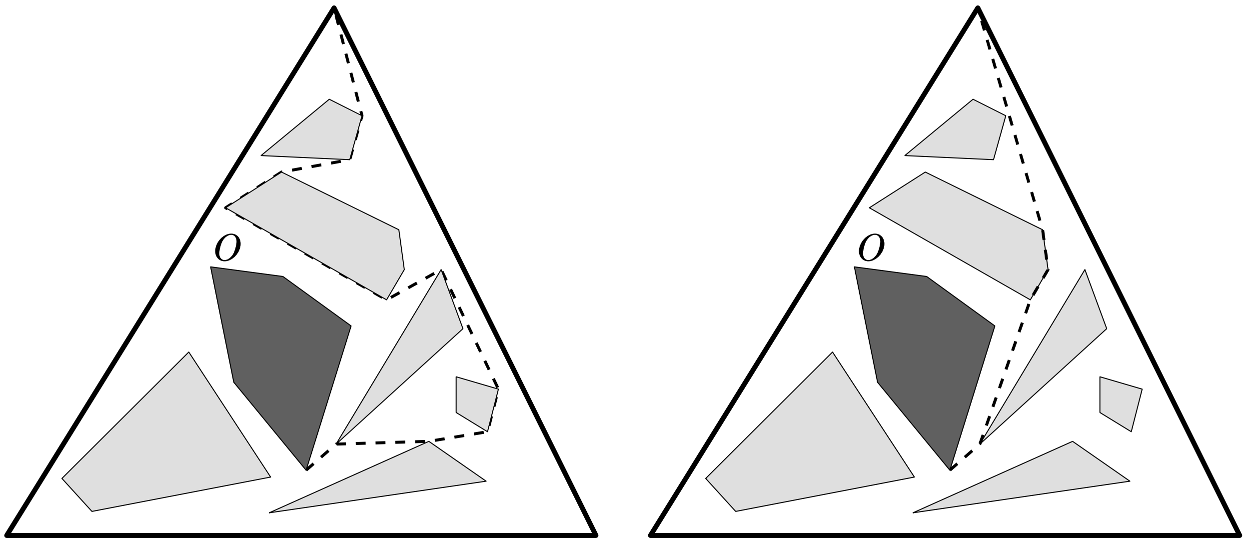

Figure 8: Shortening a vertex disjoint path.

Algorithm 2 SHORTENPATH(π)

Let $\{v_1, v_2, \dots, v_m\}$ be the vertices in path π.
while for some $i < j - 1$, v_i and v_j see each other or are incident on the same obstacle **do**
  Let P' be closed polygon formed by π and the line segment $v_i v_j$. Assume we are allowed to travel along π.
  Let $\pi_{i,j}$ be shortest geodesic path between v_i and v_j inside P'.
  Replace the portion of π between v_i and v_j by $\pi_{i,j}$.
  Exit loop when π stops changing.
end while
return π

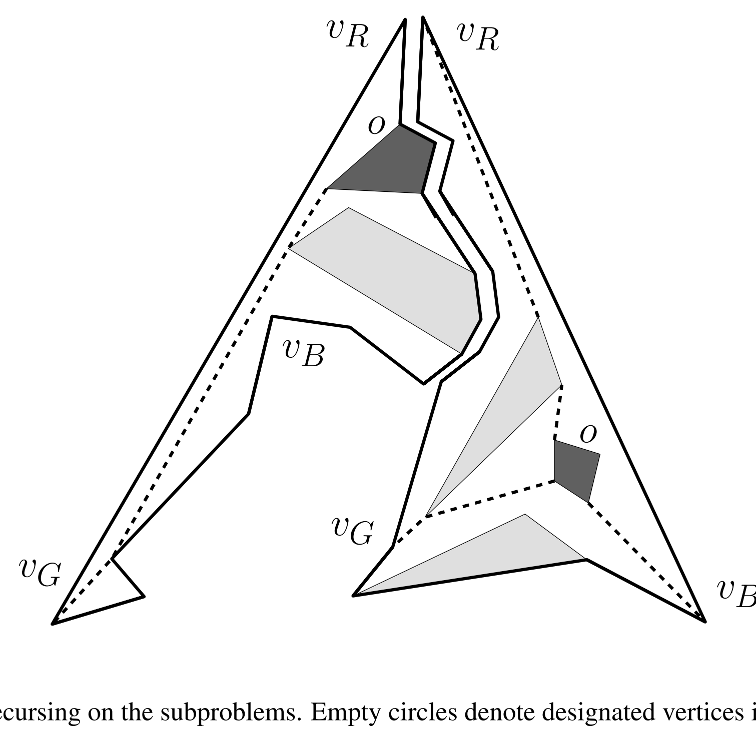

Figure 9: Recursing on the subproblems. Empty circles denote designated vertices in subproblems.

Algorithm 3 HANDLESUBPROBLEM(P, o, π_i, π_j)

Obstacle o together with π_i and π_j creates a closed polygon P' inside P
Let v_i, v_j be the designated vertices of the paths π_i and π_j.
Let $l \in \{R, G, B\} \setminus \{i, j\}$.
Designate a vertex on the obstacle o as v_l.
if There is a 3-connected triangulation of P' **then**
  AUGMENT(P', v_i, v_j, v_l)
else
  Let C_1 be an extremal 2-cut.
  Let P_1 be the polygon created by C_1.
  Let v_R be one of the designated vertices the right of the C_1 (w.l.o.g).
  Designate the two vertices of the 2-cut as V_G and V_B
  AUGMENT(P_1, v_R, v_G, v_B)
  HANDLESUBPROBLEM$(P \setminus P_1, C_1, \pi_i, \pi_j)$
end if

Lemma 1 *For a polygon P such that every triangulation of P contains a 2-cut C_i then all the designated vertices on P are not on the same side of C_i.*

Proof. As a result of the subroutine SHORTENPATH, the polygonal boundary on the either side of any 2-cut cannot consist of only one path. Since there are always two vertex disjoint paths forming the polygonal boundary, there must a designated vertex or a vertex of the obstacle o present. □

Lemma 2 *Given three vertex-disjoint paths from an obstacle to v_R, v_G, and v_B, the path to v_R cannot touch the boundary of the polygon P between the vertices v_G and v_B.*

Proof. The lemma follows from the fact that the three paths are vertex disjoint. □

4 Open problems

- Close the gap between the lower and upper bounds. We conjecture that the lower bound is the correct one. Hence, give an augmentation algorithm that adds only $3k - \frac{k-1}{s-1}$ edges.
- Provide combinatorial bounds for 3-connectivity augmentation of *2-regular* graphs.
- Similarly, set combinatorial bounds for 3-connectivity augmentation of a set of line segments (a 1-regular graph).

References

[AJIR+09] Marwan Al-Jubeh, Mashhood Ishaque, Kristóf Rédei, Diane L. Souvaine, and Csaba D. Tóth. Tri-edge-connectivity augmentation for planar straight line graphs. In *ISAAC*, pages 902–912, 2009.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, February 1993.

[BKB91] P. O. Bex, Goos Kant, and Hans L. Bodlaender. Planar graph augmentation problems. In *WADS*, pages 286–298, 1991.

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

[JJ05] Bill Jackson and Tibor Jordán. Independence free graphs and vertex connectivity augmentation. *J. Comb. Theory Ser. B*, 94(1):31–77, 2005.

[Ple76] Ján Plesník. Minimum block containing a given graph. *Archiv der Mathematik*, 27(1):668–672, 1976.

[RW08] Ignaz Rutter and Alexander Wolff. Augmenting the connectivity of planar and geometric graphs. *Electronic Notes in Discrete Mathematics*, 31:53–56, 2008.

[TV09] Csaba D. Tóth and Pavel Valtr. Augmenting the edge connectivity of planar straight line graphs to three. In *XIII Spanish Meeting on Comput. Geom.*, 2009.