Connectivity Augmentation of Plane Graphs

Krishnan Dehaleesan
University of Bergen, Norway

Asif Khan
Chennai Mathematical Institute, India

Pranabendu Misra
Chennai Mathematical Institute, India

Abstract

We study the problem of connectivity augmentation of a planar graph, while preserving planarity. This problem is motivated by many real-world settings such as road-networks, power-networks etc. In these settings, it is crucial to preserve the original planar embedding after augmentation.

In 2009, Gutwenger and Mutzel gave a constructive algorithm showing that a connected planar graph with a fixed embedding (a plane graph) can be optimally augmented to a biconnected graph without crossings while preserving the embedding. We further this line of research, by giving an algorithm that computes a minimum set of edges that makes a connected plane graph 2-edge-connected in $O(|V|(1+\alpha(|V|)))$ time and linear space, where α is the inverse Ackermann function.

We also study the 3-vertex-connectivity augmentation of biconnected outerplanar plane graphs. We present the first polynomial-time algorithm that augments such graphs to 3-connectivity with the minimum number of edges in $O(|V|(1+\alpha(|V|)))$ time and linear space while preserving the embedding, i.e. the augmented graph has a planar embedding that extends the given embedding.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph algorithms

Keywords and phrases Connectivity augmentation, Plane graphs, Bridgetree, BC-tree, Balanced graph

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.23

Related Version *Full Version*: `https://krishnand-26.github.io/papers/plane-aug.pdf`

Funding *Asif Khan*: Partially funded by a grant from Infosys foundation.

1 Introduction

The *connectivity augmentation* problem asks to increase the connectivity of a given graph to a specified level while adding as few edges as possible. This is a classic problem in graph algorithms and network design, motivated by applications in improving connectivity of real world networks, such as roads, electricity, telecommunication, and so on. While it is certainly possible to augment the connectivity by increasing the capacity of existing links, or adding parallel links, this is sub-optimal in both the costs, and in improving the network resiliency to link failures. For example, if a link in an electricity-network has failed due to some real-world incident (e.g a falling tree), it is highly likely that any parallel links would have also have failed. Hence, it is highly desirable to have connectivity via some alternative route. This motivates the algorithmic study of connectivity augmentation problems, which play a fundamental role in the design of reliable communication and infrastructure networks. For general graphs, both biconnectivity augmentation (making a graph 2-vertex-connected) and bridge-connectivity augmentation (making a graph 2-edge-connected), with uniform costs for new-links, admit linear-time algorithms [5]. However, they become NP-hard when new links have different costs.

A variant of connectivity augmentation problems considers the case where the input graph is planar and we want the augmented graph to be planar as well. This is directly motivated by real-world settings such as augmentation of road networks, electricity networks, etc. that are always embedded on a plane, and hence a plane solution is highly preferred. This variant, called *Planar BiConnectivity Augmentation* (PVCA), is known to be NP-hard [16], even with uniform new-link costs. The corresponding edge-connectivity version, *Planar BridgeConnectivity Augmentation* (PECA) is also NP-hard [18].

Let us observe that, in many motivating real-world problems, such as augmenting a road-network, not only is the input graph planar, but it has a fixed embedding of the nodes in the plane which cannot be changed. Therefore, it is highly desirable that upon augmenting the graph with new-links, the embedding is preserved. This motivates us to study the following problem:

Planar Bridgeconnectivity Augmentation for Fixed Embedding (PECA-Fix)

Input: A connected planar graph G with a fixed embedding.

Task: Find a minimum set of edges $E' \subset V \times V$ such that $G + E'$ has a planar embedding that extends the planar embedding of G and is 2-edge connected.

The vertex-connectivity variant of PECA-Fix, called PVCA-Fix, can be defined similarly. This was studied by Gutwenger, Mutzel and Zey [9] who showed that solving biconnectivity augmentation for planar graphs with a fixed embedding (PVCA-Fix) is polynomial-time solvable in near-linear time. Since plane biconnectivity augmentation was poly-time solvable, it was natural to ask the question for connectivity greater than 2. Recent work by Akitaya et al. [1] showed that, even for plane graphs, increasing connectivity from c to k is NP-hard for all $2 \leq c < k \leq 5$. This motivated us to study the tri-connectivity augmentation of graphs restricted a subclass of planar graphs, namely outerplanar graphs, with a fixed embedding:

Outerplanar Triconnectivity Augmentation for Fixed Embedding (OTA-Fix)

Input: An Outerplane biconnected graph G with a fixed embedding.

Task: Find a minimum set of edges $E' \subset V \times V$ such that $G + E'$ has a planar embedding that extends the planar embedding of G and is 3-vertex-connected.

Note that the requirement is only that the augmented graph admits a planar embedding extending the fixed embedding of G; the augmented graph need not remain outerplanar. Indeed, except for a triangle, no 3-connected graph is outerplanar.

Let us note that the triconnectivity augmentation problem for outerplanar graphs (without a fixed embedding) is well-studied. The authors of [7] gave a polynomial time algorithm for the biconnectivity augmentation problem and the bridge-connectivity augmentation problem. For k even or $k = 3$, Nagamochi and Eades [17] provided optimal outerplanar edge-connectivity augmentation algorithms. Furthermore, Kant [15] provided an optimal solution in polynomial time for the Outerplanar triconnectivity augmentation problem.

The fixed embedding constraint rules out the known techniques for OTA-Fix. For instance, Kant's exact algorithm that augments an outerplanar graph to triconnectivity does not extend to the setting where the input graph has a fixed embedding. In particular, Kant's method may alter the embedding to a non-outerplanar one while preserving planarity, and then insert edges accordingly. To illustrate this limitation, consider the graph obtained from a cycle C_k by replacing each edge with a parallel edge and subdividing it. Kant's algorithm can make this outerplanar graph 3-connected using $\frac{k}{2}$ edges by effectively moving vertices from

the outer face into inner faces. Such modifications are not allowed in the OTA-Fix setting, where the embedding must be preserved. In fact, for the example above, no augmentation respecting the fixed embedding can achieve 3-connectivity using only $\frac{k}{2}$ edges.

This raises an interesting question: "how much worse can the solution be when the embedding is fixed?" For instance, in the case of 2-connectivity augmentation of plane graphs, there exist graphs where a fixed embedding requires k edges, whereas allowing changes to the embedding admits a solution of size $\frac{k}{2}$. Our results show that this gap is significantly smaller for outerplanar graphs: although the optimal augmentation without embedding constraints uses $\frac{k}{2}$ edges, in the fixed-embedding setting the optimum increases by at most one, i.e., it is bounded by $\frac{k}{2} + 1$.

1.1 Our Contributions

2-edge Connectivity Augmentation for Plane Graphs (PECA-Fix). We present the first polynomial-time algorithm to optimally augment a connected planar graph (with a fixed embedding) to eliminate all bridges. Our algorithm runs in $O(n \cdot (1 + \alpha(n)))$ time using $O(n)$ space, where $n = |V|$ and $\alpha(\cdot)$ is the inverse Ackermann function (arising from union–find operations). We build upon the framework of [9], meant for vertex-connectivity, to handle our edge-connectivity setting.

Triconnectivity Augmentation for Outerplane Graphs (OTA-Fix). Our main result is an $O(n \cdot (1 + \alpha(n)))$ time exact algorithm for augmenting a biconnected outerplanar graph (with fixed embedding) to achieve 3-vertex-connectivity. In a setting where problems are NP-hard for connectivity greater than 2, we prove that Outerplane graphs form a sufficiently rich restriction admitting a fast polynomial time algorithm. In addition, We show that despite the embedding constraint, the optimal solution size increases by at most one in comparison with the analogous problem without a fixed embedding.

A central idea in our algorithm for OTA-fix is a face-splitting technique that could possibly be useful in other contexts as well. We identify a "large" face in the embedding that hinders connectivity, and we add internal edges to split this face into smaller ones, which simplifies the augmentation task. Moreover, for the outerplanar augmentation problem, we prove a noteworthy structural property of optimal solutions: at most two new edges need to be added inside the outerplanar graph in any optimum augmentation, no matter how large the graph is. This property significantly restricts the search space of possible solutions and underpins the correctness and optimality of our algorithm.

1.2 Related Work

Biconnectivity augmentation is the special case of k-connectivity augmentation for $k = 2$. Polynomial-time algorithms are known for $k = 3$ [12, 22] and $k = 4$ [13], while the problem remained widely open for $k \geq 5$. This was resolved when, for a fixed k, Jackson et al. [14] gave a polynomial-time algorithm, and Végh [20] showed that a $(k-1)$-connected graph can be augmented to k-connectivity in polynomial time. The analogous edge version includes bridge-connectivity augmentation ($k = 2$), and general k-edge-connectivity augmentation for which Watanabe [21] provided a polynomial-time algorithm.

For planar biconnectivity augmentation (PVCA), Kant and Bodlaender [16] gave a 2-approximation and claimed a 3/2-approximation; counterexamples were later shown by Fialko and Mutzel [6], who proposed a 5/3-approximation, which was also refuted by

Gutwenger, Mutzel and Zey [8]. Thus, the best known ratio remains 2. Geometric variants of biconnectivity augmentation have also been studied [3, 2, 19]. Other works on planar graphs include augmentation problems where the output is required to be regular (or cubic) [10, 11].

Organization of the paper. In Section 3, we give a polynomial-time algorithm for PECA-Fix in connected planar graphs with a fixed embedding. In Section 4, we present the main contribution of the paper: the OTA-Fix algorithm and its analysis. We first provide an algorithm that uses slightly more than the optimal number of augmentation edges in Section 4.3. Then, in Section 4.4 we use this to provide an exact polynomial time algorithm. Due to space limits, some algorithms, lemmas and proofs are omitted. Refer to the full version of the paper for complete details.

2 Preliminaries

For a graph G, $V(G)$ and $E(G)$ denote its vertex and edge sets. We use standard graph-theoretic terminology as in [4] for connectivity, cut-edges (also called bridges), cut-vertices and planar graphs. In particular, a graph is k-vertex-connected if it has more than three vertices and remains connected after deleting any set of at most $k-1$ vertices. Similarly, a graph is k-edge-connected if it is connected and remains so even when any subset of edges of up to $k-1$ size is deleted. A *plane graph* is a planar graph equipped with a fixed embedding in the plane.

A subset $S \subseteq V(G)$ is a *block* (or *biconnected component*) if $G[S]$ is biconnected and maximal with this property. A *bridge-component* is a maximal vertex set S such that $G[S]$ is 2-edge connected.

The *block-cut tree* (*bc*-tree) of a connected graph G, denoted $bc(G)$, is the graph whose nodes are the blocks (*b-nodes*) and cut vertices (*c-nodes*) of G, with a block node adjacent to a cut-vertex node iff the cut vertex lies in the block.

Similarly, the *bridgetree* $bt(G)$ has nodes corresponding to bridge-components (b-nodes) and cut-edges (c-nodes), where a bridge-component node is adjacent to a cut-edge node iff an endpoint of the cut-edge lies in the component. Every c-node in $bt(G)$ has degree 2. The blocks corresponding to the leaf b-nodes in $bc(G)$ are called *pendant blocks*. Pendant bridge-components are defined analogously in $bt(G)$.

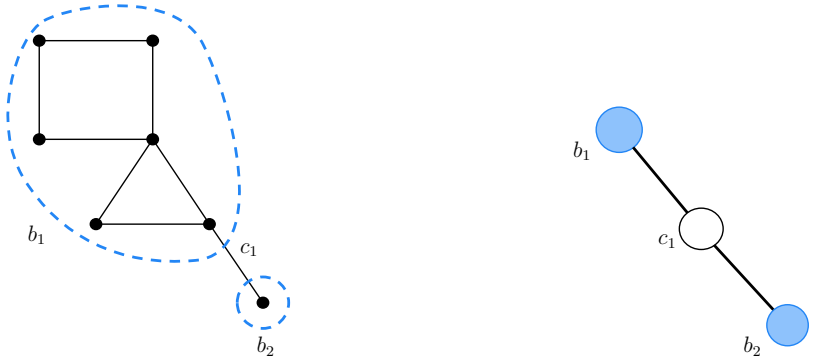
▶ **Definition 1** (Massive node & Balanced BC-tree). *Given a connected graph G, let p be the number of leaves (pendants) in $bc(G)$. A c-node c^* in $bc(G)$, is called* massive *if* $\deg_{bc(G)}(c^*) \geq \lceil \frac{p}{2} \rceil + 2$. *The bc-tree of G is called* balanced *if it does not contain any massive c-node. Otherwise, $bc(G)$ is called* unbalanced.

A bridge-component of G is called *pendant* if the node corresponding to it in the bridge-tree of G is a leaf.

Recall that G has a fixed planar embedding. Two bridge-components B_1, B_2 are *matchable* if they contain vertices on a common face, so that adding an edge between them preserves the embedding. The corresponding b-nodes in $bt(G)$ are also called matchable.

▶ **Observation 2.** *Let u, v be matchable pendant b-nodes in $bt(G)$ and let $G' = G + e$ where e joins vertices in components of u and v. Then $bt(G')$ is obtained from $bt(G)$ by contracting the u–v path in $bt(G)$ into a single b-node b', preserving all adjacencies outside the path.*

▶ **Observation 3** (Lower bound). *If p is the number of pendants in $bt(G)$, at least $\lceil p/2 \rceil$ edges are required to make G 2-edge-connected.*



(a) **(b)**

Figure 1 An example of representing a *bt*-tree. (a) Original graph with a fixed embedding; (b) The bridge tree of the graph.

For any two pendants p_1 and p_2, adding the edge between them reduces the number of pendants remaining at least by 1. We shall now define a condition for bridgetrees analogous to the leaf connectivity condition in [9].

▶ **Definition 4** (Leaf connection condition (l.c.c)). *Two pendants p_1, p_2 in $bt(G)$ satisfy the l.c.c. if the path between them contains either two b-nodes of degree at least three or one b-node of degree at least four.*

▶ **Lemma 5.** *If p_1, p_2 satisfy the l.c.c., adding an edge between them reduces the number of pendants by two.*

Bundles and *labels* were defined with the context of *bc*-tree in [6]. We shall do it for our requirement in the *bt*-tree now.

Bundles and labels. Let P be the set of pendants of $bt(G)$. A set $B \subseteq P$ is a *bundle* if (i) every pair in B is matchable, (ii) joining any pair produces a new pendant in $bt(G+e)$, and (iii) B is maximal. By Observation 2, two pendants lie in the same bundle iff the path between them in $bt(G)$ contains exactly one degree-3 b-node, called the *parent* of the bundle. A bundle together with its parent is a *label*; its size $L(\ell)$ is the number of pendants the bundle contains.

The notion of bundles and labels is useful because of the following. Inside a bundle, any two pendants are mutually matchable and behave the same: connecting any pair does not eliminate the entire group, but produces a new pendant through their common parent. Thus, the bundle acts as a single unit attached to the rest of the bridgetree at its parent. Labels therefore partition the pendants into blocks that cannot be fully resolved internally; useful augmentation edges typically connect pendants from *different* labels, which ensures progress (via the l.c.c.) in reducing the number of pendants.

▶ **Lemma 6.** *If two matchable pendants belong to different labels, then they satisfy the l.c.c.*

3 Bridgeconnectivity augmentation for a fixed embedding

In this section, we give an algorithm that solves PECA-Fix. Recall that, in PECA-Fix, the input is a connected plane graph G and the goal is to find a subset of edges $E' \subseteq V(G) \times V(G)$, of minimum size such that $G' = G + E'$ is bridgeconnected and planar. Let us restate our result for the sake of convenience.

Algorithm 1 PA_Bridgeconn.

Input: A connected plane graph G
Output: A set of vertex pairs E' such that $G + E'$ is 2-edge connected
$E' \leftarrow \emptyset$;
for *each face* F **do**
  $A \leftarrow$ Face_Conn(F);
  $E' \leftarrow E' \cup A$;
return E';

▶ **Theorem 7.** *There exists an algorithm that for a given graph G solves PECA-Fix in $\mathcal{O}(|V(G)|(1+\alpha(|V(G)|)))$ time and requires $\mathcal{O}(|V(G)|)$ space.*

3.1 Preliminaries and Structural Remarks

For a plane graph G which is a tree, we can provide a *cyclic ordering* of the leaves in this tree from its embedding. We can start from some leaf node and move along the outer face (the only face for a tree) in clockwise direction and add the leaves whenever they are visited. This provides a cyclic ordering, π, of the leaves in this tree. Two pendants are *adjacent* if they appear successively in the cyclic ordering. Two labels are said to be *neighbours* if a pendant of one is adjacent to a pendant of another. Each label has an adjacent label.

For a face F, consider the subgraph of G that consists of vertices and edges that lie on F, call it facegraph(F) (for brevity, let $H :=$ facegraph(F)).

Since augmentation edges lie inside faces, the problem reduces to making each facegraph bridgeconnected. We therefore operate on the bridgetree of each facegraph. The embedding of $bt(H)$ is inherited from the embedding of H by preserving the cyclic order of incident cut edges at each block.

3.2 Algorithm for PECA-Fix

The algorithm works by computing the optimal edges to eliminate all the bridges in each face F. For each face F, Algorithm 2 computes H, and iteratively connects two "pendant" components to reduce their number (hinging on the approach of [9] for biconnectivity).

Specifically, the following is done in H: we pick a maximum sized label and connect one of its pendant to an adjacent pendant from a different label. The algorithm keeps repeating this process over and over again until there is only one label, and then we handle one of two possible cases: the remaining label has size 3, or the remaining label has size 2. In each of the cases we find the optimal solution. Refer Figure 2 for an illustrative example of Face_Conn.

▶ **Lemma 8.** PA_Bridgeconn *outputs a set of edges E' such that $G+E'$ is planar, extends the planar embedding of G, and is bridgeconnected.*

What is left is to prove that within each face, our algorithm finds the optimal number of edges to add. We use this to conclude that the added edges in total comprise the optimal solution.

▶ **Lemma 9.** *For each face F such that $bt($Facegraph$(F))$ has p pendants, Face_Conn uses $\lceil \frac{p}{2} \rceil$ (optimal) edges to 2-edge connect Facegraph(F).*

Algorithm 2 FACE_CONN.

Input: A face F of a graph with a fixed embedding
Output: List of edges A that 2-edge-connect the face F
$H \leftarrow$ FACEGRAPH(F);
Compute $bt(H)$;
$A \leftarrow \emptyset$;
while *number of labels* > 1 **do**
  Consider $bt(F)$ and take any label l_1, and another neighbouring label l_2;
  Let p_1 from l_1 and p_2 from l_2 represent pendant blocks adjacent to each other in the cyclic ordering of pendants;
  Add an edge e in H connecting simple vertices of the blocks p_1 and p_2;
  Update $H \leftarrow H + e$;
  $A \leftarrow A \cup \{e\}$;
if *there is exactly one label* l *in* H **then**
  if size(l) $= 2$ **then**
    Let e be an edge connecting a simple vertex from each pendant block;
    $A \leftarrow A \cup \{e\}$;
  else
    if size(l) $= 3$ **then**
      Add 2 edges e_1 and e_2 between pendants p_1, p_2 and p_2, p_3;
      $A \leftarrow A \cup \{e_1, e_2\}$;
return A;

Proof. Consider a face F, and let p denote the number of pendants.

If $p = 2$, Algorithm FACE_CONN connects the two pendants by adding one edge, regardless of whether their parent in the bt-tree is a b-node or a c-node. Thus, this case is resolved.

If $p = 3$, there must be a b-node of degree 3 in the bt-tree (since every c-node has degree 2). This is the only possible node of degree 3. By the contrapositive of Lemma 6, all three pendants have the same label. The algorithm therefore adds two edges to augment the graph.

Inductive hypothesis. For any bridgetree with $p' < p$ pendants, the face can be made bridgeconnected using $\lceil \frac{p'}{2} \rceil$ edges.

Now assume $p > 3$. Then there are at least two distinct labels (each b-label has size at most 3). The algorithm adds an edge between pendants of two different labels. By Lemma 6 and Lemma 5, this reduces the number of pendants by 2.

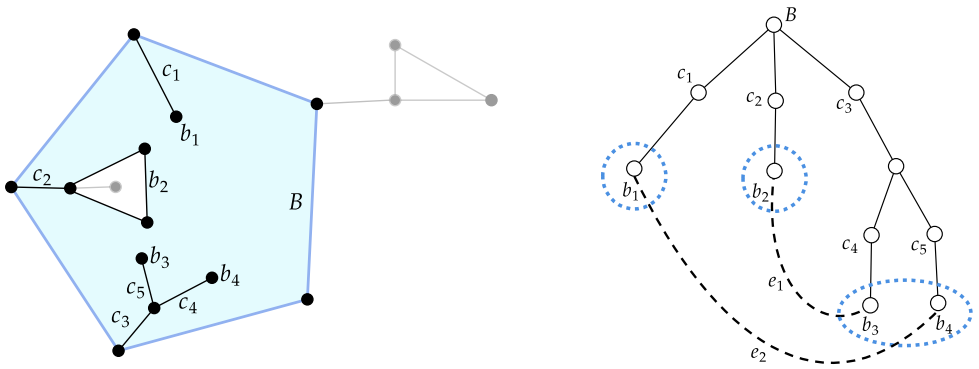
By the inductive hypothesis, $\lceil \frac{p'}{2} \rceil = \lceil \frac{p-2}{2} \rceil$ additional edges suffice to bridgeconnect the resulting graph $F + e$. Hence, the total number of edges used is

$$\left\lceil \frac{p-2}{2} \right\rceil + 1 = \left\lceil \frac{p}{2} \right\rceil,$$

which completes the proof. ◀

Combining the arguments in Lemmas 8 and 9 and arguing for the running time, we get the final theorem.

▶ **Theorem 7.** *There exists an algorithm that for a given graph G solves PECA-Fix in $\mathcal{O}(|V(G)|(1 + \alpha(|V(G)|)))$ time and requires $\mathcal{O}(|V(G)|)$ space.*



(a) **(b)**

Figure 2 (a) A connected graph with a particular highlighted face containing cut edges (denoted by c_i's) and bridge-components (denoted by b_i's); (b) The bridge-tree decomposition of the FACEGRAPH, with dotted light-blue edges denoting the labels. The dotted black edges e_1 and e_2 show the augmentation edges between two neighbouring labels, in the first and second round of FACE_CONN respectively.

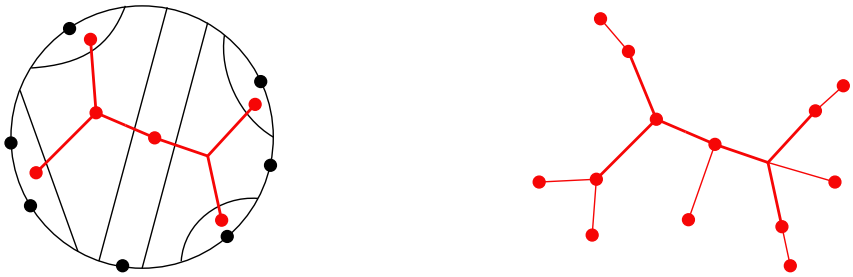
4 Triconnectivity Augmentation of Outerplanar Graphs with a fixed Embedding

In this section, we turn to the main contribution of the paper and move to the domain of Vertex Connectivity Augmentation. Recall that in OTA-Fix, the input is a biconnected Outerplane graph G. We are required to find a minimum-size set of edges E' such that $G' = G + E'$ has a planar embedding that extends the original embedding.

Section and method overview. Our strategy for solving OTA-Fix is split across various subsections. In Sections 4.1 and 4.2 we recall the concept of *treegraph* introduced by Kant [15] for outerplanar graphs and an algorithm that given a treegraph outputs a minimum set of edges that makes the treegraph biconnected such that the endpoint of every augmentation edge is a leaf vertex. Furthermore, we discuss the equivalences between the solution edges of the treegraph and solution edges of a biconnected outerplane graph. Finally we discuss our novel contribution where we find an augmentation that is at most one edge away from optimal (Section 4.3), and then adjust it to an optimal solution in Section 4.4. The high-level idea is to handle separately the case of an "unbalanced" face with odd number of degree-2 vertices by adding one or two interior edges inside a face of the graph.

4.1 Preliminaries

When we consider the input graph to be embedded on a plane, all the faces are *internal* except for one face which is *external* and unbounded. In similar vein, the edges that we add to augment the graph (edges in the solution) are of two kinds. Either they lie in an internal face, or they lie in the outer face. In case of the latter, it is called an *outer edge* otherwise, it is called an *inner edge*. Note that the distinction is made before any edges are added to the graph. For a given pair of non-adjacent vertices u, v in an internal face of the graph, the augmented edge uv can be an inner edge or an outer edge.



(a) (b)

Figure 3 (a) The graph G (black) and its dual graph (red), excluding the outer face. (b) The interior dual graph with a pendant attached for each degree-2 vertex, inheriting the canonical embedding from the embedding of G.

Before we proceed further, let us observe a lower bound on the size of a solution for OTA-Fix.

▶ **Observation 10.** *For a given 2-connected outerplane graph G with k degree 2 vertices in the graph, we require at least $\lceil \frac{k}{2} \rceil$ edges to 3-vertex connect the graph.*

Let us now look at a *Tree* structure of the graph that was originally described in [15].

▶ **Definition 11.** *$T(G)$ or* Tree of the graph *for an outerplanar plane graph G is the interior dual of G with one pendant attached for each degree-2 vertex in G, embedded according to the cyclic order inherited from G.*

Note that the interior dual for a biconnected outerplane graph is a tree. In the following, there is a change in notation; in the previous sections, we used p to denote the number of pendants. Let k denote the number of pendants for $T(G)$.

As an extension from previously used terminology, we define balanced graphs as follows based on balanced *bc*-trees

▶ **Definition 12.** *An outerplane 2-connected graph G is said to be balanced if $bc(T(G))$ is balanced.*

Figure 3 provides an example of an outerplane graph and its corresponding treegraph.

4.2 Plane Biconnectivity Augmentation for a Balanced Tree

In this subsection, we discuss an algorithm that outputs given a balanced tree, a minimum set of edges that makes the graph 2-vertex-connected ensuring that the output satisfies the additional properties stated in Lemma 16. Note that the reason we discuss the augmentation of a plane tree is revealed when it helps us with the augmentation of an outerplane biconnected graph as used in Section 4.3 and Section 4.4.

We now describe the working of the Algorithm TreeConn. Let G be a balanced plane tree. Compute its block–cut tree $bc(G)$ and the associated labels, and initialize $A := \emptyset$. While more than one label remains, select a label l_1 of maximum size and a neighboring label l_2 in the cyclic order induced by the embedding. Let $p_1 \in l_1$ and $p_2 \in l_2$ be pendant blocks that are consecutive in this cyclic order. Insert an edge e between simple vertices of p_1 and p_2, update $G \leftarrow G + e$, and add e to A.

When exactly one label l remains, if $|l| = 2$ then insert a single edge between simple vertices of the two pendant blocks. If $|l| = 3$ with pendant blocks p_1, p_2, p_3, choose simple vertices $v_i \in p_i$ that are leaves of the original tree G, and add two edges $e_1 = v_1v_2$ and

$e_2 = v_2v_3$ such that the endpoints of each edge do not share a common parent in G (i.e., neither $\{v_1, v_2\}$ nor $\{v_2, v_3\}$ are siblings). The algorithm outputs the set A of all inserted edges.

We recall a lemma that provides a sufficient condition when a c-node of high degree should be a parent of a label.

▶ **Lemma 13** ([9, Lemma 2]). *If a bc-tree $\mathcal{B}$ is not balanced, its massive c-node c^* must be the parent of a label, say ℓ_0. Furthermore, ℓ_0 is the unique maximum label in $\mathcal{B}$.*

With this, we can proceed to prove the optimality of the algorithm with extra conditions.

▶ **Lemma 14.** *Given a balanced plane tree G with p leaves, TreeConn computes an optimal augmentation of size $\lceil p/2 \rceil$ that makes G biconnected. Furthermore, every augmentation edge is incident only on leaves of G, and no augmentation edge joins two leaves with a common parent.*

The arguments for the time and space finalises the analysis.

▶ **Lemma 15.** *TreeConn can be implemented to run in $\mathcal{O}(1 + \alpha(|V(G)|)|V(G)|)$ time and linear space.*

TreeConn is an augmentation procedure that provides a minimum set of edges satisfying some special properties. This is because we would eventually like the set of edges output by TreeConn to be used for triconnectivity augmentation of biconnected outerplane graphs. The specific properties are as stated in the lemma below:

▶ **Lemma 16.** *For a balanced plane tree T with k leaves (that is not K_2 or K_3), we can compute in time $O(1 + \alpha(|V|)\,|V|)$ and linear space an optimal augmentation A_T that biconnects T with the following properties:*

- $|A_T| = \lceil \frac{k}{2} \rceil$.
- *Every edge in A_T is incident to a leaf of T.*
- *No edge in A_T joins two leaves of T that share the same parent.*

Proof. This is a direct consequence of the Lemmas 14 and 15. ◀

Now we make the transition from the biconnectivity of a tree to the triconnectivity of an outerplane graph. For this reason, we need to show that the edges obtained while making $T(G)$ biconnected can be translated and used as the edges required to make G triconnected. For the case when G is balanced, the transition is direct.

▶ **Lemma 17.** *For a given balanced outerplane graph G, we can compute A_{tri} in time $O(1 + \alpha(|V|)\,|V|)$ and linear space such that $G + A_{tri}$ is triconnected and $|A_{tri}| = \lceil \frac{k}{2} \rceil$.*

The implications of biconnecting the plane tree as shown has further consequences which shall be highlighted in Section 4.4. We shall now proceed to the algorithmic portion for solving OTA-Fix.

4.3 An Additive (+1)-Approximation Algorithm for Triconnectivity Augmentation of Outerplane Graphs

In this subsection, we present an algorithm for augmenting biconnected outerplane graphs that is almost optimal i.e, it might be off by an additional term of 1. The result is stated in the following lemma:

▶ **Lemma 18.** *TriConn triconnects an input biconnected outerplane graph $G = (V, E)$ with at most $\lceil \frac{k}{2} \rceil + 1$ edges (where k represents the number of degree-2 vertices in V) and runs in time $\mathcal{O}(|V|(1 + \alpha(|V|)))$*

We now briefly describe the structure of the algorithm used in this subsection. The algorithm TriConn computes an augmentation that is at most one edge larger than optimal. For balanced instances, it invokes OuterPlanarConnect, which is optimal. For unbalanced instances, it first adds a carefully chosen edge inside a maximum-degree face so as to balance the graph, and then applies OuterPlanarConnect to the resulting instance.

Before we proceed further, we provide essential preliminaries and structural insights.

Preliminaries and Structural Results

Throughout the remainder of the section, we talk about internal faces in the graph and make a distinction between faces that are large and otherwise. Hence, it is essential and natural to introduce some new terminology.

▶ **Definition 19.** *For an Outerplane graph G with an internal face F, the* degree of a face F *or $d(F)$ refers to the degree of the vertex v_F representing the face in $T(G)$.*

A face F is said to be *massive* if in $T(G)$ we have $d(v_F) - 1 > \lceil \frac{k}{2} \rceil$.

We observe that if a face has very high degree, it is not possible for the other faces in the graph to have a high degree.

▶ **Lemma 20.** *Let G be an outerplane graph with k degree-2 vertices. If G contains a face F that is massive, then every other face F' satisfies $d(F') \leq \lceil \frac{k}{2} \rceil$.*

Now, we look at a structural lemma that guarantees a scenario when an edge added inside a face F balances the graph.

▶ **Lemma 21.** *If G is an unbalanced graph with a massive face F such that adding e inside F leads to an old face F' in $G(\neq F)$ becoming the maximum degree face in $G+e$, then $G+e$ is a balanced graph.*

Algorithm for (+1)-approximation

The algorithm proceeds as follows: for unbalanced graphs, we cleverly pick an edge in the maximum-degree face that splits the face into two new faces of similar size and makes use of the subroutine for balanced graphs highlighted in Lemma 17.

Algorithm 3 TriConn.

Input: Outerplane 2-connected graph G
Output: List of edges E' such that $G(V, E \cup E')$ is planar and triconnected
Compute $T(G)$, $k \leftarrow$ no. of leaves, and $d \leftarrow \max_{v \in T(G)} d(v)$;
if $\lceil \frac{k}{2} \rceil \geq d - 1$ **then**
  return *OuterPlanarConnect(G)* ; // Lemma 22
else
  Let v_F be the vertex corresponding to face F with maximum-degree d;
  Let u be an arbitrary vertex in F (choose it to be a degree 2 vertex if it exists);
  Let v be another vertex in F such that $e = uv$ splits F into two faces F_1 and F_2 with $|d(v_{F_1}) - d(v_{F_2})| \leq 1$;
  return OuterPlanarConnect$(G + e) \cup \{e\}$;

Algorithm 4 OuterPlanarConnect.

Input: Outerplane 2-connected balanced graph G
Output: List of edges E' such that $G(V, E \cup E')$ is planar and triconnected
$A \leftarrow$ TreeConn$(T(G))$;
Let A' represent the corresponding edge set in G from Lemma 17;
return A';

Although this solution does not claim to be the optimal one, it is indeed optimal when the input graph is balanced.

▶ **Lemma 22.** *For a balanced outerplane 2-connected graph G, OuterPlanarConnect computes an optimal triconnectivity augmentation in $\mathcal{O}(|V|(1+\alpha(|V|)))$ time.*

Proof. If G is balanced, Lemma 17 implies that $\lceil \frac{k}{2} \rceil$ edges suffice and are necessary. The correctness and running time follow as well. ◀

We can finally prove the correctness of the solution with the desired Time Complexity.

Although TreeConn might use one edge more than the optimal, in multiple instances, it does provide the optimal solution. In fact, there are various instances where $\lceil \frac{k}{2} \rceil + 1$ edges are required to make a graph triconnected.

4.4 An Optimal Algorithm for Triconnectivity Augmentation of Outerplane Graphs

In this subsection, we finally provide an optimal algorithm for the triconnectivity augmentation problem on general biconnected outerplane graphs (OTA-Fix) building on Section 4.3 and Section 4.2. Let us restate it for convenience.

▶ **Theorem 23.** *There exists an algorithm that solves OTA-Fix by running in time $\mathcal{O}(|V|(1+\alpha(|V|)))$ time and linear space, augmenting $\mathrm{OPT}(G) \in \{\lceil k/2 \rceil, \lceil k/2 \rceil + 1\}$ edges, where α denotes the inverse of the Ackermann function.*

Proof strategy. The additive-1 algorithm of Section 4.3 already shows that every instance can be augmented using at most $\lceil k/2 \rceil + 1$ edges. To obtain an exact algorithm, we first establish structural properties of optimal solutions. In particular, we show that unbalanced instances require internal augmentation edges when only $\lceil k/2 \rceil$ edges are used. Morever, we establish that all the internal edges can be added to one face (Lemma 27). These structural results allow us to distinguish instances whose optimum is $\lceil k/2 \rceil + 1$ from those admitting a solution of size $\lceil k/2 \rceil$. The resulting algorithm either identifies a balancing edge in the maximum-degree face or reduces the problem to a small configuration, which is the final balancing case we deal with.

Before we proceed to the structural preliminaries required for this section, we record two consequences from our results in Section 4.2.

Supporting lemmas for Section 4.4

We observed that TreeConn provides a minimum edge set that helps us move from biconnectivity of a plane balanced tree to the triconnectivity of an outerplane balanced graph.

The following lemma shows that whenever an optimal augmentation using only outer edges exists for OTA-Fix, we may assume, without loss of generality, that all augmentation edges are incident only on degree-2 vertices.

▶ **Lemma 24.** *If an outerplane biconnected graph G can be triconnected using $\lceil \frac{k}{2} \rceil$ outer edges A_{opt} (and no internal edges), then there exists A_{tri} such that A_{tri} only connects degree-2 vertices in G and $|A_{tri}| = \lceil \frac{k}{2} \rceil$ such that $G + A_{tri}$ is triconnected.*

The previous lemma shows that, whenever an optimal augmentation using only outer edges exists, we may assume that the augmentation edges are incident only on degree-2 vertices. We now derive an important structural consequence of this restriction. In particular, we show that the existence of such an augmentation forces the graph to be balanced.

▶ **Lemma 25.** *If an outerplane biconnected graph G can be triconnected using $\lceil \frac{k}{2} \rceil$ outer edges (and no inner edges), then G is a balanced graph.*

By taking the contrapositive of Lemma 25 we get the following.

▶ **Corollary 26.** *If G is an unbalanced outerplane graph, then $> \lceil \frac{k}{2} \rceil$ outer edges are required to make G 3-connected.*

Unbalanced outerplane graphs requires at least 1 internal edge in any triconnectivity augmentation using only $\lceil \frac{k}{2} \rceil$ edges. In the upcoming section, we introduce how a minimum edge addition can make an unbalanced graph into a balanced one.

Structural Preliminaries and Auxiliary Results

We introduce terminology and definitions that shall be used throughout the remainder of the section. For vertices u, v on the boundary of F, let P_{uv} denote the clockwise boundary path from u to v (and P_{vu} the complementary path). Inserting the edge uv splits F into faces F_1 and F_2 bounded by $P_{uv} \cup uv$ and $P_{vu} \cup uv$, respectively. Define $t(u, v) := d(F_1) - 1$ and $t(v, u) := d(F_2) - 1$.

Two degree-2 vertices u and v on the boundary of a face F are said to be *consecutive* along P_{uv} if all the interior vertices in P_{uv} have degree more than 2. We move to important structural insights.

▶ **Lemma 27.** *There exists an optimal solution for OTA-Fix where all the internal edges are added inside the maximum-degree face.*

We can now show that for the case where the input graph G has an odd number of degree-2 vertices, the resulting augmentation is already optimal.

▶ **Lemma 28.** *If k is odd, then TriConn returns an optimal solution to OTA-Fix.*

Case when the optimum is $\left\lceil \frac{k}{2} \right\rceil + 1$

Going forward, we can assume that the input graph G has even number of degree-2 vertices, since otherwise Lemma 28 is applicable. Below, we discuss a structural case when at least $\lceil \frac{k}{2} \rceil + 1$ edges are required in an optimal solution.

▶ **Proposition 29.** *Let G be an instance of OTA-Fix and let F be a face containing two consecutive degree-2 vertices v_1 and v_2 (i.e., no other degree-2 vertex lies on the boundary of F between them). If $t(v_1, v_2) \geq \lceil \frac{k}{2} \rceil$ and k is even, then at least $\lceil \frac{k}{2} \rceil + 1$ edges are required to triconnect G.*

Algorithm 5 TriConnExact.

Input: Outerplane 2-connected graph G
Output: List of edges E' such that $G(V, E \cup E')$ is planar and triconnected
Compute $T(G)$, $k \leftarrow$ no. of leaves, and $d \leftarrow \max_{v \in T(G)} \deg(v)$;
if *$T(G)$ is balanced or k is odd* **then**
  return *TriConn(G)* ; `// Lemma 22 and Lemma 28`
Let v_F be the vertex corresponding to face F with maximum-degree d in G;
if *F has ≤ 4 degree 2 vertices on it* **then**
  Bruteforce internal (vertex-disjoint) edge addition in F with a budget of at most 2 non-crossing edges between non-adjacent degree 2 vertices;
  if *there exists a set of edges M such that $G + M$ is balanced* **then**
    return $M \cup$ *OuterPlanarConnect*$(G + M)$;
  return *TriConn(G)*; `// Lemma 30`
Let $v_1, \ldots, v_r$ (with $v_{i \bmod r}$) be the degree 2 vertices in F in consecutive cyclic order;
for $i \in [r]$ **do**
  if $t(v_i, v_{i+1}) \geq \lceil \frac{k}{2} \rceil$ **then**
    return *TriConn(G)* ; `// Proposition 29`
  else if $t(v_i, v_{i+1}) = \lceil \frac{k}{2} \rceil - 1$ **then**
    return $\{v_i v_{i+1}\} \cup$ *TriConn*$(G + v_i v_{i+1})$; `// Observation 31`
Let u be a degree 2 vertex of F;
if *there exists a degree 2 vertex v in F such that $|t(u, v) - t(v, u)| \leq 1$* **then**
  return $\{uv\} \cup$ *OuterPlanarConnect*$(G + uv)$;
else
  return *FiveVertices(G)* ; `// finds one or two internal edges to add to balance the face F`

Easy Optimal cases

In some configurations, it is possible to save case analysis. We shall now focus on the case where it is easy to compute the optimal solution.

▶ **Lemma 30** (Small-face case). *Let F be a maximum-degree face containing at most four degree-2 vertices. Then* TriConnExact *returns an optimal augmentation for OTA-Fix.*

Proof. Algorithm 5 enumerates all possibilities of inserting one or two internal edges between degree-2 vertices in F.

Suppose the optimal solution has size $\lceil \frac{k}{2} \rceil$. Since k is even, any optimal solution connects only degree-2 vertices. By Lemma 27, there exists an optimal solution whose internal edges all lie inside F. Hence, the brute-force step exhausts all possible internal-edge configurations of an optimal $\lceil \frac{k}{2} \rceil$-edge solution, and the algorithm finds such a solution.

Otherwise, every set of internal edges leaves the graph unbalanced. By Corollary 26, any triconnectivity augmentation then requires at least $\lceil \frac{k}{2} \rceil + 1$ edges. In this case, TriConn produces an optimal solution of size $\lceil \frac{k}{2} \rceil + 1$ (Lemma 18).

Thus, in both cases the algorithm outputs an optimal solution. ◀

The remaining easy case where $t(v_i, v_{i+1}) = \lceil \frac{k}{2} \rceil - 1$ for two degree-2 vertices v_i and v_{i+1} occurs as a continuation of analysis from Proposition 29. Here, one internal edge suffices to balance the graph.

▶ **Observation 31.** *If* $t(v_i, v_{i+1}) = \lceil \frac{k}{2} \rceil - 1$ *for two degree-2 vertices* v_i *and* v_{i+1}*, then* $G + v_i v_{i+1}$ *is balanced and* TriConn *gives an optimal solution.*

The detailed procedure is given in Algorithm 5.

The Final Balancing Case

The cases that we have discussed until now give us the following entry condition on the maximum-degree face F: $t(v_i, v_{i+1}) \leq \lceil k/2 \rceil - 2$ for all i, and there are at least five degree-2 vertices on the boundary of F. Under this condition, we show that it is possible to pick five degree-2 vertices from the boundary of F such that they represent the face, i.e., every internal augmentation edge is incident on one of these vertices. Once these vertices are chosen, we carefully analyze the sizes of the facial segments (captured by the values $t(v_i, v_{i+1})$) and accordingly add one or two edges inside F. We then show that the resulting graph is balanced, after which the resulting graph can be made triconnected using OuterPlanarConnect. $\mathrm{OPT}(G) = \lceil k/2 \rceil$ edges. The detailed balancing procedure and its correctness proof are deferred to the full version.

5 Conclusion

We studied both edge- and vertex-connectivity augmentation under fixed planar embeddings. For 2-edge-connectivity augmentation of plane graphs (PECA-Fix), we re-introduced and used the *bt*-tree structure and a labeling scheme that identifies safe edges, yielding a near-linear-time optimal algorithm. For 3-vertex-connectivity augmentation of biconnected outerplane graphs (OTA-Fix), we used the dual tree of inner faces and related biconnecting this tree to triconnecting the original graph. Furthermore, we developed a face-splitting technique to obtain a (+1)-approximate solution and got rid of the (+1) factor with extensive case analysis and new combinatorial insights and thus obtained a near-linear-time solution.

Several questions remain open. Is it possible to solve PECA-Fix with a better running time? Can a 2-edge-connected outerplane graph be augmented to 3-edge-connectivity in near-linear or better time? Our techniques do not extend directly, but may offer useful insights. It would be interesting to extend the problem to a weighted setting where the solution edges incur a cost.

References

1 Hugo A. Akitaya, Justin Dallant, Erik D. Demaine, Michael Kaufmann, Linda Kleist, Frederick Stock, Csaba D. Tóth, and Torsten Ueckerdt. The Price of Connectivity Augmentation on Planar Graphs. In Vida Dujmović and Fabrizio Montecchiani, editors, *33rd International Symposium on Graph Drawing and Network Visualization (GD 2025)*, volume 357 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:24, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.GD.2025.23.

2 M. Abellanas, A. García, F. Hurtado, J. Tejel, and J. Urrutia. Augmenting the connectivity of geometric graphs. *Computational Geometry*, 40(3):220–230, 2008. doi:10.1016/j.comgeo.2007.09.001.

3 Marwan Al-Jubeh, Mashhood Ishaque, Kristóf Rédei, Diane L. Souvaine, and Csaba D. Tóth. Tri-edge-connectivity augmentation for planar straight line graphs. In *Proceedings of the 20th International Symposium on Algorithms and Computation*, ISAAC '09, pages 902–912, Berlin, Heidelberg, 2009. Springer-Verlag. doi:10.1007/978-3-642-10631-6_91.

4 Reinhard Diestel. *Graph Theory*. Springer Publishing Company, Incorporated, 5th edition, 2017.

5 Kapali P. Eswaran and R. Endre Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5(4):653–665, 1976. doi:10.1137/0205044.

6 Sergej Fialko and Petra Mutzel. A new approximation algorithm for the planar augmentation problem. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 260–269, USA, 1998. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=314613.314714.

7 A. García, F. Hurtado, M. Noy, and J. Tejel. Augmenting the Connectivity of Outerplanar Graphs. *Algorithmica*, 56(2):160–179, February 2010. doi:10.1007/s00453-008-9167-1.

8 Carsten Gutwenger, Petra Mutzel, and Bernd Zey. On the hardness and approximability of planar biconnectivity augmentation. In Hung Q. Ngo, editor, *Computing and Combinatorics*, pages 249–257, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-02882-3_25.

9 Carsten Gutwenger, Petra Mutzel, and Bernd Zey. Planar biconnectivity augmentation with fixed embedding. In Jiří Fiala, Jan Kratochvíl, and Mirka Miller, editors, *Combinatorial Algorithms*, pages 289–300, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-10217-2_29.

10 Tanja Hartmann, Jonathan Rollin, and Ignaz Rutter. Cubic augmentation of planar graphs. *arXiv preprint arXiv:1209.3865*, 2012. arXiv:1209.3865.

11 Tanja Hartmann, Jonathan Rollin, and Ignaz Rutter. Regular augmentation of planar graphs. *Algorithmica*, 73(2):306–370, 2015. doi:10.1007/S00453-014-9922-4.

12 T.-S. Hsu and V. Ramachandran. A linear time algorithm for triconnectivity augmentation. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 548–559, 1991. doi:10.1109/SFCS.1991.185418.

13 Tsan-sheng Hsu. On four-connecting a triconnected graph. *J. Algorithms*, 35(2):202–234, May 2000. doi:10.1006/jagm.2000.1077.

14 Bill Jackson and Tibor Jordán. Independence free graphs and vertex connectivity augmentation. *Journal of Combinatorial Theory, Series B*, 94(1):31–77, 2005. doi:10.1016/j.jctb.2004.01.004.

15 Goos Kant. Augmenting outerplanar graphs. *J. Algorithms*, 21(1):1–25, July 1996. doi:10.1006/jagm.1996.0034.

16 Goos Kant and Hans L. Bodlaender. Planar graph augmentation problems. In Frank Dehne, Jörg-Rüdiger Sack, and Nicola Santoro, editors, *Algorithms and Data Structures*, pages 286–298, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

17 Hiroshi Nagamochi and Peter Eades. Edge-splitting and edge-connectivity augmentation in planar graphs. In Robert E. Bixby, E. Andrew Boyd, and Roger Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization*, pages 96–111, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. doi:10.1007/3-540-69346-7_8.

18 Ignaz Rutter and Alexander Wolff. Augmenting the connectivity of planar and geometric graphs. *Electronic Notes in Discrete Mathematics*, 31:53–56, 2008. The International Conference on Topological and Geometric Graph Theory. doi:10.1016/j.endm.2008.06.009.

19 Csaba D. Tóth. Connectivity augmentation in planar straight line graphs. *European Journal of Combinatorics*, 33(3):408–425, 2012. Topological and Geometric Graph Theory. doi:10.1016/j.ejc.2011.09.002.

20 László A. Végh. Augmenting undirected node-connectivity by one. *SIAM J. Discret. Math.*, 25(2):695–718, 2011. doi:10.1137/100787507.

21 Toshimasa Watanabe and Akira Nakamura. Edge-connectivity augmentation problems. *Journal of Computer and System Sciences*, 35(1):96–144, 1987. doi:10.1016/0022-0000(87)90038-9.

22 Toshimasa Watanabe and Akira Nakamura. A minimum 3-connectivity augmentation of a graph. *Journal of Computer and System Sciences*, 46(1):91–128, 1993. doi:10.1016/0022-0000(93)90050-7.