

NC Approximation Algorithms for 2-Connectivity Augmentation in a Graph ^{*}

Weifa Liang^{†‡} George Havas[†]

[†] School of Information Technology

The University of Queensland, Brisbane, QLD 4072, Australia

[‡] Department of Mathematics

The University of Queensland, Brisbane, QLD 4072, Australia

Abstract. Given an undirected graph $G = (V, E_0)$ with $|V| = n$, and a feasible set E of m weighted edges on V , the optimal 2-edge (2-vertex) connectivity augmentation problem is to find a subset $S^* \subseteq E$ such that $G(V, E_0 \cup S^*)$ is 2-edge (2-vertex) connected and the weighted sum of edges in S^* is minimized. We devise NC approximation algorithms for the optimal 2-edge connectivity and the optimal 2-vertex connectivity augmentation problems by delivering solutions within $(1 + \ln n_c)(1 + \epsilon)$ times optimum and within $(1 + \ln n_b)(1 + \epsilon) \log n_b$ times optimum when G is connected, respectively, where n_c is the number of 2-edge connected components of G , n_b is the number of biconnected components of G , and ϵ is a constant with $0 < \epsilon < 1$. Consequently, we find an approximation solution for the problem of the minimum 2-edge (biconnected) spanning subgraph on a weighted 2-edge connected (biconnected) graph in the same time and processor bounds.

1 Introduction

Augmenting the connectivity of communication networks is increasingly becoming important to provide reliable means of communication. In the following the *k-connectivity* of a graph refers to either *k-edge connectivity* or *k-vertex connectivity*. A graph is *k-edge (k-vertex) connected* if there are k edge-disjoint (vertex-disjoint) paths joining each pair of vertices in it. A 2-edge connected graph is called *bridge-connected* graph, and a 2-vertex connected graph is called *biconnected*. Given an undirected graph $G = (V, E_0)$ with $|V| = n$, and a feasible set E of m weighted edges on V such that $G(V, E_0 \cup E)$ is k -edge (k -vertex) connected, the *optimal k-connectivity augmentation problem* of $G = (V, E_0)$ is to find a subset $S^* \subseteq E$ such that $G(V, E_0 \cup S^*)$ is k -edge (k -vertex) connected and the weighted sum of edges in S^* is minimized. If the edges in the feasible set $E = E(K_n) - E_0$ are unweighted, where $E(K_n)$ is the edge set of the complete graph K_n on the vertex set V , it is already known that, for any $k < n$, the exact solution for the optimal k -edge connectivity augmentation problem can be obtained in polynomial time [5, 8, 12, 13]. However, when the

^{*} Part of this work was done when the first author was associated with the Dept. of Computer Science, Australian National University, Canberra, ACT 0200, Australia

edges in E are weighted, the situation is very different. In this case, we cannot expect to find an exact solution S^* for the optimal k -connectivity augmentation problem in polynomial time even for $k = 2$. Eswaran and Tarjan [3] first showed that if $G = (V, E_0)$ is disconnected, the optimal 2-connectivity augmentation problem is NP-complete. Frederickson and Jájá [4] further showed that even if $G = (V, E_0)$ is connected, this problem is still NP-complete [4]. Instead, Frederickson and Jájá [4] presented an $O(n^2)$ time approximation algorithm for the optimal 2-connectivity augmentation problem, and the solution delivered by their algorithm is not worse than twice the optimum if G is connected or 3 times optimum otherwise. Recently Khuller and Thurimella [7] presented another simple algorithm for this problem. Their algorithm requires $O(m + n \log n)$ time, and the solution delivered is also 2 or 3 times optimum depending on whether G is connected or disconnected.

One closely related problem is to find a minimum k -edge (k -vertex) connected spanning subgraph in a k -edge (k -vertex) weighted connected graph. This problem can be stated as follows. Given a k -edge (k -vertex) weighted connected graph $G(V, E)$ with $k > 1$, find a k -edge (k -vertex) connected spanning subgraph $G_1 = (V, E_1)$ such that G_1 has the minimum weighted sum of edges, where $E_1 \subseteq E$. This problem is a special case of the augmentation problem with $E_0 = \emptyset$. It is also NP-complete.

We focus on the optimal 2-connectivity augmentation problem by presenting parallel approximation algorithms for it. Our approach is to reduce this problem to the *minimum weighted set cover* (MWSC) problem. Our contributions include (i) an NC approximation algorithm for the optimal 2-edge connectivity augmentation problem which delivers a solution within either $(1 + \ln n_c)(1 + \epsilon)$ times optimum if G is connected, or $(1 + \ln n_c)(1 + \epsilon) + 1$ times optimum otherwise; and (ii) an NC approximation algorithm for the optimal biconnectivity augmentation problem which delivers a solution within either $(1 + \ln n_b)(1 + \epsilon) \log n_b$ times optimum if G is connected, or within $(1 + \ln n_b)(1 + \epsilon) \log n_b + 1$ times optimum otherwise, where n_c and n_b are the number of 2-edge connected components and biconnected components of $G(V, E_0)$ respectively, and ϵ is a constant with $0 < \epsilon < 1$.

2 Preliminaries

A vertex in a graph is an *articulation point* if the deletion of the vertex leaves the graph disconnected. An edge in a graph is a *bridge* if the deletion of the edge leaves the graph disconnected. Let $K = (V_K, E_K)$ be an undirected simple graph. A vertex v *dominates* a vertex u on K if and only if $(u, v) \in E_K$. If there are two vertex disjoint sets $\mathcal{A}$ and $\mathcal{B}$ of V_K , we say $\mathcal{A}$ *dominates* $\mathcal{B}$ if, for every vertex $u \in \mathcal{B}$, there is a vertex $v \in \mathcal{A}$ such that u is dominated by v . Let $T(V, E_T)$ be a rooted tree and $Z \subset V$ with $Z \neq \emptyset$. The vertex $LCA(Z)$ of T is defined as follows: if $Z = \{v\}$, then $LCA(Z) := v$; if $Z = \{u, v\}$, then $LCA(Z)$ is the vertex which is the lowest common ancestor of u and v in T ; otherwise, $LCA(Z) := LCA(Z - \{x, y\} \cup \{LCA(x, y)\})$. Note that $LCA(Z)$ is well defined

and is a unique vertex of T for a given Z . An *inverted tree* $T(V, E_T)$ is a directed tree rooted at a specified vertex $r \in V$ such that for each vertex v ($v \neq r$) there is a pointer pointing to v 's parent $F_T(v)$, directed edge $\langle v, F_T(v) \rangle \in E_T$, and $F_T(r) = r$. Given a set system $\mathcal{A} \subseteq 2^X$ and a weight function $w : \mathcal{A} \rightarrow \mathbf{R}$, the *minimum weighted set cover* problem consists of finding a minimum subcollection $\mathcal{A}' \subseteq \mathcal{A}$ such that $\bigcup \mathcal{A}' = X$, which is NP-complete [6].

3 2-Edge Connectivity Augmentation

Let $G = (V, E_0)$ be connected, and E be a feasible set with m weighted edges such that $G(V, E_0 \cup E)$ is 2-edge connected. We only need to show how to increase the edge connectivity of a tree due to the following facts. If G has nontrivial 2-edge connected components (2ECCs), then we contract the vertex sets of these components into single vertices, resulting in a tree whose edges are the bridges of $G(V, E_0)$. Let $E' \subseteq E$ be an edge set such that the edges in E to be kept in E' are the minimum edges that connect the vertices of different 2ECCs of $G(V, E_0)$. For convenience later, E' is also referred to as “superimposing” E on T . It is easy to show that the computation of E' can be finished in $O(\log n)$ time using $O(m)$ processors on a CREW PRAM provided all 2ECCs of G are given. From now on, we assume that the initial graph is a tree T rooted at r with n_c vertices where r is a degree-one vertex and n_c is the number of 2ECCs of G . A bipartite graph $B(V_1, V_2, E_b)$ is constructed as follows. V_1 is the set of all edges in E' , and V_2 is the edge set of T . There is an edge $(e_1, e_2) \in E_b$ and $e_i \in V_i$, $i = 1, 2$, if, on adding e_1 to T , e_2 is on the cycle consisting of tree edges and e_1 . That is, e_2 is no longer a bridge after adding e_1 to T .

Lemma 1. *The bipartite graph $B(V_1, V_2, E_b)$ defined above can be constructed in $O(mn_c)$ time, where $|V_1| \leq m - n_c + 1$, $|V_2| \leq n_c$, and the weight of each vertex in V_1 is the weight of the corresponding edge of G .*

Proof. We first select a degree-one vertex as the root of T , then traverse T , assigning each vertex v a pre-order numbering $pre(v)$ and the number of descendants (including itself) $nd(v)$ of v . This assignment can be done in $O(\log n)$ time using $O(n)$ processors on an EREW PRAM. The construction of B is as follows. Consider a non-tree edge $e_1 = (x, y)$ in V_1 and a tree edge $e_2 = (u, v)$ in V_2 . If u is the parent of v in T , there is an edge connecting vertices e_1 and e_2 in B if one of the following two conditions holds: (i) $pre(v) \leq pre(x) < pre(v) + nd(v)$, and either $pre(y) < pre(v)$ or $pre(y) \geq pre(v) + nd(v)$; (ii) $pre(v) \leq pre(y) < pre(v) + nd(v)$, and either $pre(x) < pre(v)$ or $pre(x) \geq pre(v) + nd(v)$. Therefore B can be constructed in $O(mn_c)$ time provided E' , T , and the pre-order numbering and the number of descendants for each vertex in T are given. $\square$

Lemma 2. *Let $G(V, E)$ be a connected undirected graph, and $T(V, E_T)$ be a spanning tree of G . Then G is 2-edge connected if and only if $V_1 (= E - E_T)$ dominates $V_2 = E_T$ in B , where the graph $B(V_1, V_2, E_b)$ induced by the tree T and the edge set $E - E_T$ is defined as above.*

Proof. If G is 2-edge connected, V_1 must dominate V_2 in B . Assume that V_1 does not dominate V_2 . Then there exists a vertex $e_2 \in V_2$ which is not dominated by any vertex in V_1 . This means that e_2 is not in any simple cycle formed by the tree edges and the non-edge tree edges, which is a contradiction.

If V_1 dominates V_2 in B , each edge in T is included in a simple cycle at least. This means that the remaining graph is still connected after deleting any edge from G , i.e., there is no bridge in G . $\square$

Now, for any subset $S \subseteq V_1$, if S dominates V_2 in B , then the edge set corresponding to S is a 2-edge connectivity augmentation of G . Let $w(S)$ be the weighted sum of the vertices in S . If such a $S^* \subseteq V_1$ with minimal $w(S^*)$ can be found, then S^* is a solution of the problem. For simplicity of expression later, S^* is called a *minimum dominator set on B* . Thus, the optimal 2-edge connectivity augmentation problem becomes to find S^* , while the problem of finding S^* is equivalent to an MWSC problem. Let $X = V_2$. For each vertex $v \in V_1$, there is a corresponding set $\mathcal{A}_v = \{u : (u, v) \in E_b, v \in V_1, u \in V_2\}$. The weight of $\mathcal{A}_v$ is the weight of the corresponding edge of v . Then to find S^* on B becomes to find a subcollection of sets $\mathcal{A}_v$ such that $\cup \mathcal{A}_v = X$ and the weighted sum of these sets is minimized. For this latter problem, Berger et al. [1] have the following theorem.

Theorem 3. [1] *Let $H = (V, E)$ be a hypergraph with $|V| = n'$ and $|E| = m'$. For any $0 < \epsilon < 1$, there is an NC algorithm for the minimum set cover problem that uses $O(m' + n')$ processors, runs in $O(\log^4 n' \log m' \log^2(n'm')/\epsilon^6)$ time, and produces a cover of weight at most $(1 + \epsilon)(1 + \ln \Delta)\tau^*$, where Δ is the maximum vertex degree and τ^* is the optimal solution.* $\square$

Recall that our approximation algorithm for the optimal 2-edge connectivity augmentation problem consists of three stages. In the first stage it generates a 2ECC tree T if G is connected. Otherwise, adding the edges in the feasible set E' yields a minimum spanning tree (MST), and adding the tree edges into G produces T . In the second stage, it constructs a bipartite graph B . In the third stage it finds an approximate solution for the minimum dominator set on B . Now we give the parallel implementation details for these three stages. First, we show how to construct the 2ECC tree T . Given a graph $G = (V, E_0)$, finding all 2ECCs and the bridges of G can be done by applying the biconnectivity algorithm of Tarjan and Vishkin [11]. That is, after finding all *biconnected components* (2VCCs), we identify those 2VCCs consisting of one edge only which are bridges of G , compute all *connected components* (CCs) of the remaining graph by deleting all bridges from G , construct a tree T in which the vertices are those CCs, and the edges are those bridges. If G is disconnected, we obtain a forest F rather than a tree T . We then add the edges in E' to G , produce an MST by the algorithm of Chin et al. [2], and yield T by adding some of the edges of the MST to F .

The construction of B is straightforward. We only need to test the two conditions in the proof of Lemma 1. This can be done easily given the tree T' and the pre-order numbering of vertices in T . Note that the degree of B is n_c . Having

the graph B , we obtain an approximation solution for the minimum dominating set S^* on B by applying the algorithm in [1].

In case $G = (V, E_0)$ is disconnected, find an MST of $G(V, E_0 \cup E)$ by assigning the edges in E_0 with weights 0 and the edges in E with their original weights, add the edges of the MST to $G(V, E_0)$, and generate T defined as before. For this latter case we show that this leads to an approximation solution within $(1 + \ln n_c)(1 + \epsilon) + 1$ times optimum, where n_c is the number of 2ECCs of $G(V, E_0)$ and ϵ is a small constant with $0 < \epsilon < 1$. Let G^* be a minimum 2-edge connected graph produced by optimal augmentation to G , and let $w(G^*)$ be the associated weight of G^* . The proof proceeds as follows. We add all superimposing edges of G^* on T , then the edges in $G^* - T$ form a dominating set on B because G^* is 2-edge connected by Lemma 2. Therefore, the set of all edges in $G^* - T$ dominates the edge set of T . Let $w(T^*)$ be the minimum 2-edge augmentation on T such that the resulting graph is 2-edge connected. Then $w(T^*) \leq w(G^* - T) \leq w(G^*)$. Meanwhile, $w(T) \leq w(G^*)$ because the MST of G is a minimum connected spanning graph. In summary, we have the following theorem.

Theorem 4. *Given a weighted graph $G = (V, E_0)$ and a feasible set E , there exists an NC approximation algorithm for the optimal 2-edge connectivity augmentation problem which delivers a solution within either $(1 + \ln n_c)(1 + \epsilon)$ times optimum if G is connected or $(1 + \ln n_c)(1 + \epsilon) + 1$ times optimum otherwise. The algorithm requires $O(\log^7 n / \epsilon^6)$ time and $O(mn_c)$ processors on a CRCW PRAM, where n_c is the number of 2ECCs of G and ϵ is a constant with $0 < \epsilon < 1$.*

Proof. Now we analyze the computational complexity of the proposed NC approximation algorithm. The 2ECC tree T can be constructed in $O(\log n)$ time using $O(m + n)$ processors on a CRCW PRAM by the biconnectivity algorithm of Tarjan and Vishkin. The construction of tree T' and the assignment of the pre-ordering numbering to the vertices in T can be done in $O(\log n)$ time using $O(n)$ processors by Schieber and Vishkin's algorithm [10]. The graph B can be constructed in $O(1)$ time using $O(mn_c)$ processors on a CREW PRAM. Finding an approximation solution for the MWSC problem induced by B can be done in $O(\log^7 n / \epsilon^6)$ time using $O(mn_c)$ processors on a CRCW PRAM because $|E_b| \leq mn_c$. The solution generated is within $(1 + \ln n_c)(1 + \epsilon)$ times optimum by Theorem 3, where n_c is the number of 2ECCs of $G(V, E_0)$ and ϵ is a constant with $0 < \epsilon < 1$. $\square$

Corollary 5. *Given a weighted 2-edge connected graph $G(V, E)$, finding a 2-edge connected spanning subgraph whose weight is $(1 + \ln n)(1 + \epsilon) + 1$ times the weight of the minimum 2-edge connected spanning subgraph can be done in $O(\log^7 n / \epsilon^6)$ time using $O(mn)$ processors on a CRCW PRAM, where ϵ is a constant and $0 < \epsilon < 1$.*

4 Biconnectivity Augmentation

Assume that $G = (V, E_0)$ is connected. Our strategy for this problem is similar to the one used in the previous section. That is, first obtain a block tree T of

the biconnected components (2VCCs) of G , which is defined as follows. The vertex set of T is $V_a \cup V_b$, where V_a is the set by all articulation points of G , and V_b is the set by all 2VCCs of G . The edge set $E(T)$ of T consists of edge (a_i, b_j) , where $a_i \in V_a$, $b_j \in V_b$, and a_i is included in b_j . In the following, by *superimposing* an edge $(x, y) \in E$ on T , we mean adding an edge between a_i and b_j , where x is either an articulation point ($x = a_i$) or x is included in the 2VCC a_i , and y is either an articulation point ($y = b_j$) or y is included in the 2VCC b_j . If there are multiple edges between two vertices in T , we just keep the edge with the minimum weight, and remove all the other edges. Let the remaining edge set be E' , then $|E'| \leq |E| \leq m$. In the rest we only consider adding some edges in E' to make G biconnected. Then the construction of the bipartite graph $B(V_1, V_2, E_b)$ is as follows. V_1 is the set of the edges in E' , and V_2 is the set of the 2VCCs of G . There is an edge $(v_1, v_2) \in E_b$ and $v_i \in V_i$, $i = 1, 2$, if adding the corresponding edge $e = (x, y)$ of v_1 to T , v_2 is in the cycle consisting of the tree edges and e . Third, we find an approximation solution S'_i of the MWSC problem induced by B_i , where B_i is obtained from B_{i-1} and $S = \cup_{j=1}^{i-1} S'_j$, $0 \leq i \leq \lceil \log |V_2| \rceil - 1$. Initially $B_0 = B$ and $S = \emptyset$. Let E'' be the corresponding edge set of $\cup_{i=0}^{\lceil \log |V_2| \rceil - 1} S'_i$. Finally adding all edges in E'' to G makes it biconnected.

Lemma 6. *Given the block tree T and E' , the graph $B(V_1, V_2, E_b)$ can be constructed in $O(mn_b)$ time, where $\sum_{i=1,2} |V_i| \leq m + n_b$, $|V_2| \leq n_b$.*

Proof. Given T , construct an auxiliary tree T' such that the LCA query of two vertices in T can be answered in $O(1)$ time. The construction of T' can be done in $O(n)$ time by Schieber and Vishkin's algorithm [10]. Now we construct the graph B as follows. Let the corresponding edge of vertex $v_1 \in V_1$ be $e = (x, y)$, and $t = LCA(\{x, y\})$ be the lowest common ancestor of x and y in T . Then there is an edge between v_1 and $v_2 \in V_2$ if either one of the following conditions holds: (i) $LCA(\{x, v_2\}) = x$ and $LCA(\{v_2, y\}) = v_2$ when $t = x$; (ii) $LCA(\{x, v_2\}) = v_2$ and $LCA(\{v_2, y\}) = y$ when $t = y$; (iii) either $LCA(\{t, v_2\}) = t$ and $LCA(\{v_2, x\}) = v_2$, or $LCA(\{t, v_2\}) = t$ and $LCA(\{v_2, y\}) = v_2$ when $t \neq x$ and $t \neq y$. Obviously B can be obtained in $O(|E_b|) = O(mn_b)$ time. $\square$

Denote by $G_B[X \cup Y]$ a subgraph of $B(V_1, V_2, E_b)$ consisting of the vertices in $X \cup Y$ and the edges between these vertices, where $X \cup Y \subseteq V_1 \cup V_2$. Then we have the following lemma which is very important to construct our algorithm.

Lemma 7. *Let a subset $S \subseteq V_1$ dominate the set V_2 . Then the graph formed by adding the corresponding edges of vertices in S to G is biconnected if and only if $G_B[S \cup V_2]$ is connected.*

Proof. Let $S \subseteq V_1$ and S dominate V_2 . Suppose G is not biconnected. We first show that if $G_B[S \cup V_2]$ is *disconnected*, $G(V, E_0 \cup S)$ is not biconnected. We then show that if $G_B[S \cup V_2]$ is *connected*, the graph formed by adding the edges in S to G is biconnected.

Assume that $G_B[S \cup V_2]$ is disconnected, and has k CCs with $k > 1$. Let A and B be two CCs among these k CCs, and $V(A)$ and $V(B)$ be the vertex sets of A and B respectively. Let $b(A) = V(A) \cap V_2$ and $b(B) = V(B) \cap V_2$. Denote by $\alpha = LCA(b(A))$ and $\beta = LCA(b(B))$ on T . Then there exists a unique path $\pi_{\alpha\beta}$ between α and β on T . Note that it is possible that $\pi_{\alpha\beta}$ consists of one vertex only. We further assume that $\pi_{\alpha\beta}$ contains no vertices belonging to other CCs except A and B . The problem now is divided into the following three cases: (i) $\alpha \neq \beta$ and neither one is the ancestor of another in T . Then $\pi_{\alpha\beta}$ contains more than one vertex, and at least one vertex v among these vertices is an articulation point of G by the property of T . So, deleting v will leave the vertices in $b(A)$ and the vertices in $b(B)$ in different CCs. Therefore, v is still an articulation point of $G(V, E_0 \cup S)$. (ii) $\alpha = \beta$. In this case we further classify whether α is an articulation point of G . If it is, then deletion of α will leave the vertices in $b(A)$ and the vertices in $b(B)$ in different CCs. Therefore, α is still an articulation point of $G(V, E_0 \cup S)$. Otherwise, α is a 2VCC vertex, which is impossible. If α is a 2VCC vertex, it must be included in $V(A)$. For the same reason, it must be included in $V(B)$ also, then A and B should be the same CC rather than two distinguished CCs, contradicting our initial assumption. Therefore, α is not a 2VCC vertex. (iii) $\alpha \neq \beta$, and one is the ancestor of another in T . Assume that β is the ancestor of α . Let T_α be a subtree of T rooted at α including all vertices in $b(A)$. By the same argument as case (ii), we can show that α is an articulation point of G only. Meanwhile, we also note that there are not any edges between a vertex other than α in T_α and a vertex in $V_a \cup V_b - V(T_\alpha)$ except the edges incident to α , which means that the deletion of α will leave the vertices in T_α and the other vertices of T separated. Therefore, $G(V, E_0 \cup S)$ is not biconnected.

Now we show the second part. Our approach is to show that every articulation point of G is no longer an articulation point of the resulting graph after adding the edges in S to G . Let v be an arbitrary articulation point of G , and v be contained in l 2VCCs $b_1, b_2, \dots, b_l$. Then v is an adjacent vertex of these l vertices in T . We need to prove that, if $G_B[S \cup V_2]$ is connected, then all 2VCCs sharing v should become a 2VCC of $G(V, E_0 \cup S)$. We start by finding all shortest paths between b_1 and b_j in $G_B[S \cup V_2]$, where $2 \leq j \leq l$. Note that these paths definitely exist in $G_B[S \cup V_2]$ because it is connected. Let the shortest path between b_1 and b_k , denoted by P_{b_1, b_k} , be the shortest among these $l-1$ shortest paths, $2 \leq k \leq l$. Assuming that the vertex sequence of P_{b_1, b_k} is $b_1, e_1, c_1, e_2, c_2, \dots, e_p, b_k, e_i \in V_1, c_j \in V_2$, where $1 \leq i \leq p, 1 \leq j \leq p-1$, and P_{b_1, b_k} does not contain any other b_j for $j \neq k$. If $|P_{b_1, b_k}| = 1$, by the definition of B , b_1 and b_k are on the cycle of tree edges of T and the edge e_1 . We merge all 2VCCs on this cycle into a 2VCC. As a result, b_1 and b_k are merged into a 2VCC b' . Now v is still an articulation point of the augmented graph shared by $l-1$ 2VCCs $b', b_2, \dots, b_{k-1}, b_{k+1}, \dots, b_l$. We follow the method above and continue merging. Finally all initial 2VCCs sharing v are merged into one 2VCC, and v is no longer an articulation point of $G(V, E_0 \cup S)$. If $|P_{b_1, b_k}| = p$ and $p > 1$, then all vertices b_j for $j \neq 1$ and $j \neq k$ do not appear on this path. By induction on p , it is easy to prove that all 2VCCs on this path can be merged into one 2VCC. That means, after merging all 2VCCs

on P_{b_1, b_k} , b_1 and b_k are merged into a 2VCC b' , and v now is an articulation point shared by $l - 1$ 2VCCs. We apply the method above again to merge all the remaining 2VCCs sharing v . As a result, all b_i for $1 \leq i \leq l$ are merged into a 2VCC, and v is no longer an articulation point of $G(V, E_0 \cup S)$. $\square$

Having the lemma above, we now assign to each vertex in V_1 the corresponding edge's weight. Let S^* be a S defined above with the minimum weighted sum. Then the remaining task is to find such a S^* . Obviously this is an NP-complete problem again. Instead we look for an approximation solution for it. The basic idea of our approximation solution is to reduce this problem to a series of MWSC problems induced by $B_i(V_1^{(i)}, V_2^{(i)}, E_b^{(i)})$, $0 \leq i \leq \lceil \log |V_2| \rceil - 1$. The bipartite graph $B_i(V_1^{(i)}, V_2^{(i)}, E_b^{(i)})$ is constructed as follows. Given B_{i-1} and a set $S \subseteq V_1$, Initially $B_0(V_1^{(0)}, V_2^{(0)}, E_b^{(0)}) := B(V_1, V_2, E_b)$ and $S := \emptyset$. we compute all CCs of $G_B[S \cup V_2]$ first. Then a vertex $v \in V_1^{(i-1)}$ is included in $V_1^{(i)}$ if and only if there exists at least two edges $(v, x), (v, y) \in E_b^{(i-1)}$ such that x and y are in different CCs of $G_B[S \cup V_2]$. $V_2^{(i)}$ is the set consisting of all CCs of $G_B[S \cup V_2]$. The edge set $E_b^{(i)}$ includes all edges (v, c) and (v, d) , where c is the CC containing x , d is the CC containing y , $c \neq d$, and $(v, x), (v, y) \in E_b^{(i-1)}$. If there is more than one edge between two vertices in B_i , we delete all duplicate edges between them but one. An approximation algorithm for finding S in Lemma 7 is as follows.

$S := \emptyset$; $V_1^{(0)} := V_1$; $V_2^{(0)} := V_2$; $E_b^{(0)} := E_b$;
 $B_0 := B(V_1^{(0)}, V_2^{(0)}, E_b^{(0)})$; $i := 0$;
While $G_B[S \cup V_2]$ is disconnected **do**
 Find the minimum dominator set S_i which dominates $V_2^{(i)}$ in B_i ;
 $S := S \cup S_i$;
 Compute all connected components of graph $G_B[S \cup V_2]$;
 Construct the bipartite graph B_{i+1} ;
 $i := i + 1$

Endwhile.

Note that S_i in the algorithm cannot be obtained in polynomial time unless $P=NP$. However an approximation solution for S_i can be found by solving an MWSC problem induced on B_i . Let S'_i be an approximation solution of S_i by the algorithm due to Berger et al. [1], then this solution is $(1 + \ln n_b)(1 + \epsilon)$ times optimum where n_b is the number of 2VCCs of $G(V, E_0)$ and ϵ is a constant with $0 < \epsilon < 1$. Therefore, we have the following lemma.

Lemma 8. *Given B is defined as above, let $S' \subseteq V_1$ dominate V_2 and $G_B[S' \cup V_2]$ be connected. Then we can find an approximation solution S' which is $(1 + \ln n_b)(1 + \epsilon) \log n_b$ times optimum, where n_b is the number of 2VCCs in a connected graph $G(V, E_0)$.*

Proof. Assume that $S^* \subseteq V_1$ has the minimum weighted sum such that S^* dominates V_2 , and $G_B[S^* \cup V_2]$ is connected. From the algorithm above, it is obvious that $w(S_i) \leq w(S^*)$ because S_i is such a vertex set with the minimum weighted

sum that dominates V_2 , while the vertex set S^* , in addition to satisfying all properties of S_i , has an additional restriction that $G_B[S^* \cup V_2]$ is connected. The other important observation is that the edges incident to each vertex in V_1 connect at least two vertices in V_2 , therefore, the number of vertices in V_2 is reduced by at least one half from B_i to B_{i+1} . However $|V_2| \leq n_b$ initially. Thus after $\lceil \log n_b \rceil$ times repetitions of the **while** loop, all vertices in V_2 are merged into the same CC. The approximation solution obtained has weight $\sum_{i=0}^{i=\lceil \log |V_2| \rceil - 1} w(S'_i) \leq \sum_{i=0}^{i=\lceil \log |V_2| \rceil - 1} (1 + \ln n_b)(1 + \epsilon)w(S_i) \leq \lceil \log n_b \rceil (1 + \ln n_b)(1 + \epsilon) \max\{w(S_i)\} \leq \lceil \log n_b \rceil (1 + \ln n_b)(1 + \epsilon)w(S^*)$. $\square$

Now we present the parallel implementation details for the optimal biconnectivity augmentation problem. The approach adopted is similar to that for the optimal 2-edge connectivity augmentation problem. The block tree T is constructed as follows. Apply the biconnectivity algorithm of Tarjan and Vishkin [11] to find all 2VCCs of G , and identify all articulation points of G . Note that a vertex is an articulation point if it appears in more than one 2VCC. After that, construct an adjacency matrix of T , and run the algorithm for computing the CCs of T due to Chin et al. [2] to establish the inverted tree T .

Lemma 9. *The block tree T (stored as an inverted tree) can be constructed in $O(\log^2 n)$ time using $O(n^2/\log n)$ processors on a CRCW PRAM.*

Proof. The algorithm for finding all biconnected components requires $O(\log n)$ time and $O(m' + n)$ processors if G has m' edges and n vertices on a CRCW PRAM [11]. The adjacency matrix of T can be constructed in $O(\log n)$ time using $O(n^2)$ processors on a CREW PRAM. The inverted tree T can be obtained in $O(\log^2 n)$ time using $O(n^2/\log n)$ processors on a CREW PRAM. $\square$

The feasible set E' can be generated in $O(\log n)$ time using $|E| \leq n^2$ processors on a CRCW PRAM. The details are as follows: assign to the endpoints of every edge in E their labels (articulation points or 2VCC identifications) in T ; sort these edges by their endpoint labels as the first key and by their associated weights as the second key; delete all other edges with the same labels but keep one with the minimum weight by applying prefix computation. So, the total computation can be finished in $O(\log n)$ time using $O(n^2)$ processors on a CRCW PRAM. The computation of CCs of $G_B[S \cup V_2]$ can be done by Chin et al's [2] algorithm which requires $O(\log^2 n)$ time and $O(mn_b)$ processors.

Lemma 10. *Given T and T' and feasible set E' , the graph B can be constructed in $O(1)$ time using $O(mn_b)$ processors on an CREW PRAM where n_b is the number of 2VCCs in G .*

Proof. The vital step in the construction of B is to test the three conditions in the proof of Lemma 6, which can be done in $O(1)$ time provided that E' , T , and T' are given. Therefore the construction of B requires $O(1)$ time and $O(|E_b|) = O(mn_b)$ processors on a CREW PRAM. $\square$

It remains to find an approximation solution $S' \subseteq V_1$ of B such that (i) S' dominates V_2 ; (ii) $G_B[S' \cup V_2]$ is connected; and (iii) $w(S') \leq \lceil \log n_b \rceil (1 + \ln n_b)(1 + \epsilon)w(S^*)$, where n_b is the number of 2VCCs in $G(V, E_0)$ and ϵ is a small constant with $0 < \epsilon < 1$. This S' can be achieved by Lemma 8. Therefore, we have the following theorem.

Theorem 11. *Given a weighted graph $G = (V, E_0)$ and a feasible set E' , there exists an NC approximation algorithm for the optimal biconnectivity augmentation problem which requires $O(\log^7 n \log n_b / \epsilon^6)$ time and $O(mn_b)$ processors on a CRCW PRAM. The solution delivered is either $(1 + \ln n_b)(1 + \epsilon) \log n_b$ times optimum if G is connected, or $(1 + \ln n_b)(1 + \epsilon) \log n_b + 1$ times optimum, where n_b is the number of 2VCCs in G and ϵ is a constant with $0 < \epsilon < 1$.*

Corollary 12. *Given a weighted biconnected graph $G(V, E)$, finding a biconnected spanning subgraph whose weight is $(1 + \ln n)(1 + \epsilon) \log n + 1$ times the weight of the minimum biconnected spanning subgraph can be done in $O(\log^8 n / \epsilon^6)$ time using $O(mn)$ processors on a CRCW PRAM, where ϵ is a constant with $0 < \epsilon < 1$.*

References

1. B. Berger, J. Rompel and P. W. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. *J. Comp. and Sys. Sci.*, 49, 1994, 454-477.
2. F. Y. Chin, J. Lam and I.-N. Chen. Efficient parallel algorithms for some graph problems. *Comm. ACM.*, 25, 1982, 659-665.
3. K. P. Eswaran and R. E. Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4), 1976, 653-665.
4. G. N. Frederickson and J. Jájá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2), 1981, 270-283.
5. H. N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *Proc. 32nd Annual ACM Symp. on Theory of Comp.*, 1991, 112-122.
6. R. Karp. Reducibility among combinatorial problems. *Complexity and Computer Computations*, R.E. Miller and J. W. Thatcher, Eds, Plenum, NY, 1975.
7. S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. *J. Algorithms*, 14, 1993, 214-225.
8. D. Naor, D. Gusfield and C. Martel. A fast algorithm for optimally increasing the edge-connectivity. *Proc. 31st Ann. IEEE Symp. on Found. of Comp. Sci.*, 1990, 698-707.
9. A. Rosenthal and A. Goldner. Smallest augmentations to biconnect a graph. *SIAM J. Comput.*, 6(1), 1977, 55-66.
10. B. Schieber and U. Vishkin. On finding lowest common ancestors: simplification and parallelization. *SIAM J. Comput.*, 17(6), 1988, 1253-1262.
11. R. E. Tarjan and U. Vishkin. An efficient parallel biconnectivity algorithm. *SIAM J. Comput.*, 14, 1985, 862-874.
12. T. Watanabe and A. Nakamura. Edge-connectivity augmentation problem. *J. Comput. Syst. Sci.*, 35(1), 1987, 96-144.
13. T. Watanabe and A. Nakamura. A minimum 3-connectivity augmentation of a graph. *J. Comput. Syst. Sci.*, 46, 1993, 91-128.