TR-94-003

On Four-Connecting a Triconnected Graph

Tsan-sheng Hsu

On Four-Connecting a Triconnected Graph[†]

Tsan-sheng Hsu

Institute of Information Science
Academia Sinica, Nankang
Taipei 11529, Taiwan, ROC
tshsu@iis.sinica.edu.tw

December 23, 1993

Abstract

We consider the problem of finding a smallest set of edges whose addition four-connects a triconnected graph. This is a fundamental graph-theoretic problem that has applications in designing reliable networks.

We present an $O(n \cdot \alpha(m,n)+m)$ time sequential algorithm for four-connecting an undirected graph G that is triconnected by adding the smallest number of edges, where n and m are the number of vertices and edges in G, respectively, and $\alpha(m,n)$ is the inverse Ackermann function. This is the first polynomial time algorithm to solve this problem exactly.

In deriving our algorithm, we present a new lower bound for the number of edges needed to four-connect a triconnected graph. The form of this lower bound is different from the form of the lower bound known for biconnectivity augmentation and triconnectivity augmentation. Our new lower bound applies for arbitrary k, and gives a tighter lower bound than the one known earlier for the number of edges needed to k-connect a $(k-1)$-connected graph. For $k = 4$, we show that this lower bound is tight by giving an efficient algorithm for finding a set of edges with the required size whose addition four-connects a triconnected graph.

[†]This work was supported in part by NSF Grant CCR-90-23059 and by an IBM graduate fellowship. An extended abstract of this work appeared in the Proceedings of 33rd Annual Symposium on Foundations of Computer Science, 1992, pp. 70–79. Part of this work was done when the author was at Department of Computer Sciences, University of Texas at Austin.

1 Introduction

The problem of augmenting a graph to reach a certain connectivity requirement by adding edges has important applications in network reliability [FC70, JG86, SWK69] and fault-tolerant computing. One version of the augmentation problem is to augment the input graph to reach a given connectivity requirement by adding a smallest set of edges. We refer to this problem as the *smallest augmentation* problem.

In this paper, we describe an almost linear time algorithm for optimally four-connecting a triconnected graph. This is the first polynomial time algorithm to solve this problem exactly. We first present a lower bound for the number of edges that must be added in order to reach four-connectivity. Note that lower bounds different from the one we give here are known for the number of edges needed to biconnect a connected graph [ET76] and to triconnect a biconnected graph [HR91]. It turns out that in both cases, we can always augment the graph using exactly the number of edges specified in this above lower bound [ET76, HR91]. However, an extension of this type of lower bound for four-connecting a triconnected graph does not always give us the exact number of edges needed [Jor92, KT91]. (For details and examples, see Section 4.) We present a new type of lower bound that equals the exact number of edges needed to four-connect a triconnected graph. By using our new lower bound, we derive an $O(n \cdot \alpha(m,n)+m)$ time sequential algorithm for finding a smallest set of edges whose addition four-connects a triconnected graph with n vertices and m edges, where $\alpha(m,n)$ is the inverse Ackermann function. Our new lower bound applies for arbitrary k, and gives a tighter lower bound than the one known earlier for the number of edges needed to k-connect a $(k-1)$-connected graph. The new lower bound and the algorithm described in this paper may lead to a better understanding of the problem of optimally k-connecting a $(k-1)$-connected graph, for an arbitrary k.

The algorithmic notation used is pseudo-Pascal and is similar to the notation of Tarjan [Tar83] and Ramachandran [Ram93]. We enclose comments between '{*' and '*}'. Parameters are called by value unless they are declared with the keyword **modifies** in which case they are called by value and result. Graphs used in this paper is undirected and triconnected unless specified explicitly otherwise.

The organization of this paper is as follows. Section 2 lists related work. Section 3 gives definitions used in this paper. Section 4 gives a lower bound of the number of edges needed to

four-connect a triconnected graph. Section 5 gives our algorithm for finding a smallest four-connectivity augmentation based on the lower bound shown in Section 4. Finally, Section 6 concludes this paper.

2 Related Work

2.1 Vertex-Connectivity Augmentations

The following results are known for solving the smallest augmentation problem on an undirected graph to satisfy a vertex-connectivity requirement.

Eswaran and Tarjan [ET76] (and Plesník [Ple76], independently) gave a lower bound for the smallest number of edges needed to biconnect an undirected graph and proved that the lower bound can always be achieved. Rosenthal and Goldner [RG77] developed a linear time sequential algorithm for finding a smallest biconnectivity augmentation; however, the algorithm in [RG77] contains an error. Hsu and Ramachandran [HR93] gave a corrected linear time sequential algorithm. An $O(\log^2 n)$ time parallel algorithm on an EREW PRAM using a linear number of processors for this problem was also given in Hsu and Ramachandran [HR93].

Fernández-Baca and Williams [FBW89] considered the smallest augmentation problem for reaching biconnectivity on hierarchically defined graphs. This version of the augmentation problem has applications in VLSI circuit design. They obtained polynomial time algorithms for the above problems.

Watanabe and Nakamura [WN93, WN88, WN90] gave an $O(n \cdot (n+m)^2)$ time sequential algorithm for finding a smallest augmentation to triconnect a graph with n vertices and m edges. Hsu and Ramachandran [HR91] gave a linear time algorithm for this problem. (Independently, Jordán [Jor93b] gave a different linear time algorithm for the special case of optimally triconnecting a biconnected graph.)

There is no polynomial time algorithm known for finding a smallest augmentation to k-vertex-connect an undirected graph, for $k > 4$. Although no polynomial time solution is known for this problem. Jordán [Jor93b] gave an approximation algorithm for undirected graphs that uses no more than $k-3$ edges to k-vertex-connect a $(k-1)$-vertex-connected graph. There are also some results known for augmenting planar graphs and outerplanar

graphs [Kan93].

The above results are for augmenting undirected graphs. For directed graph augmentation, Masuzawa, Hagihara, and Tokura in [MHT87] studied this problem when the input graph is a directed oriented tree. Their algorithm runs in $O(\lambda \cdot n)$ time where λ is the vertex-connectivity of the resulting graph. Jordán [Jor93a] gave a polynomial time approximation algorithm that uses no more than k extra edges for augmenting a $(k-1)$-vertex-connected directed graph to achieve k-vertex-connectivity. Very recently, Frank and Jordán [FJ93] gave a polynomial-time algorithm to solve the smallest vertex-connectivity augmentation problem on directed graphs exactly. Their algorithm increases the vertex-connectivity of a directed graph by any given δ optimally.

2.2 Edge-Connectivity Augmentations

For the problem of finding a smallest augmentation for a graph to reach a given edge connectivity property, several polynomial time algorithms and efficient parallel algorithms on outerplanar graphs, hierarchically defined graphs, undirected graphs, directed graphs and mixed graphs are known. These results can be found in [CS89, ET76, FBW89, Fra92, Gab91, Gus87, Hsu93, KU86, Kan93, NGM90, Sor88, UKW88, Wat87, WN87, WYO91].

2.3 Augmenting a Weighted Graph

Another version of the problem is to augment a graph, with a weight assigned to each edge, to meet a connectivity requirement using a set of edges with a minimum total cost. The decision version of several related problems have been proved to be **NP**-hard. These results can be found in [ET76, Fra92, FJ81, KT92, WHN90, WN93, WNN89].

3 Definitions

We use the following notations for performing operations on graphs. Let $G = (V, E)$ be a graph with the set of vertices V and the set of edges E, and let U be a set of vertices in G. The graph $G - U$ is the induced subgraph of G on $V \setminus U$. Let E' be a subset of edges of E. $G - (E' \cup U)$ is the resulting graph obtained from $G - U$ after removing edges in E'.

We then give definitions used in this paper. They are used in characterizing triconnected graphs.

Vertex-Connectivity

A graph G with at least $k+1$ vertices is *k-connected*, $k \geq 2$, if and only if G is a complete graph with $k+1$ vertices or the removal of any set of vertices with cardinality less than k does not disconnect G. The *vertex-connectivity* of G is k if G is k-connected, but not $(k+1)$-connected. Let $\mathcal{U}$ be a minimal set of vertices such that the number of components in the resulting graph obtained from G by removing $\mathcal{U}$ is more than the number of components in G. The set of vertices $\mathcal{U}$ is a *separating k-set.*

Separating Triplet

Let $\mathcal{U}$ be a separating 3-set in a triconnected graph. If $|\mathcal{U}| = 3$, it is a *separating triplet.* Let $\text{com}(G)$ be the number of connected components in a graph G. The *degree* $d(\mathcal{S})$ of a separating set $\mathcal{S}$ in G is $\text{com}(G-\mathcal{S}) - \text{com}(G) + 1$, which is at least 2. Note that if G is connected, $d(\mathcal{S})$ equals the number of connected components in the resulting graph obtained from G by removing $\mathcal{S}$. Thus the minimum number of edges needed to add to G such that $\mathcal{S}$ is no longer a separating triplet.

It is worthwhile noting that though this paper uses several properties of triconnected graphs derived in [KTDBC91], the definition of a separating triplet given in [KTDBC91] is different from what we have in this paper. In [KTDBC91], a separating triplet in a triconnected graph G is a set $\mathcal{Z} = \{\tau_1, \tau_2, \tau_3\}$ such that $G - \mathcal{Z}$ is disconnected, where each τ_i, $1 \leq i \leq 3$, is either a vertex or an edge. Allowing edges in a separating triplet can reduce the total number of separating triplets by a constant factor. The structure of the set of all separating triplets is also easier to describe than the case when allowing vertex-only separating triplets. In the above, $\mathcal{Z}$ represents all combinations of vertex-only separating triplets of the form $\{v_1, v_2, v_3\}$, where v_i is an endpoint of τ_i if τ_i is an edge; $v_i = \tau_i$ if τ_i is a vertex. For example, the two vertex-only separating triplets obtained from expanding {(1, 2), 3, 4} in Figure 1 are {1, 3, 4} and {2, 3, 4}. The definition for separating triplets given in [KTDBC91] and this paper is equivalent. We can "expand" each separating triplet $\mathcal{Z}$ given in [KTDBC91] (that might contain edges) to a set of vertex-only separating triplets in constant time. For the rest of this paper, separating triplets contains only vertices unless stated otherwise.

Using this above definition for separating triplets gives the following unwanted side

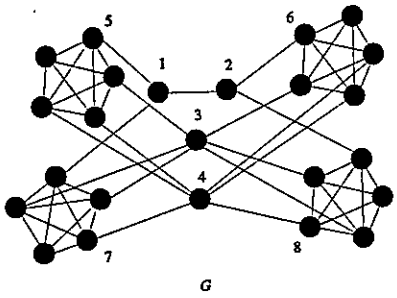

Figure 1: Illustrating a different definition of separating triplets given in [KTDBC91]. The set {(1, 2), 3, 4} is a separating triplet according to [KTDBC91], but it is not a separating triplet according to our definition.

effect that we want to avoid. Consider the graph G in Figure 1, $\mathcal{Z} = \{(1,2),3,4\}$ is a valid separating triplet as defined in [KTDBC91]. The resulting graph obtained by removing $\mathcal{Z}$ contains 2 connected components. We can no longer have the desired property that by adding $d(\mathcal{Z})-1$ edges, the set of separating triplets represented by $\mathcal{Z}$ are no longer separating triplets in the resulting graph. In fact, we must add at least two edges (e.g., (5, 6) and (7, 8)) to G to four-connect G.

Redundant Separating Triplet

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two separating triplets. Let $\{\mathcal{C}_1,\ldots,\mathcal{C}_h\}$ be the connected components in $G-\mathcal{S}_1$ and let $\{\mathcal{D}_1,\ldots,\mathcal{D}_{h'}\}$ be the connected components in $G-\mathcal{S}_2$. Let $V(\mathcal{C}_i)$ be the set of vertices in $\mathcal{C}_i$, $1 \le i \le h$, and let $V(\mathcal{D}_i)$ be the set of vertices in $\mathcal{D}_i$, $1 \le i \le h'$. If all of the following conditions are true: (1) $h \ge h'$; (2) we can partition $\{\mathcal{C}_1,\ldots,\mathcal{C}_h\}$ into h' disjoint subsets $\Im_1,\ldots,\Im_{h'}$ such that for all i either $\Im_i$ is $\{D_i\}$ or

$$\left(\bigcup_{\forall j,\mathcal{C}_j\in\Im_i} V(\mathcal{C}_j)\right) \setminus V(\mathcal{D}_i) \subseteq \mathcal{S}_2,$$

then $\mathcal{S}_1$ is *superfluous* with respect to $\mathcal{S}_2$. A separating triplet $\mathcal{S}$ is *redundant* in a set of separating triplets $\aleph$ if $\mathcal{S}$ is superfluous with respect to a separating triplet in $\aleph$ and each degree-3 vertex in $\mathcal{S}$ is contained in a separating triplet in $\aleph$.

Let $\aleph$ be a set of separating triplets in a triconnected graph G and let $\mathcal{S}$ be a redundant separating triplet in $\aleph$. Given a set of edges $\mathcal{A}$, it is easy to see that if $\aleph \setminus \mathcal{S}$ contains no separating triplet in $G \cup \mathcal{A}$, then $\mathcal{S}$ is not a separating triplet in $G \cup \mathcal{A}$. Thus we do not have to "worry about" $\mathcal{S}$ in four-connecting G if we can "take care of" the rest of the separating triplets in $\aleph$.

We use the following lemma to derive a set of maximal separating triplets without redundancy in a triconnected graph.

Lemma 1 *Given all separating triplets in a triconnected graph G as defined in [KTDBC91] with possibly edges in each of the separating triplets, we can select separating triplets (containing only vertices) obtained from expanding a separating triplet containing edges such that there are no redundant separating triplets and all degree-3 vertices in G are contained in exactly one expanded separating triplet.*

Proof: We expand all separating triplets that might contain edges and consider each expanded separating triplet one by one. Let $\mathcal{Z}$ be an original separating triplet that includes edges.
Case 1: If there is only one edge (u, v) in $\mathcal{Z}$, let $\mathcal{Z} = \{(u, v), w, x\}$. The two possible candidates for vertex-only separating triplets are $\{u, w, x\}$ and $\{v, w, x\}$. If both of them are redundant, then we remove an arbitrary redundant one.
Case 2: If there are two edges (u_1, u_2) and (v_1, v_2) in $\mathcal{Z}$, let w be the vertex in $\mathcal{Z}$. Assume without lose of generality that u_2 and v_2 are connected in $G - \{u_1, v_1, w\}$. Thus u_1 and v_1 are connected in $G - \{u_2, v_2, w\}$. The separating triplets $\{u_1, v_2, w\}$ and $\{u_2, v_1, w\}$ are redundant.
Case 3: If there are three edges (u_1, u_2), (v_1, v_2), and (w_1, w_2) in $\mathcal{Z}$. Assume without lose of generality that u_2, v_2, and w_2 are connected in $G - \{u_1, v_1, w_1\}$. Thus u_1, v_1, and w_1 are connected in $G - \{u_2, v_2, w_2\}$. Any cardinality-3 subset in $\{u_1, u_2, v_1, v_2, w_1, w_2\}$ except $\{u_1, v_1, w_1\}$ and $\{u_2, v_2, w_2\}$ is a redundant separating triplet. □

For the rest of this paper, a separating triplet of a graph G is obtained from a maximal set of separating triplets by applying Lemma 1 unless specified otherwise. The previous lemma states that it might be possible that a special 4-block leaf is inside two separating triplets. The following corollary rules out this possibility.

Corollary 1 *Any special 4-block leaf cannot be contained in more than one separating triplet.*

Proof: Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two separating triplets that are expanded from the same phase given in the proof of Lemma 1.
Case 1: There is exactly one vertex u in $\mathcal{S}_1 \cap \mathcal{S}_2$ and the degree of u is 3. Then u must be adjacent to exactly one vertex v in one of the connected components in $G - \mathcal{S}_1$. Thus $\mathcal{S}_1' = \mathcal{S}_1 \cup \{v\} \setminus \{u\}$ is also a separating triplet. Our expanding algorithm given in the proof

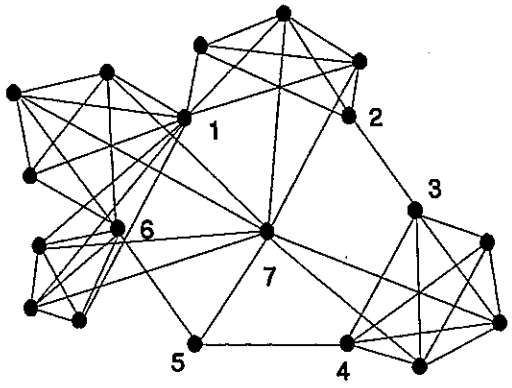

Figure 2: Illustrating a wheel $\{7\} \cup \{1,2,3,4,5,6\}$. The degree of this wheel is 5, i.e., 4 (the number of components we got after removing the wheel) plus 1 (the number of degree-3 vertices in the wheel that are adjacent to the center vertex 7).

of Lemma 1 must have generated $\mathcal{S}_1'$ and $\mathcal{S}_2$ instead of $\mathcal{S}_1$ and $\mathcal{S}_2$.
Case 2: There are exactly two vertices u and v in $\mathcal{S}_1 \cap \mathcal{S}_2$. If the degree of both u and v are 3, then we can derive this corollary using case 1. It is impossible that $\{u,v\}$ is a special 4-block leaf, since otherwise we can find a vertex w that is adjacent to both u and v such that $(\mathcal{S}_1 \cup \{w\}) \setminus \mathcal{S}_2$ is a cardinality-2 separating set. □

Wheel and Flower

A set of at least three separating triplets with one common vertex c is a *wheel* in [KTDBC91]. A wheel can be represented by the set of vertices $\{c\} \cup \{s_0, s_1, \ldots, s_{q-1}\}$ which satisfies the following conditions: (i) $q > 2$; (ii) $\forall i \neq j$, $\{c, s_i, s_j\}$ is a separating triplet unless in the case that $j = ((i+1) \bmod q)$ and (s_i, s_j) is an edge in G; (iii) c is adjacent to a vertex in each of the connected components created by removing any of the separating triplets in the wheel; (iv) $\forall j \neq (i+1) \bmod q$, $\{c, s_i, s_j\}$ is a degree-2 separating triplet. The vertex c is the *center* of the wheel [KTDBC91]. For more details, see [KTDBC91].

The *degree* of a wheel $\mathcal{W} = \{c\} \cup \{s_0, s_1, \ldots, s_{q-1}\}$, $d(\mathcal{W})$, is the number of connected components in $G - \{c, s_0, \ldots, s_{q-1}\}$ plus the number of degree-3 vertices in $\{s_0, s_1, \ldots, s_{q-1}\}$ that are adjacent to c. The degree of a wheel must be at least 3. Note that the number of degree-3 vertices in $\{s_0, s_1, \ldots, s_{q-1}\}$ that are adjacent to c is equal to the number of separating triplets in $\{(c, s_i, s_{(i+2) \bmod q}) \mid 0 \leq i < q$, such that $s_{(i+1) \bmod q}$ is degree 3 in $G\}$. An example is shown in Figure 2.

A separating triplet is a *flower* [KTDBC91] if its degree is greater than 2, or is not

in any wheel. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two separating triplets. We denote $\mathcal{S}_1 \Re \mathcal{S}_2$ if if $\mathcal{S}_1$ and $\mathcal{S}_2$ are obtained from the same expanding phase as described in Lemma 1. Given a separating triplet $\mathcal{S}$, the *flower cluster* for $\mathcal{S}$ is the set of all separating triplets $\{\mathcal{S}' \mid \mathcal{S} \Re \mathcal{S}'\}$. Note that $\Re$ defines a binary relation that is symmetric and reflexive, but not transitive.

Each of the (non-redundant) separating triplets in a triconnected graph G is either represented by a flower or is in a wheel. We can construct an $O(n)$-space representation for all (non-redundant) separating triplets (i.e., flowers and wheels) in a triconnected graph with n vertices and m edges using $O(n \cdot \alpha(m,n) + m)$ time [KTDBC91].

K-Block

Let $G = (V, E)$ be a graph with vertex-connectivity $k-1$. The *neighbor* of a set of vertices U in G is the set of vertices (not including any vertex in U) that are adjacent to a vertex in U. A *k-block* in G is either (i) a minimal set of vertices $\mathcal{B}$ in a separating $(k-1)$-set with exactly $k-1$ neighbors in $V \setminus \mathcal{B}$ (these are *special k-blocks*) or (ii) a maximal set of vertices $\mathcal{B}$ such that there are at least k vertex-disjoint paths in G between any two vertices in $\mathcal{B}$ and $\mathcal{B}$ is not a special k-block (these are *non-special k-blocks*). Note that a set consisting of a single vertex of degree $k-1$ in G is a k-block by (ii). A *k-block leaf* in G is a k-block $\mathcal{B}_l$ with exactly $k-1$ neighbors in $V \setminus \mathcal{B}_l$. Note also that every special k-block is a k-block leaf. Given a non-special 4-block leaf $\mathcal{B}$, the vertices in $\mathcal{B}$ that are not in any separating triplet that can separate part of vertices in $\mathcal{B}$ from the rest of the vertices in G are *demanding vertices.* We let every vertex in a special 4-block leaf be a demanding vertex. Intuitively, after adding an edge between a demanding vertex of $\mathcal{B}$ and a vertex not in $\mathcal{B}$ or any separating triplet, $\mathcal{B}$ is no longer a 4-block leaf.

The following claim states that we can always find a demanding vertex in every 4-block leaf.

Claim 1 *Every 4-block leaf contains at least one demanding vertex.*

Proof: By definition, every vertex in a special 4-block leaf is a demanding vertex. Thus we only have to prove this claim holds for non-special 4-block leaves. Let $\mathcal{B}$ be a non-special 4-block with more than one vertex. (If $\mathcal{B}$ contains only one vertex u, then u is a demanding vertex by definition.) Let $\mathcal{S}'$ be the neighbor of $\mathcal{B}$. By definition, $\mathcal{S}'$ is a separating triplet. If for each vertex u in $\mathcal{B}$, u is in a separating triplet $\mathcal{S}_u$ such that the induced subgraph on vertices $\mathcal{B} \setminus \mathcal{S}_u$ is a connected component in $G - \mathcal{S}_u$, then there is no demanding vertex in $\mathcal{B}$. We prove in the following cases that the above condition is impossible. In proving the

following cases, we assume that $\mathcal{B} \notin \mathcal{S}_u$ since otherwise $\mathcal{B}$ is a special 4-block leaf. Let u' be a vertex in $\mathcal{B} \setminus \mathcal{S}_u$.
Case 1: $\mathcal{S}_u \notin \mathcal{B}$ and $\mathcal{S}_u \cap \mathcal{S}' = \emptyset$. This implies $\mathcal{S}_u \setminus \mathcal{B}$ is a separating set with cardinality less than 3. This contradicts the fact that G is triconnected.
Case 2: $\mathcal{S}_u \in \mathcal{B}$. If u' is also in a separating triplet $\mathcal{S}''$, then $\mathcal{S}'' \in \mathcal{B}$. All paths from u to u' must pass through $\mathcal{S}''$, and u and u' are not adjacent in G. Thus they cannot be in the same 4-block.
Case 3: $\mathcal{S}_u \cap \mathcal{S}' \neq \emptyset$ and $\mathcal{S}_u \in \mathcal{B} \cup \mathcal{S}'$. If u' is also in a separating triplet $\mathcal{S}''$, then we have reduce the case to case 2.
Case 4: $\mathcal{S}_u \cap \mathcal{S}' \neq \emptyset$, $\mathcal{S}_u \cap \mathcal{B} \neq \emptyset$, and $\mathcal{S}_u \notin \mathcal{B} \cup \mathcal{S}'$. Let $\mathcal{S}_u = \{u, c, v\}$ such that $\mathcal{S}_u \cap \mathcal{S}' = \{c\}$. Then $\mathcal{S}$ and $\mathcal{S}'$ form a wheel with the center c. Let $w \in \mathcal{S}'$ such that the set of vertices $\mathcal{B} \setminus \{u\}$ form a connected component in $G - (\{u, c, w\} \cup \mathcal{S}')$. If u' is also in a separating triplet $\mathcal{S}''$, then we can reduce this case to case 3. □

Using procedures in [KTDBC91], we can find all of the 4-block leaves in a triconnected graph with n vertices and m edges in $O(n \cdot \alpha(m, n) + m)$ time.

Four-Block Tree

From [KTDBC91] we know that we can decompose vertices in a triconnected graph into the following 3 types: (i) 4-blocks; (ii) wheels; (iii) separating triplets that are not in a wheel. We modify the decomposition tree in [KTDBC91] to derive the *four-block tree*, 4-blk(G), of a triconnected graph G as follows. We create an R-vertex for each 4-block that is not special (i.e., not in a separating triplet), an F-vertex for each separating triplet that is not in a wheel, and a W-vertex for each wheel. For each wheel $\mathcal{W} = \{c\} \cup \{s_0, s_1, \ldots, s_{q-1}\}$, we also create the following vertices. An F-vertex is created for each separating triplet of the form $\{c, s_i, s_{(i+1) \bmod q}\}$ in $\mathcal{W}$. An R-vertex is created for every degree-3 vertex s in $\{s_0, s_1, \ldots, s_{q-1}\}$ that is adjacent to c and an F-vertex is created for the three vertices that are adjacent to s.

Let r be an R-vertex we created for the 4-block tree and let $\mathcal{B}_r$ be its corresponding 4-block in G. Let f be an F-vertex we created for the 4-block tree and let $\mathcal{S}_f$ be its corresponding separating triplet in G. We create an edge in the 4-block tree between f and r if in the graph G, each vertex in $\mathcal{S}_f$ is either in $\mathcal{B}_r$ or adjacent to a vertex in $\mathcal{B}_r$. There is an edge between an F-vertex f and a W-vertex w if the the wheel corresponding to w contains the separating triplet corresponding to f. A *dummy R-vertex* is created and adjacent to each pair of F-vertices f_1 and f_2 with the properties that they are not already connected

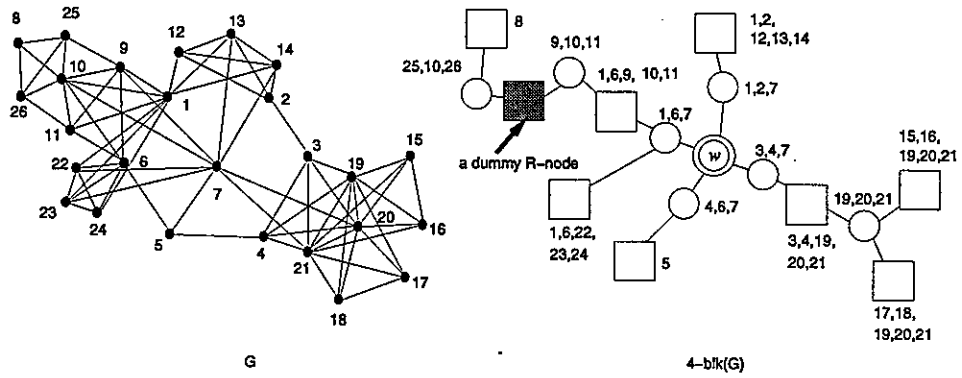

Figure 3: Illustrating a triconnected graph and its 4-blk(G). We use rectangles, circles and two concentric circles to represent R-vertices, F-vertices and W-vertices, respectively. The numbers beside each vertex u in 4-blk(G) represent the set of vertices corresponding to u. The W-vertex w in 4-blk(G) corresponds to the wheel $\{7\} \cup \{1,2,3,4,5,6\}$ in G.

and their corresponding separating triplets are in the same flower cluster. An example of a 4-block tree is shown in Figure 3.

Note that a degree-1 R-vertex in 4-blk(G) corresponds to a 4-block leaf, but the reverse is not necessarily true. We do not represent certain special 4-block leaves (explained later) and all degree-3 vertices that are centers of wheels in 4-blk(G). A special 4-block leaf $\{v\}$, where v is a vertex, is represented by an R-vertex in 4-blk(G) if v is not the center of a wheel w and it is in one of separating triplets of w. The degree of a flower $\mathcal{F}$ in G is the degree of its corresponding F-vertex in 4-blk(G). Note also that the degree of a wheel $\mathcal{W}$ in G is equal to the number of components in 4-blk(G) by removing its corresponding W-vertex w and all F-vertices that are adjacent to w.

Star Wheel

A wheel $\mathcal{W}$ in G is a *star wheel* if $d(\mathcal{W})$ equals the number of leaves in 4-blk(G) and every special 4-block leaf in $\mathcal{W}$ is either adjacent to or equal to the center of $\mathcal{W}$. A star wheel $\mathcal{W}$ with the center c has the property that every 4-block leaf in G (not including $\{c\}$ if it is a 4-block leaf) can be separated from G by a separating triplet containing the center c. If G contains a star wheel $\mathcal{W}$, then $\mathcal{W}$ is the only wheel in G. Note also that the degree of a wheel is less than or equal to the degree of its center in G.

K-Connectivity Augmentation Number

The *k-connectivity augmentation number* for a graph G is the smallest number of edges that

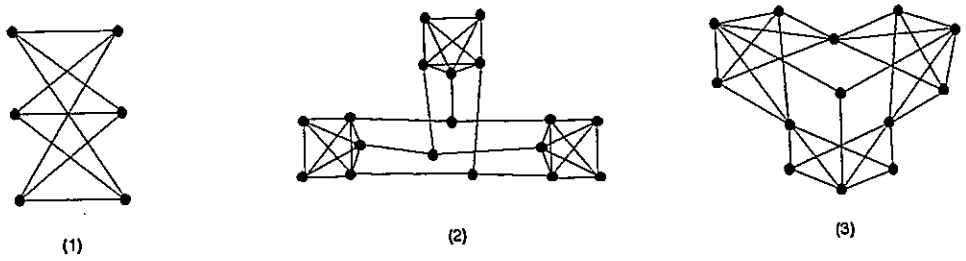

Figure 4: Illustrating three graphs where in each case the value derived by applying a simple lower bound does not equal its four-connectivity augmentation number.

must be added to G in order to k-connect G.

4 A Lower Bound for the Four-Connectivity Augmentation Number

We first give a simple lower bound for the four-connectivity augmentation number that is similar to the ones for biconnectivity augmentation [ET76] and triconnectivity augmentation [HR91]. We show that this above lower bound is not always equal to the four-connectivity augmentation number [Jor92, KT91]. We then give a modified lower bound. This new lower bound turns out to be the exact number of edges that we must add to reach four-connectivity (see proofs in Section 5). Finally, we show relations between the two lower bounds.

4.1 A Simple Lower Bound

Given a graph G with vertex-connectivity $k-1$, it is well-known that $\max\{\lceil \frac{\ell_k}{2} \rceil, d-1\}$ is a lower bound for the k-connectivity augmentation number, where ℓ_k is the number of k-block leaves in G and d is the maximum degree among all separating $(k-1)$-sets in G [ET76]. It is also well-known (see, for example, [ET76, HR91]) that for $k = 2$ and 3, this lower bound equals the k-connectivity augmentation number. For $k = 4$, however, several researchers [Jor92, KT91] have observed that this value is not always equal to the four-connectivity augmentation number. Examples are given in Figure 4. Figure 4.(1) is from [Jor92] and Figure 4.(2) is from [KT91]. Note that if we apply the above lower bound in each of the three graphs in Figure 4, the values we obtain for Figures 4.(1), 4.(2), and 4.(3) are 3, 3, and 2, respectively, while we need one more edge in each graph to four-connect it.

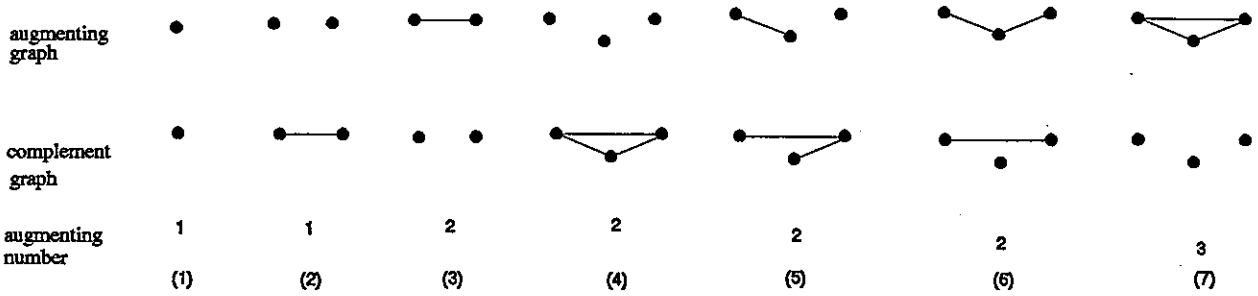

Figure 5: Illustrating the seven types of augmenting graphs, their complement graphs and augmenting numbers that one can get for a separating triplet in a triconnected graph.

4.2 A Better Lower Bound

Notice that in the previous lower bound, for every separating triplet $\mathcal{S}$ in the triconnected graph $G = \{V, E\}$, we must add at least $d(\mathcal{S}) - 1$ edges among vertices in $V \setminus \mathcal{S}$ to four-connect G, where $d(\mathcal{S})$ is the degree of $\mathcal{S}$ (i.e., the number of connected components in $G - \mathcal{S}$); otherwise, $\mathcal{S}$ remains a separating triplet. Let the set of edges added be $\mathcal{A}_1(\mathcal{S})$. Recall that we must add at least one edge into every 4-block leaf $\mathcal{B}$ to four-connect G; otherwise, $\mathcal{B}$ remains a 4-block leaf. Since it is possible that $\mathcal{S}$ contains some 4-block leaves, we need to know the minimum number of edges needed to eliminate all 4-block leaves inside $\mathcal{S}$. Let the set of edges added be $\mathcal{A}_2(\mathcal{S})$. We know that $\mathcal{A}_1(\mathcal{S}) \cap \mathcal{A}_2(\mathcal{S}) = \emptyset$. The previous lower bound gives a bound on the cardinality of $\mathcal{A}_1(\mathcal{S})$, but not that of $\mathcal{A}_2(\mathcal{S})$. In the following paragraph, we define a quantity to measure the cardinality of $\mathcal{A}_2(\mathcal{S})$.

Let two 4-block leaves $\mathcal{B}_1$ and $\mathcal{B}_2$ be *adjacent* if there is an edge in G between every demanding vertex in $\mathcal{B}_1$ and every demanding vertex in $\mathcal{B}_2$. We create an *augmenting graph for* $\mathcal{S}$, $\mathcal{G}(\mathcal{S})$, as follows. Let $\mathcal{Q}_\mathcal{S}$ be the set of special 4-block leaves that are in the separating triplet $\mathcal{S}$ of a triconnected graph G. For each special 4-block leaf in $\mathcal{Q}_\mathcal{S}$, we create a vertex in $\mathcal{G}(\mathcal{S})$. There is an edge between two vertices v_1 and v_2 in $\mathcal{G}(\mathcal{S})$ if their corresponding 4-block leaves are adjacent. Let $\overline{\mathcal{G}(\mathcal{S})}$ be the complement graph of $\mathcal{G}(\mathcal{S})$. The seven types of augmenting graphs and their complement graphs are illustrated in Figure 5.

Definition 1 *The* **augmenting number** $a(\mathcal{S})$ *for a separating triplet* $\mathcal{S}$ *in a triconnected graph is the number of edges in a maximum matching* $\mathcal{M}$ *of* $\overline{\mathcal{G}(\mathcal{S})}$ *plus the number of isolated vertices in* $\mathcal{M}$.

The augmenting numbers for the seven types of augmenting graphs are shown in Figure 5. Note that in a triconnected graph, each special 4-block leaf must receive at least one new incoming edge in order to four-connect the input graph. The augmenting number $a(\mathcal{S})$ is exactly the minimum number of edges needed in the separating triplet $\mathcal{S}$ in order to four-connect the input graph, i.e., $\mathcal{A}_2(\mathcal{S})$. The augmenting number of a separating set that does not contain any special 4-block leaf is 0. Note also that we can define the *augmenting number* $a(\{c\})$ for a set $\{c\}$ such that c is the center of a wheel using a similar approach. It is obvious that $a(\{c\}) \leq 1$.

We also need the following definition before we show our new lower bound for the four-connectivity augmentation number.

Definition 2 *Let G be a triconnected graph with ℓ 4-block leaves. The* **leaf constraint** *of G, $\mathrm{lc}(G)$, is $\lceil \frac{\ell}{2} \rceil$. The* **degree constraint** *of a separating triplet S in G, $\mathrm{dc}(\mathcal{S})$, is $d(\mathcal{S}) - 1 + a(\mathcal{S})$, where $d(\mathcal{S})$ is the degree of $\mathcal{S}$ and $a(\mathcal{S})$ is the augmenting number of $\mathcal{S}$. The degree constraint of G, $\mathrm{dc}(G)$, is the maximum degree constraint among all separating triplets in G. The* **wheel constraint** *of a star wheel $\mathcal{W}$ with center c in G, $\mathrm{wc}(\mathcal{W})$, is $\lceil \frac{d(\mathcal{W})}{2} \rceil + a(\{c\})$, where $d(\mathcal{W})$ is the degree of $\mathcal{W}$ and $a(\{c\})$ is the augmenting number of $\{c\}$. The wheel constraint of G, $\mathrm{wc}(G)$, is zero if there is no star wheel in G; otherwise it is the wheel constraint of the star wheel in G.*

Note that we only define wheel constraint for a star wheel. Intuitively, the value of the leaf constraint is the minimum number of edges we need to add to eliminate all 4-block leaves. The value of the degree constraint for a separating triplet is the minimum number of edges we need to ass to eliminate the separating triplet as well as the 4-block leaves within it. The value of the wheel constraint for a graph with a star wheel is the minimum number of edges we need to add to eliminate the star wheel from the graph. We now give a better lower bound on the 4-connectivity augmentation number for a triconnected graph.

Lemma 2 *We need at least* $\max\{\mathrm{lc}(G), \mathrm{dc}(G), \mathrm{wc}(G)\}$ *edges to four-connect a triconnected graph G.*

Proof: Let $\mathcal{A}$ be a set of edges such that $G' = G \cup \mathcal{A}$ is four-connected. For each 4-block leaf B in G, we need one new incoming edge to a vertex in B; otherwise B is still a 4-block leaf in G'. This gives the first component of the lower bound.

For each separating triplet $\mathcal{S}$ in G, $G - \mathcal{S}$ contains $d(\mathcal{S})$ connected components. We need to add at least $d(\mathcal{S})-1$ edges between vertices in $G-\mathcal{S}$, otherwise $\mathcal{S}$ is still a separating triplet in G'. In addition to that, we need to add at least $a(\mathcal{S})$ edges such that at least one of the two end points of each new edge is in $\mathcal{S}$; otherwise $\mathcal{S}$ contains a special 4-block leaf. This gives the second term of the lower bound.

Recall that G contains at most one star wheel and if there exists a star wheel, then it is the only wheel in G. Given the star wheel $\mathcal{W}$ with the center c, 4-blk(G) contains exactly $d(\mathcal{W})$ degree-1 R-vertices. Thus we need to add at least $\lceil \frac{d(\mathcal{W})}{2} \rceil$ edges between vertices in $G - \{c\}$; otherwise, G' contains some 4-block leaves. In addition to that, we need to add $a(\{c\})$ non-self-loop edges such that at least one of the two end points of each new edge is in $\{c\}$; otherwise $\{c\}$ is still a special 4-block leaf. This gives the third term of the lower bound. □

It is worthwhile noting that if max{lc(G), dc(G), wc(G)} is zero, then G is four-connected.

4.3 A Comparison of the Two Lower Bounds

We first observe the following relation between the wheel constraint and the leaf constraint. Note that if there exists a star wheel $\mathcal{W}$ with degree $d(\mathcal{W})$, there are exactly $d(\mathcal{W})$ 4-block leaves in G if the center is not degree-3. If the center of the star wheel is degree-3, then there are exactly $d(\mathcal{W}) + 1$ 4-block leaves in G. Thus the wheel constraint is greater than the leaf constraint if and only if there is a star wheel with a degree-3 center. We know that the degree of any wheel is less than or equal to the degree of its center. Thus the value of wc(G) s 3, if wc(G) is greater than lc(G).

The following claim states the relation between the degree constraint of a separating triplet and the leaf constraint.

Claim 2 *Let $\mathcal{S}$ be a separating triplet with degree $d(\mathcal{S})$ and h special 4-block leaves. Then there are at least $h + d(\mathcal{S})$ 4-block leaves in G.*

Proof: It is easy to see that there is a demanding vertex in each of connected components in $G-\mathcal{S}$. Thus we can find at least one 4-block leaf in each of the $d(\mathcal{S})$ connected components in $G - \mathcal{S}$. Hence the claim holds. □

The following easily proved claim and its corollary states the relation between the degree constraint of a separating triplet and the number of special 4-block leaves within it.

Claim 3 *Let* $\{a_1, a_2, a_3\}$ *be a separating triplet in a triconnected graph* G. *Then* a_i, $1 \leq i \leq 3$, *is incident on a vertex in every connected component in* $G - \{a_1, a_2, a_3\}$. □

Corollary 2 *The degree of a separating triplet* $\mathcal{S}$ *is no more than the largest degree among all vertices in* $\mathcal{S}$. □

From Corollary 2, we know that it is not possible for a triconnected graph to have type (6) or type (7) augmenting graphs as shown in Figure 5, since the degree of their underling separating triplet is 1. We also know that the degree of a separating triplet with a special 4-block leaf is at most 3 and at least 2. Thus $\mathrm{dc}(\mathcal{S})$ is greater than $d(\mathcal{S}) - 1$ if $\mathrm{dc}(\mathcal{S})$ equals either 3 or 4. Hence we have the following lemma.

Lemma 3 *Let* $\mathrm{low}_1(G)$ *be the lower bound given in Section 4.1 for a triconnected graph* G *and let* $\mathrm{low}_2(G)$ *be the lower bound given in Lemma 2 (Section 4.2). Then*
(i) $\mathrm{low}_1(G) = \mathrm{low}_2(G)$ *if* $\mathrm{low}_2(G) \notin \{3, 4\}$.
(ii) $\mathrm{low}_2(G) - \mathrm{low}_1(G) \in \{0, 1\}$. □

Thus the simple lower bound given in Section 4.1 extended from biconnectivity and triconnectivity is a good approximation for the four-connectivity augmentation number. The difference between two lower bounds is at most one.

5 Finding a Smallest Four-Connectivity Augmentation for a Triconnected Graph

We first explore properties of the 4-block tree that we will use to develop an algorithm for finding a smallest 4-connectivity augmentation. Then we describe our algorithm.

5.1 Properties of the Four-Block Tree

Degree of Separating Triplets

We will prove a lemma that relates the degree constraint of a separating triplet to the total

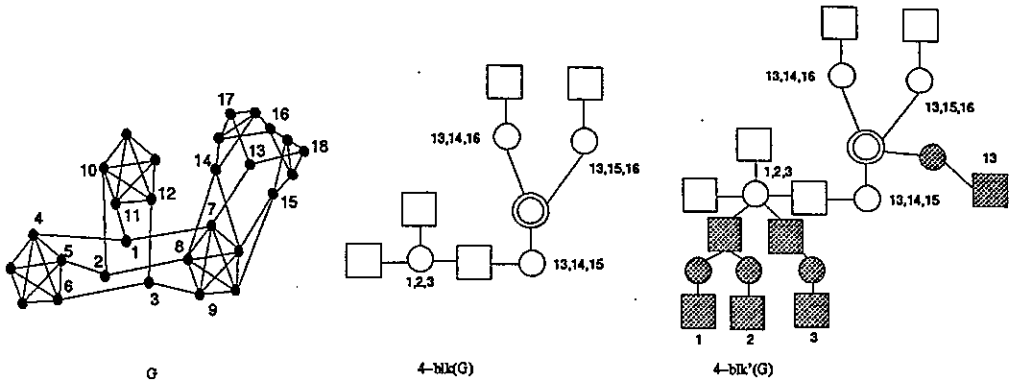

Figure 6: Illustrating a triconnected graph G, its 4-block graph 4-blk(G), and the modified 4-block graph 4-blk$'$(G). The added vertices in 4-blk$'$(G) are shadowed.

number of 4-block leaves in G. By doing this, we can identify graphs whose degree constraint dominates the lower bound given in Lemma 2. To reduce the four-connectivity augmentation number of these graphs by 1 by adding an edge, our algorithm only has to reduce the degree constraint by 1.

Lemma 4 *Let $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ be any three separating triplets in G. Then $\sum_{i=1}^{3} \mathrm{dc}(\mathcal{S}_i) \leq \ell+1$, where ℓ is the number of 4-block leaves in G.*

Proof: The input graph G is triconnected. We modify 4-blk(G) in the following way such that the number of leaves in the resulting tree equals ℓ and the degree of any F-vertex equals $\mathrm{dc}(\mathcal{S}) + 1$. Let the modified four-block graph be 4-blk$'$(G). For each W-vertex w with a degree-3 center c, we create an R-vertex r_c for c, an F-vertex f_c for the three vertices that are adjacent to c in G. We add edges (w, f_c) and (f_c, r_c). Thus r_c is a leaf. For each F-vertex whose corresponding separating triplet $\mathcal{S}'$ contains h special 4-block leaves, we attach $a(\mathcal{S}')$ subtrees with a total number of h degree-1 R-vertices. To do this, we might have to add a few "glue" vertices. According to Corollary 1, we know that all 4-block leaves is added once. From Figure 5, we also know that the number of special 4-block leaves in any separating triplet is greater than or equal to its augmenting number. Thus the above addition of subtrees can always be done. An example is illustrated in Figure 6.

The number of leaves in 4-blk$'$(G) is ℓ. Let f be an F-node in 4-blk$'$(G) whose corresponding separating triplet is $\mathcal{S}$. We know that the degree of f equals $\mathrm{dc}(\mathcal{S}) + 1$. It is easy to verify that the sum of degrees of any three internal vertices in a tree is less than or equal to 4 plus the number of leaves in a tree. □

Massive Vertex, Critical Vertex and Balanced Graph

A separating triplet $\mathcal{S}$ in a graph G is *massive* if $\mathrm{dc}(\mathcal{S}) > \mathrm{lc}(G)$. If the corresponding separating triplet of an F-vertex f is massive, then f is massive in 4-blk(G). A separating triplet $\mathcal{S}$ in a graph G is *critical* if $\mathrm{dc}(\mathcal{S}) = \mathrm{lc}(G)$. If the corresponding separating triplet of an F-vertex f is critical, then f is critical in 4-blk(G). A graph G is *balanced* if there is no massive separating triplet in G. If G is balanced, then its 4-blk(G) is also balanced. The following corollary of Lemma 4 gives the number of massive and critical vertices in 4-blk(G) and can easily be verified.

Corollary 3 *Let G be a graph with more than two non-special 4-block leaves. Then the following three conditions are true.*
(i) There is at most one massive F-vertex in 4-blk(G).
(ii) If there is a massive F-vertex, there is no critical F-vertex.
(iii) There are at most two critical F-vertices in 4-blk(G). □

Updating the Four-Block Tree

Let v_1 and v_2 be two demanding vertices. Let $\mathcal{B}_i$ be the 4-block leaf that contains v_i, $i \in \{1,2\}$. Let b_i, $i \in \{1,2\}$, be the vertex in 4-blk(G) such that (1) if v_i is in a non-special 4-block leaf $\mathcal{B}$, then b_i is the R-vertex in 4-blk(G) whose corresponding 4-block is $\mathcal{B}$; (2) if v_i is in a special 4-block leaf that is contained in a flower, then b_i is the F-vertex in 4-blk(G) whose corresponding separating triplet contains v_i; (3) if v_i is the center of a wheel $\mathcal{W}$, b_i is the W-vertex in 4-blk(G) whose corresponding wheel is $\mathcal{W}$. The vertex b_i is the *implied vertex* for $\mathcal{B}_i$, $i \in \{1,2\}$. The *implied path P between $\mathcal{B}_1$ and $\mathcal{B}_2$* is the path in 4-blk(G) between b_1 and b_2. Given 4-blk(G) and an edge (v_1, v_2) not in G, we can obtain 4-blk($G \cup \{(v_1, v_2)\}$) by performing local updating operations on P. For details, see [KTDBC91].

In summary, all 4-blocks corresponding to R-vertices in P are collapsed into a single 4-block. Edges in P are deleted. Every F-vertex in P is connected to the new R-vertex created. We *crack* wheels in a way that is similar to the cracking of a polygon for updating 3-block graphs (see [HR91] and [DBT90] for details). We define that P is *non-adjacent* on a wheel $\mathcal{W}$ if the cracking of $\mathcal{W}$ creates two new wheels. Note that it is possible that a separating triplet $\mathcal{S}$ in the original graph is no longer a separating triplet in the resulting graph by adding an edge. Recall that special 4-block leaves in a separating triplet are not represented in the 4-blk(G). Thus that it is possible that some special 4-block leaves in the original graph are no longer special after adding an edge, in which case we must add their

corresponding R-vertices to 4-blk(G).

Reducing the Degree Constraint of a Separating Triplet

We know that the degree constraint of a separating triplet by doing the following. Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two non-special 4-block leaves in G and let b_i, $1 \leq i \leq 2$, be the corresponding R-vertex of $\mathcal{B}_i$ in 4-blk(G). The path between b_1 and b_2 in 4-blk(G) passes through the F-vertex corresponding to $\mathcal{S}$. Let u_i, $1 \leq i \leq 2$, be a demanding vertex in $\mathcal{B}_i$. By adding an edge between u_1 and u_2 in G, the degree constraint of $\mathcal{S}$ is reduced by 1. We also notice the following corollary from the definitions of 4-blk(G) and the degree constraint.

Corollary 4 *Let $\mathcal{S}$ be a separating triplet that contains a special 4-block leaf. Then the following two conditions are true.*
(i) We can reduce dc($\mathcal{S}$) *by one by adding an edge between demanding vertices of two special 4-block leaves $\mathcal{B}_1$ and $\mathcal{B}_2$ in $\mathcal{S}$ such that $\mathcal{B}_1$ and $\mathcal{B}_2$ are not adjacent.*
(ii) Let $\mathcal{B}$ be a 4-block leaf not in $\mathcal{S}$ and let $\mathcal{B}'$ be a special 4-block leaf in $\mathcal{S}$. Let $\mathcal{S}'$ be a separating triplet corresponding to an internal vertex in the implied path of 4-blk(G) *between $\mathcal{B}$ and $\mathcal{B}'$. If we add an edge between a demanding vertex in $\mathcal{B}$ and a demanding vertex in $\mathcal{B}'$, the degree constraint of $\mathcal{S}'$ is reduced by one.* □

Note that part (i) in Corollary 4 can be verified by observing all different augmenting graphs for a triconnected graph (shown in Figure 5).

Reducing the Number of Four-Block Leaves

We now consider the conditions under which the adding of an edge reduces the leaf constraint lc(G) by 1. Before giving the condition, we need the following definition.

Definition 3 *Let f be an F-vertex in* 4-blk(G) *and let $\mathcal{S}$ be its corresponding separating triplet in G. The* **real degree** *of f is* dc($\mathcal{S}$)+1. *The real degree of a W-node with a degree-3 center in G is 1 plus its degree in* 4-blk(G). *The real degree of any other node is equal to its degree in* 4-blk(G).

Intuitively, the real degree of a vertex u is the degree of u in the modified 4-block tree given in the proof of Lemma 4.

Definition 4 (The leaf-connecting condition) *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two non-adjacent 4-block leaves in G. Let P be the implied path between $\mathcal{B}_1$ and $\mathcal{B}_2$ in* 4-blk(G). *Two 4-block*

leaves $\mathcal{B}_1$ and $\mathcal{B}_2$ satisfy the **leaf-connecting condition** *if at least one of the following conditions is true.*
(i) There are at least two vertices with real degree at least three in P.
(ii) There is at least one R-vertex with degree at least four in P.
(iii) The path P is non-adjacent on a W-vertex in P.
(iv) There is an internal vertex with real degree at least three in P and at least one of the 4-block leaves in $\{\mathcal{B}_1, \mathcal{B}_2\}$ is special.
(v) Both $\mathcal{B}_1$ and $\mathcal{B}_2$ are special and they do not share the same set of neighbors.

Lemma 5 *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two 4-block leaves in G that satisfy the leaf-connecting condition. We can find vertices v_i in $\mathcal{B}_i$, $i \in \{1,2\}$, such that* $\mathrm{lc}(G \cup \{(v_1, v_2)\}) = \mathrm{lc}(G) - 1$, *if* $\mathrm{lc}(G) \geq 2$.

Proof: Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the two 4-block leaves that satisfy the leaf-connecting condition. If they satisfy parts (i) to (iii) of the leaf-connecting, proofs similar to the ones given in [HR91] for finding a smallest triconnectivity augmentation can be used to prove this lemma.

Assume that $\mathcal{B}_1$ and $\mathcal{B}_2$ satisfy part (iv) or part (v) of the leaf-connecting condition. Since we add an edge between $\mathcal{B}_1$ and $\mathcal{B}_2$, both $\mathcal{B}_1$ and $\mathcal{B}_2$ are no longer 4-block leaves after adding the edge. We have to show that the new 4-block created is not a 4-block leaf. If $\mathcal{B}_1$ and $\mathcal{B}_2$ satisfy part (iv) of the leaf-connecting condition, without lose of generality, assume that $\mathcal{B}_1$ is special and is contained in the separating triplet represented by the F-vertex f. The new 4-block created is adjacent to at least two F-vertices. One of them is the degree-3 vertex q in P if q is an F-vertex; otherwise it is an F-node adjacent to q. The other F-vertex adjacent to the created 4-block is f. Thus the new 4-block created is not a 4-block leaf.

If $\mathcal{B}_1$ and $\mathcal{B}_2$ satisfy part (v) of the leaf-connecting condition and they are in the same separating triplet, then no 4-block is created. Otherwise, the created 4-block is adjacent to the two F-vertices whose corresponding separating triplets contain $\mathcal{B}_1$ and $\mathcal{B}_2$.

From the above discussion, we know that we can eliminate two 4-block leaves by adding an edge. Thus the lemma holds. □

5.2 The Algorithm

We now describe an algorithm for finding a smallest augmentation to four-connect a triconnected graph. Let $\delta = \mathrm{dc}(G) - \mathrm{lc}(G)$. The algorithm first adds 2δ edges to the graph such

that the resulting graph is balanced and the lower bound given in Lemma 2 is reduced by 2δ in the resulting graph.

After the input graph is balanced, we use different strategies to reduce the lower bound by 1 by properly adding an edge depending on which of the three constraints dominate the lower bound. If $\mathrm{lc}(G) \neq 2$ or $\mathrm{wc}(G) \neq 3$, there is no star wheel with a degree-3 center. Thus it is impossible for $\mathrm{wc}(G)$ to dominate the lower bound. We add an edge such that the number of 4-block leaves is reduced by 2. We also make sure that the degree constraint $\mathrm{dc}(G)$ is reduced by 1 if $\mathrm{dc}(G) = \mathrm{lc}(G)$. Since there is no star wheel with a degree-3 center, $\mathrm{wc}(G)$ is also reduced by 1 if $\mathrm{wc}(G) = \mathrm{lc}(G)$. Each time we add an edge, the resulting graph stays balanced and the lower bound is reduced by 1.

If $\mathrm{lc}(G) = 2$ and $\mathrm{wc}(G) = 3$, then there exists a star wheel with a degree-3 center. We reduce $\mathrm{wc}(G)$ by 1 by adding an edge between the degree-3 center and a demanding vertex of a 4-block leaf. Since $\mathrm{lc}(G) = 2$ and $\mathrm{wc}(G) = 3$, $\mathrm{dc}(G)$ is at most 2. Thus the lower bound can be reduced by 1 by adding an edge. We keep adding an edge at a time such that the lower bound given in Lemma 2 is reduced by 1. Thus we can find a smallest augmentation to four-connect a triconnected graph. We now describe our algorithm in detail.

Input Graph is Not Balanced

We use an approach that is similar to the one used in biconnectivity [HR93] and triconnectivity augmentations [HR91] to balance the input graph. Given a tree T and a vertex v in T, a *v-chain* [RG77] is a component in $T - \{v\}$ without any vertex of degree more than 2. The leaf of T in each v-chain is a *v-chain leaf* [RG77]. Given an unbalanced graph G, let $\delta = \mathrm{dc}(G) - \mathrm{lc}(G)$ and let 4-blk$'(G)$ be the modified 4-block tree given in the proof of Lemma 4. Let f be a massive F-vertex. It is easy to show that there are at least $2\delta + 2$ f-chains in 4-blk$'(G)$ [RG77, HR91]. Let β_i be a demanding vertex in the ith f-chain leaf. We add the set of edges $\{(\beta_i, \beta_{i+1}) \mid 1 \leq i \leq 2\delta\}$. It is also easy to show that the lower bound given in Lemma 2 is reduced by 2δ and the graph is balanced.

Input Graph is Balanced

We first describe the algorithm in Algorithm 1. Note that Algorithm 1 uses a subroutine shown in Algorithm 2 to handle the special case when the 4-block graph is a star. Then we give its proof of correctness.

Before we show the correctness of algorithm `aug3to4` in Theorem 1, we need the following claim and corollaries.

```
graph function aug3to4(graph G);
   T := 4-blk(G); root T at an arbitrary vertex; ℓ̃ := number of degree-1 R-vertices in T;
   while ∃ a 4-block leaf in G do
      if ∃ a degree-3 center c then
         u1 := the 4-block leaf {c};
1.       if lc(G) = 2 and wc(G) = 3 then {* The vertex c is the center of the only star wheel in G. *}
            let u2 be a a non-special 4-block leaf
         else if ∃ another degree-3 center c' non-adjacent to c then
            let u2 be the 4-block leaf {c'}
         else if ∃ a special 4-block leaf b non-adjacent to u1 then let u2 := b
         else if ∄ (degree-3 center or special 4-block leaf) non-adjacent to u1 then
            let u2 be a a 4-block leaf s.t. ∃ an internal vertex with
            real degree ≥ 3 in the implies path between u1 and u2 fi
      else if lc(G) ≠ 2 or wc(G) ≠ 3 then
         if ℓ̃ > 2 and ∃ two critical F-vertices f1 and f2 then
2.          find two non-special 4-block leaves u1 and u2 s.t. the implied path
            between them passes through f1 and f2
         else if ℓ̃ > 2 and ∃ only one critical F-vertex f1 then
            if ∃ two non-adjacent special 4-block leaves in the
            separating triplet S1 corresponding to f1 then
3.             let u1 and u2 be two non-adjacent 4-block leaves in S1
            else if ∄ two non-adjacent special 4-block leaves in the
            separating triplet S1 corresponding to f1 then
4.             let v be a vertex with the largest real degree among all vertices in T − {f1};
               if real degree of v in T ≥ 3 then
                  find two non-special 4-block leaves u1 and u2
                  s.t. the implied path between them passes through f1 and v fi
            fi {* The case when the degree of v in T < 3 will be handled in step 8 of procedure star. *}
         else if ∃ two vertices v1 and v2 with real degree ≥ 3 then
5.          find two non-special 4-block leaves u1 and u2 such
            that the implied path between them passes through v1 and v2
         else if ∃ an R-vertex v with degree ≥ 4 then
6.          find two non-special 4-block leaves u1 and u2 such
            that the implied path between them passes through v
         else if ∃ a W-vertex v with degree ≥ 4 then
7.          let u1 and u2 be two non-special 4-block leaves such
            that the implied path between them is non-adjacent on v
         else {* The graph T is a star with the center v. *} star(u1, u2, ℓ̃, T)
      fi;
      let yi, i ∈ {1, 2}, be a demanding vertex in ui s.t. (y1, y2) is not an edge in the current G;
      G := G ∪ {(y1, y2)}; update T, ℓ̃, lc(G), wc(G), and dc(G)
   od;
   return G
end aug3to4;
```

Algorithm 1: Algorithm for finding a smallest four-connectivity augmentation of a triconnected graph.

```
{* The input 4-block tree T is a star. Find u1 and u2 in T
such that we can connect them and reduce the augmentation number. *}
procedure star(modifies vertex u1, u2, integer ℓ̃, tree T);
        if there is one vertex v in T with degree ≥ 3 then
8.        find a vertex w that is closest to v in T s.t. w is a degree-1 R-vertex or
          an F-vertex whose corresponding separating triplet contains a special 4-block leaf v1;
          let w' be a vertex that is closest to w s.t. either w' is a degree-1 R-vertex or an F-vertex
          whose corresponding separating triplet contains a special 4-block leaf non-adjacent to v1;
          find 4-block leaves u1 and u2 whose implied path passes through w, w', and v
          {* The above step can be always done, since T is a star. *}
        {* Note that T is a path for all the cases below. *}
        else if ∃ two non-adjacent special 4-block leaves in a separating triplet S then
9.        let u1 and u2 be two non-adjacent special 4-block leaves in S
        else if ∃ two non-adjacent special 4-block leaves then
10.       let u1 and u2 be two special 4-block leaves
        else if ∃ a special 4-block leaf u1 then
              let u2 be a non-special 4-block leaf
        else {* There is no special 4-block leaf and ℓ̃ = 2. *}
          let u1 and u2 be the two 4-block leaves
          corresponding to the two degree-1 R-vertices in T
        fi
end star;
```

Algorithm 2: A subroutine called by algorithm aug3to4 to handle the case when the 4-block graph is a star.

Claim 4 *If* 4-blk(G) *contains two critical F-vertices* f_1 *and* f_2*, then every leaf is either in an* f_1*-chain or in an* f_2*-chain and the degree of any other vertex in* 4-blk(G) *is at most two.*

Proof: The proof of this claim is the same with a proof given in [RG77] for describing a similar situation in finding a biconnectivity augmentation. □

Corollary 5 *Let* f_1 *and* f_2 *be two critical vertices in* 4-blk(G) *and let* $\mathcal{S}_i$*,* $i \in \{1,2\}$*, be the corresponding separating triplet of* f_i*. If* $\mathcal{S}_i$*,* $i \in \{1,2\}$*, contains a special 4-block leaf, then the augmenting number of* f_i *is equal to the number of special 4-block leaves in* $\mathcal{S}_i$*.*

Proof: It is easy to check that Claim 4 holds for the modified 4-block tree we give in the proof of Lemma 4. We observe from Figure 5 that the augmenting number of a separating triplet is at most equal to the number of special 4-block leaves in it. If we have more special 4-block leaves than its augmenting number, then the modified 4-block tree we built does not satisfy the condition imposed by Claim 4. □

Corollary 6 *Let* f_1 *and* f_2 *be two critical F-vertices in* 4-blk(G)*. If the number of degree-1 R-vertices in* 4-blk(G) *is greater than 2 and the corresponding separating triplet of* f_i*,*

$i \in \{1,2\}$, contains a 4-block leaf $\mathcal{B}_i$, we can add an edge between a vertex in $\mathcal{B}_1$ and a vertex in $\mathcal{B}_2$ to reduce the lower bound given in Lemma 2 by one. □

Theorem 1 *Algorithm* `aug3to4` *adds the smallest number of edges to four-connect a triconnected graph.*

Proof: We will prove in the following paragraphs that the lower bound given in Lemma 2 is reduced by 1 each time we add an edge in all possible cases. We keep on adding edges this way until the lower bound is zero, in which case the graph is four-connected. Thus the number of edges added is minimum and the resulting graph is four-connected.

We first observe that if the wheel constraint wc(G) dominates the lower bound, then there exists exactly one wheel w. The wheel w is a star wheel and has a degree-3 center. We also know that 4-blk(G) contains three non-special 4-block leaves and there is no critical F-vertex. The pair of vertices found in step 1 satisfy part (iv) or part (v) of the leaf-connecting condition. Thus step 1 of algorithm `aug3to4` finds the right pair of vertices between which a new edge is added if wc(G) dominates.

If the degree constraint dominates, then there is at least one critical vertex. Steps 2, 3, 4, 8, and 9 make sure that the degree constraint of any critical vertex is reduced by 1 by adding the new edge found. (Note that steps 8, 9, and 10 are in Algorithm 2.) Corollary 6 makes sure the implied path between the pair of vertices found in step 9 passes through all critical vertices, if any. The pair of vertices found in steps 2 and 4 satisfy part (i) of the leaf-connecting condition. The pair of vertices found in steps 3 and 8 satisfy part (iv) of the leaf-connecting condition. The pair of vertices found in step 9 satisfy part (v) of the leaf-connecting condition. Thus we reduce both dc(G) and lc(G) by 1. Hence the lower bound is reduced by 1 by adding an edge.

We now prove the case when the leaf constraint dominates. We have to make sure that the pair of vertices found satisfy the leaf-connecting condition. In the following, we show that in each step, the part of the leaf-connecting condition that is satisfied if the number of 4-block leaves is at least 4. Step 2: part (i); step 3: part (v); step 4: part (i); step 5: part (i); step 6: part (ii); step 7: part (iii); step 8: part (iv) or part (v); step 9: part (v); step 10: part (v). If there are less than three 4-block leaves in G, we can add an edge between demanding vertices of any arbitrary two 4-block leaves. Thus lc(G) is reduced by 1 each time we add an edge. Hence the lower bound is reduced by 1 by adding an edge. □

We now describe an efficient way of implementing algorithm `aug3to4`. The 4-block tree can be computed in $O(n \cdot \alpha(m,n) + m)$ time for a graph with n vertices and m edges [KTDBC91]. We know that the leaf constraint, the degree constraint of any separating triplet and the wheel constraint of any wheel in G can only be decreased by adding an edge. We also know that $\text{lc}(G)$, the sum of degree constraints of all separating triplets, and the sum of wheel constraints of all wheels are all $O(n)$. Thus we can use the technique in [RG77] to maintain the current leaf constraint, the degree constraint of each separating triplet, and the wheel constraint of each wheel in $O(n)$ time for the entire execution of the algorithm. We also visit each vertex and each edge in the 4-block tree a constant number of times before deciding to collapse them. There are $O(n)$ 4-block leaves and $O(n)$ vertices and edges in 4-blk(G). We use a set-union-find algorithm to maintain the identities of vertices after collapsing. Hence the overall time for updating the 4-block tree is $O(n \cdot \alpha(n,n))$. We have the following claim.

Claim 5 *Algorithm* `aug3to4` *can be implemented in* $O(n \cdot \alpha(m,n) + m)$ *time where* n *and* m *are the number of vertices and edges in the input graph, respectively, and* $\alpha(m,n)$ *is the inverse Ackermann function.* □

6 Concluding Remarks

We have given a sequential algorithm for finding a smallest set of edges whose addition four-connects a triconnected graph. The algorithm runs in $O(n \cdot \alpha(m,n) + m)$ time using $O(n+m)$ space. We used the following approach to develop our algorithm. We first gave a 4-block tree data structure for a triconnected graph that is similar to the one given in [KTDBC91]. We then described a lower bound on the smallest number of edges that must be added based on the 4-block tree of the input graph. We further showed that it is possible to decrease this lower bound by 1 by adding an appropriate edge. The lower bound that we gave here is different from the ones that we have for biconnecting a connected graph and for triconnecting a biconnected graph. We also showed relations between these two lower bounds. This new lower bound applies for arbitrary k, and gives a tighter lower bound than the one known earlier for the number of edges needed to k-connect a $(k-1)$-connected graph. It is likely that techniques presented in this paper may be used in finding the k-connectivity augmentation number of a $(k-1)$-connected graph, for an arbitrary k.

Acknowledgments

We would like to thank Vijaya Ramachandran for helpful discussions and comments. We also thank Tibor Jordán, Arkady Kanevsky, and Roberto Tamassia for useful information and comments.

References

[CS89] G.-R. Cai and Y.-G. Sun. The minimum augmentation of any graph to a k-edge-connected graph. *Networks*, 19:151–172, 1989.

[DBT90] G. Di Battista and R. Tamassia. On-line graph algorithms with SPQR-trees. In *Proc. 17th Int'l Conf. on Automata, Language and Programming*, volume LNCS # 443, pages 598–611. Springer-Verlag, 1990.

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

[FBW89] D. Fernández-Baca and M. A. Williams. Augmentation problems on hierarchically defined graphs. In *1989 Workshop on Algorithms and Data Structures*, volume LNCS # 382, pages 563–576. Springer-Verlag, 1989.

[FC70] H. Frank and W. Chou. Connectivity considerations in the design of survivable networks. *IEEE Trans. on Circuit Theory*, CT-17(4):486–490, December 1970.

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

[FJ93] A. Frank and T. Jordán. Minimal edge-coverings of pairs of sets. Manuscript, June 1993.

[Fra92] A. Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM J. Disc. Math.*, 5(1):25–43, February 1992.

[Gab91] H. N. Gabow. Applications of a poset representation to edge connectivity and graph rigidity. In *Proc. 32th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 812–821, 1991.

[Gus87] D. Gusfield. Optimal mixed graph augmentation. *SIAM J. Comput.*, 16(4):599–612, August 1987.

[HR91] T.-s. Hsu and V. Ramachandran. A linear time algorithm for triconnectivity augmentation. In *Proc. 32th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 548–559, 1991.

[HR93] T.-s. Hsu and V. Ramachandran. On finding a smallest augmentation to biconnect a graph. *SIAM J. Compu.*, 22(5):889–912, 1993.

[Hsu93] T.-s. Hsu. *Graph Augmentation and Related Problems: Theory and Practice.* PhD thesis, University of Texas at Austin, 1993.

[JG86] S. P. Jain and K. Gopal. On network augmentation. *IEEE Trans. on Reliability*, R-35(5):541–543, 1986.

[Jor92] T. Jordán, February 1992. private communications.

[Jor93a] T. Jordán. Increasing the vertex-connectivity in directed graphs. In *Proc. 1st Euorpean Symp. on Algorithms*, 1993, to appear.

[Jor93b] T. Jordán. Optimal and almost optimal algorithms for connectivity augmentation problems. In *Proc. 3rd IPCO Conference*, 1993, to appear.

[Kan93] G. Kant. *Algorithms for Drawing Planar Graphs.* PhD thesis, Utrecht University, the Netherlands, 1993.

[KT91] A. Kanevsky and R. Tamassia, October 1991. private communications.

[KT92] S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. In *Proc. 19th Int'l Conf. on Automata, Language and Programming*, volume LNCS #623, pages 330–341. Springer-Verlag, 1992.

[KTDBC91] A. Kanevsky, R. Tamassia, G. Di Battista, and J. Chen. On-line maintenance of the four-connected components of a graph. In *Proc. 32th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 793–801, 1991.

[KU86] Y. Kajitani and S. Ueno. The minimum augmentation of a directed tree to a k-edge-connected directed graph. *Networks*, 16:181–197, 1986.

[MHT87] T. Masuzawa, K. Hagihara, and N. Tokura. An optimal time algorithm for the k-vertex-connectivity unweighted augmentation problem for rooted directed trees. *Discrete Applied Mathematics*, pages 67–105, 1987.

[NGM90] D. Naor, D. Gusfield, and C. Martel. A fast algorithm for optimally increasing the edge-connectivity. In *Proc. 31th Annual IEEE Symp. on Foundations of Comp. Sci.*, pages 698–707, 1990.

[Ple76] J. Plesník. Minimum block containing a given graph. *ARCHIV DER MATHEMATIK*, XXVII:668–672, 1976.

[Ram93] V. Ramachandran. Parallel open ear decomposition with applications to graph biconnectivity and triconnectivity. In J. H. Reif, editor, *Synthesis of Parallel Algorithms*, pages 275–340. Morgan-Kaufmann, 1993.

[RG77] A. Rosenthal and A. Goldner. Smallest augmentations to biconnect a graph. *SIAM J. Comput.*, 6(1):55–66, March 1977.

[Sor88] D. Soroker. Fast parallel strong orientation of mixed graphs and related augmentation problems. *Journal of Algorithms*, 9:205–223, 1988.

[SWK69] K. Steiglitz, P. Weiner, and D. J. Kleitman. The design of minimum-cost survivable networks. *IEEE Trans. on Circuit Theory*, CT-16(4):455–460, 1969.

[Tar83] R. E. Tarjan. *Data Structures and Network Algorithms.* SIAM Press, Philadelphia, PA, 1983.

[UKW88] S. Ueno, Y. Kajitani, and H. Wada. Minimum augmentation of a tree to a k-edge-connected graph. *Networks*, 18:19–25, 1988.

[Wat87] T. Watanabe. An efficient way for edge-connectivity augmentation. Tech. Rep. ACT-76-UILU-ENG-87-2221, Coordinated Science lab., University of Illinois, Urbana, IL, 1987.

[WHN90] T. Watanabe, Y. Higashi, and A. Nakamura. Graph augmentation problems for a specified set of vertices. In *Proc. 1st Annual Int'l Symp. on Algorithms*, volume LNCS #450, pages 378–387. Springer-Verlag, 1990. Earlier version in Proc. 1990 Int'l Symp. on Circuits and Systems, pages 2861–2864.

[WN87] T. Watanabe and A. Nakamura. Edge-connectivity augmentation problems. *J. Comp. System Sci.*, 35:96–144, 1987.

[WN88] T. Watanabe and A. Nakamura. 3-connectivity augmentation problems. In *Proc. of 1988 IEEE Int'l Symp. on Circuits and Systems*, pages 1847–1850, 1988.

[WN90] T. Watanabe and A. Nakamura. A smallest augmentation to 3-connect a graph. *Discrete Applied Mathematics*, 28:183–186, 1990.

[WN93] T. Watanabe and A. Nakamura. A minimum 3-connectivity augmentation of a graph. *J. Comp. System Sci.*, 46:91–128, 1993.

[WNN89] T. Watanabe, T. Narita, and A. Nakamura. 3-edge-connectivity augmentation problems. In *Proc. of 1989 IEEE Int'l Symp. on Circuits and Systems*, pages 335–338, 1989.

[WYO91] T. Watanabe, M. Yamakado, and K. Onaga. A linear time augmenting algorithm for 3-edge-connectivity augmentation problems. In *Proc. of 1991 IEEE Int'l Symp. on Circuits and Systems*, pages 1168–1171, 1991.