Optimal Bi-Level Augmentation for Selectively Enhancing Graph Connectivity with Applications*

Tsan-sheng Hsu† and Ming-Yang Kao‡

Abstract

Our main problem is abstracted from several optimization problems for protecting information in cross tabulated tables and for improving the reliability of communication networks. Given an undirected graph G and two vertex subsets H_1 and H_2, the *smallest bi-level augmentation problem* is that of adding to G the smallest number of edges such that G contains two internally vertex-disjoint paths between every pair of vertices in H_1 and two edge-disjoint paths between every pair of vertices in H_2. We give a data structure to represent essential connectivity information of H_1 and H_2 simultaneously. Using this data structure, we solve the bi-level augmentation problem in $O(n+m)$ time, where n and m are the numbers of vertices and edges in G. Our algorithm can be parallelized to run in $O(\log^2 n)$ time using $n+m$ processors on an EREW PRAM. By properly setting G, H_1 and H_2, our augmentation algorithm also subsumes several existing optimal algorithms for graph augmentation.

1 Introduction

The problem of adding the minimum number of edges to make a graph satisfy a given connectivity requirement is called the *smallest augmentation problem*. This is a fundamental problem in graph theory [2, 3] and has applications on improving network reliability [5, 13, 24], on protecting sensitive information in cross tabulated tables [7, 17, 16, 20], and on

*An extended abstract of this paper appears in *Proceedings of the Second International Conference on Computing and Combinatorics*, Springer-Verlag, LNCS#1090, pp. 169–178, 1996.

†Institute of Information Science, Academia Sinica, Nankang 11529, Taipei, Taiwan, ROC. E-mail: tshsu@iis.sinica.edu.tw. Research supported in part by NSC Grant 85-2213-E-001-003.

‡Department of Computer Science, Duke University, Durham, North Carolina 27706, USA. E-mail: kao@cs.duke.edu. Research supported in part by NSF Grant CCR-9101385.

drawing planar graphs nicely [14]. This problem has been extensively studied for the cases of making a whole graph k-edge-connected or k-vertex-connected for various values of k (see the survey in [9]).

Recently, there have been studies on how to make only a given vertex subset satisfy a connectivity requirement [4, 25, 28, 29, 30]. This generalization of the smallest augmentation problem arises naturally from practical applications. For example, in the case of improving network reliability, a system designer is usually concerned with the connectivity only of certain critical nodes, instead of all the nodes in a system. Thus we are required to improve the reliability only of those important nodes. In [28], it is mentioned that 2-edge-connecting or biconnecting a given vertex subset can be done in linear time by adapting linear-time algorithms for 2-edge-connecting or biconnecting a whole graph, respectively. In [30], a linear-time algorithm is given to 3-edge-connect a given vertex subset. In [25], an $O(\lambda^2 \cdot n \cdot (n + k \cdot \log \lambda) + m)$-time algorithm is given to raise by one the edge-connectivity of a given subset of k vertices H where λ is the edge-connectivity of H, and the input graph is a multi-graph with n vertices and m edges. When the edge-connectivity of H is equal to that of the input graph, an $O(n \cdot \log n + m)$-time algorithm is also presented. In [4], the problem of satisfying edge-connectivity requirements on a specified subset of vertices for undirected graphs and directed graphs is solved.

In this paper, we further generalize the smallest augmentation problem as follows. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two connectivity requirements with the following properties:

- $\mathcal{R}_1$ and $\mathcal{R}_2$ are monotonic with respect to edge additions, i.e., if a graph holds $\mathcal{R}_i$, the same graph augmented with any additional edges also holds $\mathcal{R}_i$.
- If $\mathcal{R}_1$ is satisfied, $\mathcal{R}_2$ is satisfied, i.e., if a graph holds $\mathcal{R}_1$, it also holds $\mathcal{R}_2$.

Let $G = (V, E)$ be an undirected graph with n vertices and m edges. Let $H_1 \subseteq V$ and $H_2 \subseteq V$. The *smallest bi-level augmentation problem* is that of inserting the smallest number of edges to G such that in the resulting graph, the vertices in H_1 satisfy $\mathcal{R}_1$ and the vertices in H_2 satisfy $\mathcal{R}_2$. This paper focuses on the case where $\mathcal{R}_1$ is biconnectivity and $\mathcal{R}_2$ is 2-edge-connectivity.

We present a data structure to capture essential connectivity information of H_1 and H_2 simultaneously. This data structure can be efficiently updated after a new edge is added to G, and is useful for dynamically maintaining bi-level-connectivity information [19, 22, 31]. Using this data structure, we solve the bi-level augmentation problem in $O(n+m)$ time. Our algorithm can be parallelized to run in $O(\log^2 n)$ time using $n + m$ processors on an EREW PRAM. We use the algorithm to solve several optimization problems for protecting sensitive information in cross tabulated tables (see §3.1) and for improving the reliability of critical

nodes in communication networks (see §3.2). Furthermore, by properly setting G, H_1 and H_2, our augmentation algorithm also subsumes several existing optimal algorithms for graph augmentation, including those for 2-edge-connecting a vertex subset [29], 2-edge-connecting a whole graph [3], biconnecting a vertex subset [29], and biconnecting a whole graph [12, 23].

The main result of this paper is formally stated in the following theorem.

Theorem 1.1 *Given G, H_1 and H_2, the smallest bi-level augmentation problem can be solved in optimal linear time.*

Before proving this theorem in §4 through §6, we give some key definitions in §2 and solve application problems with this theorem in §3.

2 Definitions

Let u and v be two distinct vertices in G. If u and v cannot be separated by removing any edge in G, then u and v are *2-edge-connected.* If every pair of vertices in G are 2-edge-connected, then G is *2-edge-connected.* A set of vertices H is 2-edge-connected, if every pair of vertices in H are 2-edge-connected. If u and v cannot be separated by removing any single edge or any single vertex in G, then u and v are *biconnected.* If every pair of vertices in G are biconnected, then G is biconnected. A set of vertices H is biconnected, if every pair of vertices in H are biconnected.

A *2-edge-block* in G is either (1) a set $\{u\}$, where u is a degree-0 or degree-1 vertex, or (2) a maximal subset of vertices where every pair of vertices are 2-edge-connected. A *2-block* is either (1) a set $\{u\}$, where u is a degree-0 or degree-1 vertex, or (2) a maximal subset of vertices where every pair of vertices are biconnected. A 2-block or 2-edge-block consisting of exactly one vertex is called *trivial.*

An edge whose removal makes the resulting graph contain more connected components than the original graph is called a *cut edge.* A vertex whose removal makes the resulting graph contain more connected components than the original graph is a *cut vertex.* Let $nc(G)$ denote the number of connected components in G. A *strict* cut vertex is a cut vertex c such that (1) c is not an endpoint of a cut edge, or (2) $nc(G-\{c\})-nc(G) \geq 2$. A singular block consisted of a strict cut vertex is a *strict cut 2-block.*

Given two subsets of vertices H_1 and H_2 in $G = (V, E)$, G is *bi-level-connected with respect to H_1 and H_2*, if H_1 is biconnected and H_2 is 2-edge-connected.

Most of the definitions given here can be found in [1, 8, 11, 12].

3 Motivations

By properly choosing G, H_1 and H_2, we can use our augmentation algorithm to solve several optimization problems for protecting sensitive information in cross tabulated tables [6, 7, 17, 15, 16, 20] and for improving the reliability of communication networks [5, 13, 24].

3.1 Protecting sensitive information in cross tabulated tables

To protect sensitive information in a cross tabulated table $\mathcal{T}$, it is a common practice to suppress some of the cells in $\mathcal{T}$. A fundamental issue concerning the effectiveness of this practice is how a table maker can suppress a small number of cells in addition to the sensitive ones so that the resulting $\mathcal{T}$ does not leak significant information.

This protection problem can be reduced to various augmentation problems for a bipartite graph $\mathcal{B}$ constructed from $\mathcal{T}$ [6, 7, 17, 15, 16, 20]. The rows and the columns of $\mathcal{T}$ form the two vertex sets of $\mathcal{B}$, respectively; $\mathcal{B}$ has an edge between row i and column j if and only if the value of the (i, j)-th cell in $\mathcal{T}$ is suppressed [7].

Let U be a set of rows and columns in $\mathcal{T}$. We say that $\mathcal{T}$ and U are *bi-level protected* if the value of each suppressed cell cannot be uniquely determined and no nontrivial information about each row or column in U is revealed. It has been shown [7] that $\mathcal{B}$ has a cut edge between row i and column j if and only if the value of the (i, j)-th cell in $\mathcal{T}$ can be deduced from the published data of $\mathcal{T}$. Furthermore, a row vertex in $\mathcal{B}$ is a cut vertex if and only if nontrivial information about that row in $\mathcal{T}$ is revealed [17]. The same holds for columns. Therefore, $\mathcal{T}$ and U are bi-level protected if and only if $\mathcal{B}$ has no cut edge and none of the vertices in U is a cut vertex of $\mathcal{B}$ [17]. As a consequence, the problem of suppressing the minimum number of additional cells to bi-level protect $\mathcal{T}$ and U is equivalent to that of adding the minimum number of edges to $\mathcal{B}$ such that $\mathcal{B}$ has no cut edge and U contains no cut vertex of $\mathcal{B}$. Note that the newly added edge must preserve the bipartiteness of $\mathcal{B}$. We call this graph augmentation problem the *bipartite bi-level graph augmentation problem.* A linear-time algorithm for this graph problem can be directly used to bi-level protect $\mathcal{T}$ and U in optimal linear time [17].

Theorem 3.1 *Given $\mathcal{B}$ and U, we can solve the bipartite bi-level graph augmentation problem in linear time in the size of $\mathcal{B}$.*

Proof. In [10], techniques are developed to find a smallest biconnectivity augmentation while preserving the bipartiteness of the input graph. Using the above techniques and Theorem 1.1, we can easily derive this theorem. □

3.2 Improving network reliability

A communication network N can be modeled as a graph $G = (V, E)$, where the graph vertices are the network nodes and the graph edges correspond to the communication links. For N to satisfy a given reliability requirement, G often must hold some equivalent connectivity property [5, 13, 24]. We wish to add to N as few new communication links as possible such that the resulting N meets its desired reliability requirement. Below we discuss several network reliability requirements for N [5, 13, 24]. For each requirement, we solve the problem of enhancing the reliability of N by first properly setting H_1 and H_2 and then using our graph augmentation algorithm to solve the equivalent augmentation problem for G. The running time of our algorithms are linear and are as efficient as the best previous known algorithms.

- **No single link failure may disconnect the network[3].** The corresponding graph problem is that of adding edges to 2-edge-connect G. We set $H_1 = \emptyset$ and $H_2 = V$.

- **No single link failure may disconnect a given set H of critical nodes[29].** The equivalent graph problem is that of 2-edge-connecting H. We set $H_1 = \emptyset$ and $H_2 = H$.

- **No single node failure may disconnect the network[12, 23].** The corresponding graph problem is that of adding edges to biconnect G. We set $H_1 = H_2 = V$.

- **No single node failure may disconnect a given set H of critical nodes[29].** The equivalent graph problem is that of biconnecting H. We set $H_1 = H_2 = H$.

- **A bi-level reliability requirement.** Suppose that the communication nodes in N are assigned two different levels of importance. A certain subset H_1 of nodes may not be disconnected by a single node failure, and another subset H_2 of nodes may not be disconnected by a single link failure. The corresponding graph problem is that of adding edges to biconnect H_1 and to 2-edge-connect H_2 simultaneously. Our graph augmentation algorithm solves the reliability problem in optimal linear time. This solution is the first known polynomial-time algorithm for this reliability problem.

4 Preliminaries

In this paper, all graphs are undirected. Given a graph G', an edge subset E' and a vertex subset V', $G' - V'$ denotes G' without the vertices in V' and their adjacent edges. $G' - E'$ is the graph constructed from G' by removing the edges in E'. $G' \cup E'$ is the resulting G' after the edges in E' are added to G'. In a un-rooted tree, a *leaf* is a degree-1 vertex. In a rooted tree, a *leaf* is a degree-1 vertex that is not the root. An *isolated* vertex in a forest is a degree-0 vertex.

We now describe several basic properties concerning 2-edge-connectivity.

Lemma 4.1 *1. Let u and v be 2-edge-connected in G. Let G' be the graph obtained from G by adding a set E' of edges. If u and w are 2-edge-connected in G', then v and w are 2-edge-connected in G'.*

2. A vertex is in exactly one 2-edge-block.

3. Let u and v be 2-edge-connected in G. Let G' be the graph obtained from G by adding a set E' of edges. If u and w are biconnected in G', then u, v, and w are in the same 2-edge-block

Proof. It is well-known that vertices in 2-edge-connected components of a graph can be defined by the following equivalence relation [8]. Two vertices u and v are in the same equivalence class if u and v are 2-edge-connected. Hence we can derive all three properties stated in this lemma. □

4.1 Two-edge-block graphs

We construct a forest 2EBLK(G), called the *2-edge-block graph* of G by shrinking vertices in a 2-edge-block into a vertex [8]. The graph 2EBLK(G) is a forest and it is a tree if G is connected.

Given $G = (V, E)$ and $H \subseteq V$, we also construct the *specified 2-edge-block graph* S2EBLK(G, H) as follows. Let $\{Y_1, \ldots, Y_b\}$ be the set of 2-edge-blocks of G. Let $W = \{Y_i \mid H \cap Y_i \neq \emptyset\}$. Let $P_{a,b}(\text{2EBLK}(G))$ be the simple path in 2EBLK(G) between two 2-edge-blocks a and b if a and b are connected in 2EBLK(G). The graph S2EBLK(G, H) is the induced subgraph of 2EBLK(G) on the set of vertices $W \cup \{Y_i \mid \exists a, b \in W$such that $Y_i \in P_{a,b}(\text{2EBLK}(G))\}$.

Intuitively, a leaf in S2EBLK(G, H) is a 2-edge-block containing a vertex in H. Any 2-edge-block containing a vertex in H is in S2EBLK(G, H). In addition, a 2-edge-block B containing no vertex in H may be in S2EBLK(G, H), if B is in a path (in 2EBLK(G)) between two 2-edge-blocks containing vertices in H. A cut edge is in S2BLK(G, H) if its removal separates two vertices in H. An example of a graph, its 2-edge-block graph and its specified 2-edge-block graph is shown in Fig. 1.

Lemma 4.2 ([8, 29]) *The graph* S2EBLK(G, H) *is a forest, and* S2EBLK(G, H) *is an isolated vertex if there are two edge-disjoint paths between every pair of vertices in H. Furthermore,* S2EBLK(G, H) *is a tree if and only if H is in a connected component of G.*

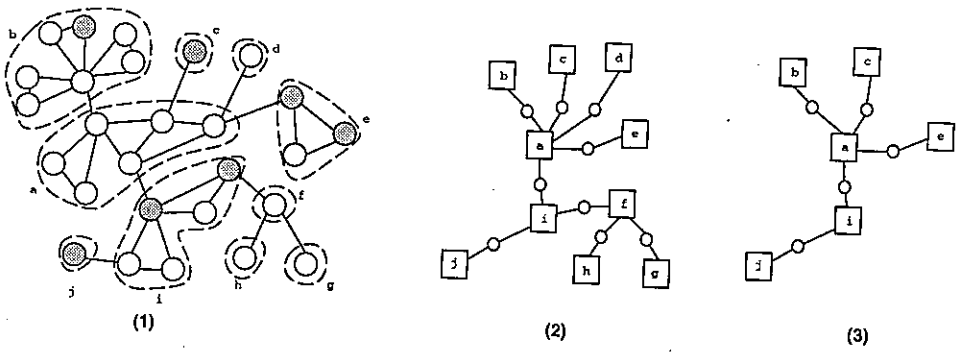

Figure 1: The input graph is shown in (1) where vertices in a 2-edge-block are grouped into a dashed circle. Its 2-edge-block graph is shown in (2) where rectangles are 2-edge-blocks and circles are cut edges. Its specified 2-edge-block graph on the set of vertices that are colored grey is shown in (3).

Lemma 4.3 *Given $G = (V, E)$ and $H \subseteq V$, we can construct its specified 2-edge-block graph* $\textsc{s2eblk}(G, H)$ *in linear time.*

Proof. Let P be the simple path in the rooted tree between two vertices u and v. Note that if w is on P, then w is an ancestor of either u or v. By using this observation, we give the following algorithm to construct $\textsc{s2eblk}(G, H)$.

1. let $F = \textsc{2eblk}(G)$;
2. for each 2-edge-block B in F, decide whether it contains a vertex in H;
3. remove trees in F that do not contain any 2-edge-block with a vertex in H;
4. for each remaining tree T in F do the following:
   (a) root T at a 2-edge-block that contains a vertex in H;
   (b) removing a vertex u and all adjacent edges from T if in the subtree of T rooted at u, there is no 2-edge-block containing a vertex in H;
5. return the resulting forest as $\textsc{s2eblk}(G, H)$;

The 2-edge-block graph can be constructed in linear time [8]. The rest of the operations take a total of linear time. Hence the lemma holds. □

Let B be a 2-edge-block in $\textsc{s2eblk}(G, H)$. The *2-edge-connectivity demanding vertex* for B with respect to $\textsc{s2eblk}(G, H)$, $\textsc{edem}(G, H, B)$ is (1) the (only) vertex in B if B is

trivial or (2) a vertex in B that is not an endpoint of any cut edge in S2EBLK(G, H). If B is a leaf or an isolated vertex in S2EBLK(G, H), then the 2-edge-connectivity demanding vertex for B is different from the demanding vertex for any other 2-edge-block in S2EBLK(G, H). By adding an edge to connect a demanding vertex in a 2-edge-block B that is a leaf (respectively, isolated vertex) in S2EBLK(G, H) and a vertex that is not in B, B is no longer a leaf (respectively, isolated vertex).

In the following lemma, let Y_1 and Y_2 be 2-edge-blocks that are leaves in S2EBLK(G). Let e be an edge between a demanding vertex in Y_1 and one in Y_2. Let P be the tree path of S2EBLK(G) between Y_1 and Y_2.

Lemma 4.4 ([3, 8, 12]) *1. The 2-edge-blocks of* S2EBLK(G) *on P are collapsed into a new 2-edge-block in* S2EBLK$(G \cup \{e\})$. *All the other 2-edge-blocks remain the same.*

2. If P contains two vertices of degrees at least three or one vertex of degree at least four, then the number of leaves in S2EBLK$(G \cup \{e\})$ *is equal to the number of leaves in* S2EBLK(G) *minus two.*

Lemma 4.5 *Let ℓ be the number of degree-1 vertices in* S2EBLK(G, H) *and let q be the number of isolated vertices in* S2EBLK(G, H). *We need to add at least $q + \lceil \ell/2 \rceil$ edges to G in order for H to be 2-edge-connected, if $q + \ell > 1$. Furthermore, we can add a set of edges with the above cardinality to make H 2-edge-connected.*

Proof. Similar to the proof for augmenting all vertices to meet the 2-edge-connectivity requirement [3]. □

4.2 Two-block graphs

We construct a forest 2BLK(G), called the *2-block graph* of G, to organize the 2-blocks that are not strict cut 2-blocks, cut edges, or strict cut vertices in G. The 2-block graph constructed below is a minor variation of the *bc-forest* given in [8, 11, 27]. Let $\{Y_1, \ldots, Y_b\}$ be the set of 2-blocks that are not strict cut 2-blocks of G. Let $\{u_1, \ldots, u_c\}$ be the set of strict cut vertices. Let $\{e_1, \ldots, e_w\}$ be the set of cut edges.

The vertex set of 2BLK(G) is $\{Y_1, \ldots, Y_b\} \cup \{u_1, \ldots, u_c\} \cup \{e_1, \ldots, e_w\}$, i.e., each non-strict cut 2-block, strict cut vertex or cut edge of G is regarded as a vertex in 2BLK(G). The vertices in 2BLK(G) corresponding to non-strict cut 2-blocks are *b-vertices*, and those corresponding to cut vertices and cut edges are *c-vertices*. The edge set of 2BLK(G) is the union of the sets $\{(Y_i, e_j) \mid$ an endpoint v of e_j is in Y_i and v is not a strict cut vertex$\}$,

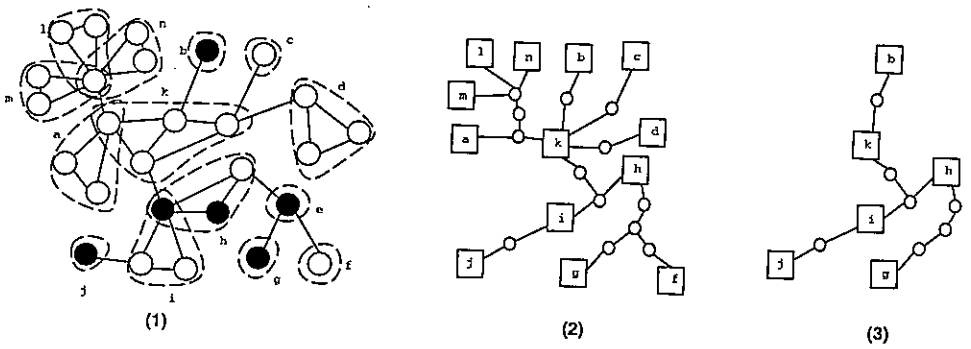

Figure 2: The input graph is shown in (1), where vertices in a 2-block are grouped into a dashed circle. Its 2-block graph is shown in (2), where rectangles are 2-blocks that are not strict cut 2-blocks, and circles are strict cut vertices or cut edges. Its specified 2-block graph on the set of vertices that are colored black is shown in (3).

$\{(u_i, e_j) \mid u_i \text{ is an endpoint of } e_j\}$, and $\{(Y_i, u_j) \mid u_j \in Y_i\}$. In other words, there is an edge between Y_i and u_j if and only if u_j is a vertex in the block Y_i. There is an edge between Y_i and e_j if and only if one endpoint of e_j is in the block Y_i and this endpoint is not a strict cut vertex. There is an edge between u_i and e_j if and only if u_i is an endpoint of e_j. Remark that if a strict cut vertex v forms a block $\{v\}$ by itself, then $\{v\}$ does not appear in the 2-block graph as a b-vertex. Instead, v appears as a c-vertex in the 2-block graph.

Given $G = (V, E)$ and $H \subseteq V$, we can also construct the *specified 2-block graph* $\text{S2BLK}(G, H)$ as follows. Let $\{Y_1, \ldots, Y_b\}$ be the set of 2-blocks in G. Let $W = \{Y_i \mid H \cap Y_i \neq \emptyset\}$. Let $P_{a,b}(\text{2BLK}(G))$ be the simple path in $\text{2BLK}(G)$ between two 2-blocks a and b if a and b are connected in $\text{2BLK}(G)$. The graph $\text{S2BLK}(G, H)$ is the induced subgraph of $\text{2BLK}(G, H)$ on the set of vertices $W \cup \{Y_i \mid \exists a, b \in W \text{such that } Y_i \in P_{a,b}(\text{2BLK}(G))\}$. Intuitively, a leaf in $\text{S2BLK}(G, H)$ must be a 2-block containing a vertex in H. A 2-block containing a vertex in H must be in $\text{S2BLK}(G, H)$. In addition, a 2-block B containing no vertex in H may be in $\text{S2BLK}(G, H)$, if B is in a path (in $\text{2BLK}(G)$) between two 2-blocks containing vertices in H. A cut edge or a strict cut vertex is in $\text{S2BLK}(G, H)$ if its removal separates two vertices in H. An example of a graph, its 2-block graph and its specified 2-block graph is shown in Fig. 2.

Lemma 4.6 *The graph* $\text{S2BLK}(G, H)$ *is a forest and* $\text{S2BLK}(G, H)$ *is an isolated vertex if there are two internally vertex-disjoint paths between every pair of vertices in* H. *Furthermore,* H *is in a connected component of* G *if and only if* $\text{S2BLK}(G, H)$ *is a tree.*

Proof. The proof of this lemma is similar to the proof of Lemma 4.2 for the specified 2-edge-block graph. □

Lemma 4.7 *Given $G = (V, E)$ and $H \subseteq V$, we can construct the specified 2-block graph* $\textsc{s2blk}(G, H)$ *in linear time.*

Proof. Similar to the proof of Lemma 4.3. □

Let B be a 2-block in $\textsc{s2blk}(G, H)$. The *biconnectivity demanding vertex* for B with respect to $\textsc{s2blk}(G, H)$, $\textsc{vdem}(G, B, H)$, is (1) the (only) vertex in B if B is trivial or (2) a vertex in B that is not a cut vertex or an endpoint of a cut edge in $\textsc{s2blk}(G, H)$. If B is a leaf or an isolated vertex in $\textsc{s2blk}(G, H)$, its biconnectivity demanding vertex is different from the demanding vertex of any other 2-block in $\textsc{s2blk}(G, H)$.

In the following lemma, let Y_1 and Y_2 be 2-blocks that are leaves in $\textsc{s2blk}(G)$. Let e be an edge between a demanding vertex in Y_1 and one in Y_2. Let $G' = G \cup \{e\}$. Let P be the tree path in $\textsc{s2blk}(G)$ between Y_1 and Y_2. The next lemma analyzes the case of adding an edge within a connected component.

Lemma 4.8 ([3, 8, 23]) *When an edge is added between demanding vertices of two 2-blocks that are leaves in* $\textsc{s2blk}(G)$*, we can apply the following operations to obtain the resulting 2-block graph from the original specified 2-block graph.*

1. *The 2-blocks on P are merged into a new 2-block Y' in* $\textsc{s2blk}(G \cup \{e\})$*. All the other 2-blocks remain the same.*
2. *The c-vertices on P that are of degree two are no longer c-vertices in* $\textsc{s2blk}(G \cup \{e\})$*. All the other c-vertices remain the same.*
3. *If a c-vertex is adjacent to a 2-block on P in* $\textsc{s2blk}(G)$*, then it is adjacent to the new 2-block Y' in* $\textsc{s2blk}(G \cup \{e\})$*. All the other edges remain the same.*

Lemma 4.9 *Let ℓ be the number of degree-1 vertices in* $\textsc{s2blk}(G, H)$ *and let q be the number of isolated vertices in* $\textsc{s2blk}(G, H)$*. If $q + \ell > 1$, then we need to add at least* $\max\{c - 1, q + \lceil \ell/2 \rceil\}$ *edges to G in order for H to be biconnected, where c is the maximum number of trees in the resulting graph of* $\textsc{s2blk}(G, H)$ *obtained by removing a strict cut vertex. Furthermore, we can add a set of edges with the above mentioned cardinality to make H biconnected.*

Proof. Similar to the proof in [3, 12] for augmenting all vertices to meet the biconnectivity requirement. □

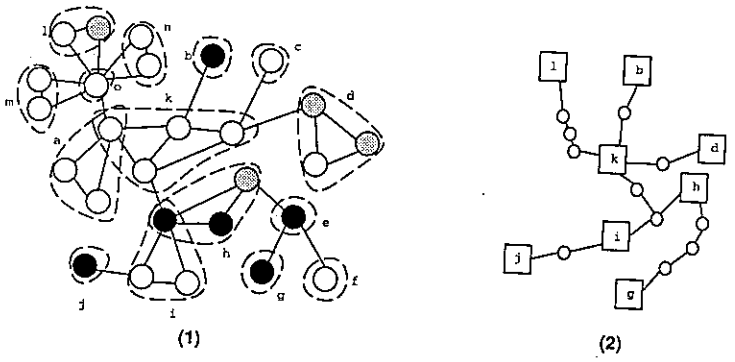

Figure 3: The input graph is shown in (1), where vertices in a 2-block are grouped into a dashed circle. Its specified bi-level-block graph for biconnecting the set of vertices that are colored black and for 2-edge-connecting the set of vertices that are colored black or grey is shown in (2). In (2), rectangles are 2-blocks, and circles are strict cut vertices or cut edges.

5 The specified bi-level-block graph

Given an undirected graph $G = (V, E)$ and two subsets of vertices H_1 and H_2 such that $H_1 \subseteq H_2 \subseteq V$, we construct the *specified bi-level-block graph* $\text{SBIBLK}(G, H_1, H_2)$ to represent the bi-level-connectivity information. The graph $\text{SBIBLK}(G, H_1, H_2)$ is a subgraph of $\text{2BLK}(G)$.

A 2-block containing a vertex in H_1 remains in $\text{SBIBLK}(G, H_1, H_2)$. A strict cut vertex or cut edge separates two vertices in H_1 or in H_2 also remains in $\text{SBIBLK}(G, H_1, H_2)$. All cut edges in $\text{S2EBLK}(G, H_2)$ remain in $\text{SBIBLK}(G, H_1, H_2)$. Let S be the set of remaining strict cut vertices and endpoints of remaining cut edges. For each 2-edge-block B that is a leaf or an isolated vertex in $\text{S2EBLK}(G, H_2)$, $B \cap H_1 = \emptyset$, and $B \cap H_2 \neq \emptyset$, we pick a 2-block in B containing exactly one vertex in S to remain in $\text{SBIBLK}(G, H_1, H_2)$. A 2-block also remains in $\text{SBIBLK}(G, H_1, H_2)$ if it is in a path (in $\text{2BLK}(G)$) between two previous chosen remaining 2-blocks. The $\text{SBIBLK}(G, H_1, H_2)$ is the induced subgraph of $\text{2BLK}(G)$ on the above set of chosen vertices.

Intuitively, in the specified bi-level-block graph, we preserve all the cut vertices and cut edges information in order for G to reach bi-level-connectivity with respect to H_1 and H_2. We preserve all 2-blocks containing vertices in H_1. Given a 2-edge-block W containing only vertices in H_2, one of its 2-blocks B is in the specified bi-level-block graph. Once we make B to biconnect with a 2-block in a 2-edge-block W', W also 2-edge-connects with W'. We add edges to the bi-level-block graph such that all of its 2-blocks are biconnected. By doing so, we make sure that H_1 is biconnected and at the same time H_2 is 2-edge-connected.

An example of a graph and its specified bi-level-block graph is given in Fig. 3.

Lemma 5.1 *(1) The specified bi-level block graph can be constructed in linear time. (2) For each connected component W containing a vertex in H_2,* $\textsc{sbiblk}(W, H_1, H_2)$ *is a tree. Furthermore, if W is bi-level-connected with respect to H_1 and H_2, then* $\textsc{sbiblk}(W, H_1, H_2)$ *is an isolated vertex. (3) The graph* $\textsc{s2blk}(G, H_1)$ *is a subgraph of* $\textsc{sbiblk}(G, H_1, H_2)$*. Furthermore, the graph* $\textsc{sbiblk}(G, H_1, H_2)$ *equals to* $\textsc{s2blk}(G, H_1)$ *if $H_1 = H_2$.*

Proof.

Part (1): We construct $\textsc{sbiblk}(W, H_1, H_2)$ using the following algorithm.

1. construct $\textsc{s2blk}(G, H_1)$, $\textsc{s2blk}(G, H_2)$, and $\textsc{s2eblk}(G, H_2)$;
2. let $F = \textsc{2blk}(G)$;
3. for each 2-block W_1 in F, decide whether it contains a vertex in H_1; let this set of 2-blocks be B_1;
4. for each 2-edge-block W_2 that is degree-0 or degree-1 in $\textsc{s2eblk}(G, H_2)$ and $W_2 \cap H_1 = \emptyset$, pick a 2-block in W_2 that is degree-0 or degree-1 in $\textsc{s2blk}(G, H_2)$; let this set of 2-blocks be B_2;
5. remove trees in F that do not contain any 2-block with a vertex in $H_1 \cup H_2$;
6. for each remaining tree T in F do the following:
   (a) root T at a 2-block that contains a vertex in $H_1 \cup H_2$;
   (b) removing a vertex u and all adjacent edges from T if in the subtree of T rooted at u, there is no 2-block in $B_1 \cup B_2$;
7. return the resulting forest as $\textsc{sbiblk}(G, H_1, H_2)$;

From Lemmas 4.3 and 4.7, all the the above steps can be computed in linear time. Thus this part of the lemma holds.

Parts (2) and (3): Straightforward. □

Let B be a 2-block in $\textsc{sbiblk}(G, H_1, H_2)$. A *demanding vertex* for B with respect to H_1 and H_2 is either (1) the (only) vertex in B if B is trivial, or (2) a vertex in B that is not a strict cut vertex in $\textsc{sbiblk}(G, H_1, H_2)$ or an endpoint of a cut edge in $\textsc{sbiblk}(G, H_1, H_2)$. It is easy to see that if B is a leaf or an isolated vertex in $\textsc{sbiblk}(G, H_1, H_2)$, then B has a demanding vertex that is different from the demanding vertex of any other 2-block in $\textsc{sbiblk}(G, H_1, H_2)$. By adding an edge to connect a demanding vertex in a 2-block B that is a leaf (respectively, isolated vertex) and a vertex not in B, B is no longer a leaf (respectively, isolated vertex).

Lemma 5.2 *Let B be a 2-block that is a leaf or an isolated vertex in* $\textsc{sbiblk}(G, H_1, H_2)$ *and $B \cap H_1 = \emptyset$. Let W be the 2-edge-block where B is contained. Then (1) The 2-edge-block W is a leaf or an isolated vertex in* $\textsc{s2eblk}(G, H_2)$. *(2) A demanding vertex for B in* $\textsc{sbiblk}(G, H_1, H_2)$ *is also a demanding vertex for W in* $\textsc{s2eblk}(G, H_2)$.

Proof. Note that part (1) easily follows from the fact that a 2-block that is a leaf or an isolated vertex in $\textsc{sbiblk}(G, H_1, H_2)$ is chosen within a 2-edge-block that is a leaf or an isolated vertex in $\textsc{s2eblk}(G, H_2)$. Part (2) follows from the fact that a demanding vertex in B is not an endpoint of a cut edge in $\textsc{s2eblk}(G, H_2)$. Thus it is also a demanding vertex for W in $\textsc{s2eblk}(G, H_2)$. □

In the following corollary, let W be a 2-edge-block in G such that $W \cap H_1 = \emptyset$ and $W \cap H_2 \neq \emptyset$. Let B be a 2-block in W that is also in $\textsc{sbiblk}(G, H_1, H_2)$. Let u be a vertex in W and let v be a vertex in B. Let w be a vertex in G.

Corollary 5.3 *If v and w are biconnected in G', the resulting graph of G obtained by adding edges, then u, v, and w are 2-edge-connected in G'.*

Proof. By Lemmas4.1 and 5.2. □

6 Our augmentation algorithm

In this section, let G be an undirected graph and let H_1 and H_2 be two subsets of vertices in G.

We first establish a lower bound on the number of edges needed to add in order for a graph to be bi-level-connected. This lower bound is based on the specified bi-level-block graph. We then give an algorithm that adds exactly the number of edges as given in the lower bound.

Given a graph G, let $\textsc{nc}(G)$ be the number of connected components in G. Thus $\textsc{nc}(\textsc{sbiblk}(G, H_1, H_2))$ is the number of connected components in $\textsc{sbiblk}(G, H_1, H_2)$. Given a vertex $c \in \textsc{s2blk}(G, H_1)$, let F_c be the connected component in $\textsc{sbiblk}(G, H_1, H_2)$ containing c. Let $K_c = F_c - \{c\}$. We partition K_c into K'_c and K''_c where K'_c consists of connected components in which there is a 2-block with a vertex in H_1 and K''_c consists of connected components in which there is no 2-block with a vertex in H_1. Note that each connected component in K''_c contains a 2-block with a vertex in $H_2 \setminus H_1$. We define $\textsc{sc}(c, G, H_1, H_2) = \textsc{nc}(K'_c)$. We further define $\textsc{le}(c, G, H_1, H_2)$ to be the number of 2-blocks in K''_c that are leaves in $\textsc{sbiblk}(G, H_1, H_2)$. The *cut constraint* for a vertex $c \in \textsc{s2blk}(G, H_1)$

is $\text{CC}(c, G, H_1, H_2) = \text{SC}(c, G, H_1, H_2) + \left\lceil \frac{\text{LE}(c,G,H_1,H_2)}{2} \right\rceil + \text{NC}(\text{SBIBLK}(G, H_1, H_2)) - 1$. For convenience, if a cut vertex $c \notin \text{S2BLK}(G, H_1)$, then $\text{CC}(c, G, H_1, H_2)$ is the degree of c in $\text{SBIBLK}(G, H_1, H_2)$.

The *degree constraint* $\text{DC}(G, H_1, H_2) = \max_{c \in \text{S2BLK}(G,H_1)} \text{CC}(c, G, H_1, H_2) - 1$. The *leaf constraint* $\text{LC}(G, H_1, H_2) = q + \lceil \ell/2 \rceil$, where q and ℓ are the numbers of isolated vertices and degree-1 vertices in $\text{S2BLK}(G, H_1, H_2)$, respectively. A graph is *bi-level-balanced* with respect to H_1 and H_2 if there is no cut vertex $c \in \text{SBIBLK}(G, H_1, H_2)$ with $\text{CC}(c, G, H_1, H_2) - 1 > \text{LC}(G, H_1, H_2)$. Note that removing c might not separate two vertices in H_1, instead, it might separate two vertices in H_2. The following corollary states the relation between the degree constraint and the leaf constraint.

Corollary 6.1 *Assume that* $\text{SBIBLK}(G, H_1, H_2)$ *is connected. Let* c_1 *and* c_2 *be two cut vertices in* $\text{SBIBLK}(G, H_1, H_2)$. *Then* $(\sum_{i=1}^{2} \text{CC}(c_i, G, H_1, H_2)) - 1 \leq \ell$, *where* ℓ *is the number of leaves in* $\text{SBIBLK}(G, H_1, H_2)$.

Proof. Let $\ell' = \ell - \sum_{i=1}^{2} \text{LE}(c_i, G, H_1, H_2)$. As a corollary of [12, Lemma 3.1], $\ell' \geq (\sum_{i=1}^{2} \text{SC}(c_i, G, H_1, H_2)) - 1$. Thus $\ell \geq (\sum_{i=1}^{2} \text{CC}(c_i, G, H_1, H_2)) - 1$. □

Lemma 6.2 *If* $q + \ell > 1$, *we need to add at least* $\max\{\text{DC}(G, H_1, H_2), \text{LC}(G, H_1, H_2)\}$ *edges to* G *such that in the resulting graph* H_1 *is biconnected and* H_2 *is 2-edge-connected.*

Proof. Note that a tree with more than one vertex contains at least two leaves. Thus if $q + \ell \leq 1$, then $q = 1$ and $\ell = 0$. Hence G is bi-level-connected with respect to H_1 and H_2.

We first prove the first component of the lower bound. Let $\text{NC}(\text{SBIBLK}(G, H_1, H_2)) = p_1 + p_2$, where there are p_1 connected components in $\text{SBIBLK}(G, H_1, H_2)$ with a vertex in H_1. For the first component of the lower bound, note that removing a vertex $c \in \text{S2BLK}(G, H_1)$ separates G into $\text{SC}(c, G, H_1, H_2) + p_1 - 1$ connected components where each of them contains a vertex in H_1. In order for H_1 to be biconnected, we need to add at least $\text{SC}(c, G, H_1, H_2) + p_1 - 2$ edges to connect these $\text{SC}(c, G, H_1, H_2) + p_1 - 1$ connected components. In addition, in order for H_2 to be 2-edge-connected, we need to add two edges to each of the p_2 connected components in $\text{SBIBLK}(G, H_1, H_2)$ where each of them contains a vertex in H_2. Each 2-block that is a leaf in K_c'' corresponds to an 2-edge-block that is a leaf in $\text{S2EBLK}(G, H_2)$, to which we must add an edge. Thus we also need to add an additional of $\left\lceil \frac{\text{LE}(c,G,H_1,H_2)}{2} \right\rceil + p_2$ edge. Hence given a vertex $c \in \text{S2BLK}(G, H_1)$, we need to add at least $\text{CC}(c, G, H_1, H_2)$ edges.

We then prove the second component of the lower bound. Assume that W is a 2-block in G. If W is degree-0, then W is in a connected component which needs to add at least two

more edges to 2-edge-connect (or biconnect) vertices in W with vertices in other connected components. Thus we need to add two edges to each 2-block that is degree-0.

We now assume that W is degree-1. If $W \cap H_1 \neq \emptyset$, then there is exactly one cut vertex or one cut edge whose removal separates a vertex in W and a vertex in $H_1 \setminus W$. Let this cut vertex or cut edge be d. Let G_W be the connected component in $G - \{d\}$ that contains a vertex in W. We need to add an edge to G_W in order for vertices in H_1 to be biconnected. If $W \cap H_1 = \emptyset$, then $W \cap H_2 \neq \emptyset$. Furthermore, W is in a 2-edge-block B_W with $B_W \cap H_1 = \emptyset$ and $W \cap H_2 \neq \emptyset$. The 2-edge-block B_W is degree-1 in $\text{S2EBLK}(G, H_2)$. Assume that B_W is adjacent to the cut edge e in $\text{S2EBLK}(G, H_2)$ and let G_W be the connected component that contains B_W in $G - \{e\}$. We need to add at least one edge to G_W in order for vertices in H_2 to be 2-edge-connected. Given two 2-blocks W_1 and W_2 that are both degree-1, it is clear that $G_{W_1} \cap G_{W_2} = \emptyset$. From the above, we can conclude that we need to add at least one edge for each 2-block that is degree-1.

By the above discussion and and the fact that an edge has two endpoints, the second component of the lower bound holds. □

6.1 $\text{SBIBLK}(G, H_1, H_2)$ is disconnected

We first consider the case when vertices in $H_1 \cup H_2$ are in several connected components in G.

Lemma 6.3 *If $\text{SBIBLK}(G, H_1, H_2)$ is disconnected, then we can add $r - 1$ edges to G in linear time such that in the resulting graph G', $\text{DC}(G', H_1, H_2) = \text{DC}(G, H_1, H_2) - (r - 1)$, $\text{LC}(G', H_1, H_2) = \text{LC}(G, H_1, H_2) - (r - 1)$ and vertices in H_2 are in the same connected component in G', where $r = \text{NC}(\text{SBIBLK}(G, H_1, H_2))$.*

Proof. By Lemma 5.1.(2), there are r trees in $\text{SBIBLK}(G, H_1, H_2)$. We number trees in $\text{SBIBLK}(G, H_1, H_2)$ consecutively from 1 to r. Note that if a tree is not an isolated vertex, then there are at least two distinct degree-1 vertices. If the ith tree, $1 \leq i \leq r$, is not an isolated vertex, then let R_i and L_i be two distinct degree-1 vertices in it. If the ith tree is an isolated vertex W, then let $R_i = L_i = W$. Note that both R_i and L_i, $1 \leq i \leq r$, are 2-blocks. We add the set of $r - 1$ edges $E' = \{(u_i, v_i) \mid 1 \leq i < r\}$ to G, where u_i and v_i are demanding vertices of R_i and L_i, respectively.

In the resulting graph G' by adding E', all vertices in H_2 are in the same connected component. For a cut vertex c, $\text{CC}(c, G', H_1, H_2) = \text{CC}(c, G, H_1, H_2) - (r - 1)$. Thus $\text{DC}(G', H_1, H_2) = \text{DC}(G, H_1, H_2) - (r - 1)$.

If $r > 1$, then there is no isolated vertex in $\text{SBIBLK}(G', H_1, H_2)$. The number of leaves in $\text{SBIBLK}(G', H_1, H_2)$ is $2 \cdot (r-1-q)$ less than that of $\text{SBIBLK}(G, H_1, H_2)$, where q is the number of isolated vertices in $\text{SBIBLK}(G, H_1, H_2)$. Thus $\text{LC}(G', H_1, H_2) = \text{LC}(G, H_1, H_2) - (r-1)$. Hence the lemma holds. □

6.2 $\text{SBIBLK}(G, H_1, H_2)$ is connected, but not bi-level-balanced

If $\text{SBIBLK}(G, H_1, H_2)$ is connected, then $\text{LC}(G, H_1, H_2) = \lceil \ell/2 \rceil$, where ℓ is the number of degree-1 vertices in $\text{SBIBLK}(G, H_1, H_2)$. For a cut vertex $c \in \text{S2BLK}(G, H_1)$, $\text{CC}(c, G, H_1, H_2)$ equals to the degree of c in $\text{S2BLK}(G, H_1)$ plus $\left\lceil \frac{\text{LE}(c,G,H_1,H_2)}{2} \right\rceil$.

Given a vertex v in a tree T, a *v-chain* is a path in T from v to a leaf, in which every internal vertex is degree-2 in T [23]. In the following two lemmas, let $\delta = \text{DC}(G, H_1, H_2) - 1 - \text{LC}(G, H_1, H_2)$.

Lemma 6.4 *(1) There can be at most one cut vertex $c \in \text{SBIBLK}(G, H_1, H_2)$ with $\text{CC}(c, G, H_1, H_2) - 1 > \text{LC}(G, H_1, H_2)$. Furthermore, (2) if $c \notin \text{S2BLK}(G, H_1)$, then there are $2 \cdot \delta + 2$ distinct c-chains in $\text{SBIBLK}(G, H_1, H_2)$ in which each of them contains a vertex in H_2. (3) If $c \in \text{S2BLK}(G, H_1)$, then there are $2 \cdot \delta + 1$ distinct c-chains in $\text{SBIBLK}(G, H_1, H_2)$ where each of them contains a vertex in H_1.*

Proof. Assume that there are two vertices $c_i \in \text{SBIBLK}(G, H_1, H_2)$, $i \in \{1, 2\}$, such that $\text{CC}(c_i, G, H_1, H_2) - 1 > \text{LC}(G, H_1, H_2)$. We first consider the case when $c_1 \in \text{S2BLK}(G, H_1)$ and $c_2 \in \text{S2BLK}(G, H_1)$. Let ℓ be the number of degree-1 vertices in $\text{SBIBLK}(G, H_1, H_2)$. Then $\ell \geq \sum_{i=1}^{2} (\text{SC}(c_i, G, H_1, H_2) + \text{LE}(c_i, G, H_1, H_2)) - 1$. This violates the fact that $\text{CC}(c_i, G, H_1, H_2) - 1 = \text{SC}(c_i, G, H_1, H_2) + \left\lceil \frac{\text{LE}(c_i,G,H_1,H_2)}{2} \right\rceil - 1 \geq \left\lceil \frac{\ell}{2} \right\rceil$. Thus there is at most one vertex $c \in \text{S2BLK}(G, H_1)$ with $\text{CC}(c, G, H_1, H_2) - 1 > \text{LC}(G, H_1, H_2)$. The cases when $c_1 \notin \text{S2BLK}(G, H_1)$ or $c_2 \notin \text{S2BLK}(G, H_1)$ can be proved using an approach similar to the above.

Thus we may assume there is only one cut vertex c with $\text{CC}(c, G, H_1, H_2) - 1 > \text{LC}(G, H_1, H_2)$. Part (2) is proved in [23]. We now assume that $c \in \text{S2BLK}(G, H_1)$. Let x be the number c-chains containing a vertex in H_1. Then $\ell \geq 2 \cdot \text{SC}(c, G, H_1, H_2) + \text{LE}(c, G, H_1, H_2) - x$. However, $2 \cdot \text{CC}(c, G, H_1, H_2) - 2 \cdot \delta \geq \ell$. Thus we can deduce the fact that $x \geq 2 \cdot \delta + 1$. □

Lemma 6.5 *Let G be a graph such that $\text{SBIBLK}(G, H_1, H_2)$ is connected, but is bi-level-unbalanced with respect to H_1 and H_2. Let $c \in \text{SBIBLK}(G, H_1, H_2)$ be the vertex with $\text{NC}(c, G, H_1, H_2) - 1 > \text{LC}(G, H_1, H_2)$. (1) If $c \notin \text{S2BLK}(G, H_1)$, then we can add $2 \cdot \delta$ edges to G in linear time such that in the resulting graph G', $\text{LC}(G', H_1, H_2) = \text{LC}(G, H_1, H_2) - \delta$,*

and G' is bi-level-balanced with respect to H_1 and H_2. (2) If $c \in$ S2BLK(G, H_1), then we can add $2 \cdot \delta$ edges to G in linear time such that in the resulting graph G', DC$(G', H_1, H_2) =$ DC$(G, H_1, H_2) - 2 \cdot \delta$, and G' is bi-level-balanced with respect to H_1 and H_2.

Proof. **Part (1):** By (2) in Lemma 6.4, there are at least $2 \cdot \delta + 2$ c-chains. If $c \notin$ S2BLK(G, H_1), then every 2-block in each c-chain of SBIBLK(G, H_1, H_2) does not contain a vertex in H_1. Thus there is a 2-edge-block W in S2EBLK(G, H_2) whose degree is equal to DC(G, H_1, H_2) and $c \in W$. Each c-chain in SBIBLK(G, H_1, H_2) corresponds to a W-chain in S2EBLK(G, H_2). Hence there are $2 \cdot \delta + 2$ W-chains in S2EBLK(G, H_2). We number $2 \cdot \delta + 2$ of the W-chains consecutively from 1 to $2 \cdot \delta + 2$. Let v_i, $1 \leq i \leq 2 \cdot \delta + 2$, be a demanding vertex of the leaf in the ith W-chain. We add the set of new edges $E' = \{(v_i, v_{i+\delta}) \mid 1 \leq i \leq \delta\}$ to G. Let $G' = G \cup E'$. By Lemma 4.4, 2-edge-blocks in the first $2 \cdot \delta$ W-chains are collapsed into a new 2-edge-block together with W. Thus the number of c-chains in SBIBLK(G', H_1, H_2) is $2 \cdot \delta$ less than the number of c-chains in SBIBLK(G, H_1, H_2) and LC$(G', H_1, H_2) =$ LC$(G, H_1, H_2) - \delta$. Thus CC$(c, G', H_1, H_2) - 1 =$ LC(G', H_1, H_2). For any cut vertex $c' \neq c$, CC$(c', G', H_1, H_2) =$ CC(c', G, H_1, H_2). From Corollary 6.1, we know that CC$(c', G, H_1, H_2) - 1 \leq$ LC$(G, H_1, H_2) - \delta$. Thus G' is bi-level-balanced.

Part (2): By (3) in Lemma 6.4, we can find $2 \cdot \delta + 1$ c-chains with a vertex in H_1. Let u_i be a demanding vertex in the leaf of the ith c-chain. We add the set of new edges $E' = \{(u_i, u_{i+1}) \mid 1 \leq i \leq 2 \cdot \delta\}$. After adding E' to G, the resulting graph is bi-level-balanced. □

6.3 SBIBLK(G, H_1, H_2) is connected and bi-level-balanced

Note that in this case, for every cut vertex $c \in$ SBIBLK(G, H_1, H_2), $d(c, G, H_1, H_2) - 1 \leq$ LC(G, H_1, H_2), where $d(c, G, H_1, H_2)$ is the degree of c in SBIBLK(G, H_1, H_2).

Lemma 6.6 *If* SBIBLK(G, H_1, H_2) *is connected and bi-level-balanced, then we can add* LC(G, H_1, H_2) *edges in linear time to G such that in the resulting graph G', H_1 is biconnected and H_2 is 2-edge-connected.*

Proof. Note that in this case, DC$(G, H_1, H_2) \leq$ LC(G, H_1, H_2). Furthermore, for every cut vertex $c \in$ SBIBLK(G, H_1, H_2), $d(c, G, H_1, H_2) - 1 \leq$ LC(G, H_1, H_2), where $d(c, G, H_1, H_2)$ is the degree of c in SBIBLK(G, H_1, H_2). From §4.3 in [12], we know that if $d(c, G, H_1, H_2) - 1 \leq$ LC(G, H_1, H_2), for all cut vertex c, then we can biconnected the graph using exactly LC(G, H_1, H_2) edges. We thus can apply the smallest biconnectivity augmentation algorithm on the specified bi-level-block graph SBIBLK(G, H_1, H_2). Whenever there is an edge added in the algorithm between two 2-blocks, we add a corresponding edge between demanding

vertices of these two 2-blocks. The number of edges added is exactly $\text{LC}(G, H_1, H_2)$ and the running time of the algorithm is linear.

In the resulting graph, every pair of vertices in H_1 are biconnected. Let u_1 and u_2 be two vertices in H_2. If both u_1 and u_2 are in H_1, then they are biconnected. Thus they are also 2-edge-connected.

Assume that it is not the case that both u_1 and u_2 are in H_1. If u_i, $i \in \{1,2\}$, is in $H_2 \setminus H_1$, then u_i is in a 2-edge-block W_i and there is a 2-block B_i in W_i that is in $\text{SBIBLK}(G, H_1, H_2)$. If u_i, $i \in \{1,2\}$, is in H_1, then u_i is in a 2-block B_i. Note that u_i is 2-edge-connected to every vertex in B_i in this case. Note also that the biconnectivity augmentation algorithm biconnects every vertex in B_1 and every vertex in B_2. By Lemma 4.1, u_1 and u_2 are 2-edge-connected. Thus H_2 is 2-edge-connected. □

6.4 Proof of Theorem 1.1

Based on the previous discussion, we are ready to prove the correctness of our main theorem. We will describe an augmentation algorithm that adds exactly $\max\{\text{DC}(G, H_1, H_2), \text{LC}(G, H_1, H_2)\}$ edges to bi-level-connect G. We then prove that this algorithm adds the smallest number of edges and it runs in linear time.

Our algorithm first adds edges to connect $\text{SBIBLK}(G, H_1, H_2)$. We then add edges to balance $\text{SBIBLK}(G, H_1, H_2)$. Finally, we add edges to bi-level-connect G. Our algorithm is as follows.

1. If $\text{SBIBLK}(G, H_1, H_2)$ is disconnected, then we add edges according to Lemma 6.3 to connect the graph.
2. If $\text{SBIBLK}(G, H_1, H_2)$ is bi-level-unbalanced, then we add edges according to Lemma 6.5 to balance the graph.
3. Otherwise, we use the smallest biconnectivity augmentation algorithm to make H_1 biconnected and H_2 2-edge-connected according to Lemma 6.6.

Lemmas 6.6 makes sure that in the resulting graph H_1 is biconnected and H_2 is 2-edge-connected. According to Lemmas 6.3, 6.5, and 6.6, the number of edges added is $\max\{\text{DC}(G, H_1, H_2), \text{LC}(G, H_1, H_2)\}$. Thus by Lemma 6.2, we know that this is the smallest number of edges to add to G to make the resulting graph bi-level-connected with respect to H_1 and H_2.

By Lemmas 4.3, 4.7, and 5.1.(1), we can build the specified 2-edge-block graph, 2-block graph and bi-level-block graph in linear time. By Lemmas 6.3, 6.5, and 6.6, our algorithm runs in linear time given the above three graphs. Thus our overall algorithm runs in linear time.

Remark: Our augmentation algorithm can be efficiently implemented on an EREW PRAM to run in $O(\log^2 n)$ time using $n + m$ processors by first showing the specified bi-level-block graph can be constructed in the same complexities and then using the result in [12].

7 Parallel implementations

In this section, we give efficient parallel implementations for our graph augmentation algorithm on an EREW PRAM.

Lemma 7.1 *Given G, H_1 and H_2, we can construct the specified 2-edge-block graph, the specified 2-block graph and the specified bi-level-block graph in $O(\log^2 n)$ time using $n + m$ processors.*

Proof. It takes $O(\log^2 n)$ time using $n + m$ processors to construct the 2-edge-block forest and the 2-block forest [18, 21]. To construct the specified 2-edge-block (respectively, 2-block) forest we need to root each tree in the forest. We also need to check, for each vertex v in the rooted tree, whether there is a 2-edge-block (respectively, 2-block) containing a vertex of H_2 (respectively, H_1) in the subtree rooted at v. These operations can be done using efficient techniques for Euler tour and range-minimum queries in $O(\log^2 n)$ time using $n + m$ processors [26]. The specified bi-level-block graph can be constructed using similar techniques. □

Theorem 7.2 *Given G, H_1 and H_2, we can add the minimum number of edges to bi-level-connect G with respect to H_1 and H_2 in $O(\log^2 n)$ time using $n + m$ processors.*

Proof. By Lemma 7.1, we can construct key data structures used in our augmentation algorithm in $O(\log^2 n)$ time using $n + m$ processors. Then, it takes $O(1)$ time and $n + m$ processors to implement our augmentation algorithm when the specified bi-block graph is disconnected and bi-level-unbalanced. If the specified bi-level-graph is balanced, our augmentation algorithm takes $O(\log^2 n)$ time using $n + m$ processors [12]. □

References

[1] B. Bollobás. *Graph Theory: An Introductory Course.* Springer-Verlag, New York, 1979.

[2] N. Christofides and C. A. Whitlock. Network synthesis with connectivity constraints — a survey. In *Operational Research '81*, pages 705–723, 1981.

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

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

[5] 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.

[6] D. Gusfield. Optimal mixed graph augmentation. *SIAM Journal on Computing*, 16:599–612, 1987.

[7] D. Gusfield. A graph theoretic approach to statistical data security. *SIAM Journal on Computing*, 17:552–571, 1988.

[8] F. Harary. *Graph Theory.* Addison-Wesley, Reading, MA, 1969.

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

[10] T.-s. Hsu and M. Y. Kao. Optimal augmentation for bipartite componentwise biconnectivity in linear time. In *Proc. 7th International Symp. on Algorithms and Computation*, 1996, to appear.

[11] 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.

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

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

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

[15] M. Y. Kao. Linear-time optimal augmentation for componentwise bipartite-completeness of graphs. *Information Processing Letters*, pages 59–63, 1995.

[16] M. Y. Kao. Total protection of analytic invariant information in cross tabulated tables. *SIAM Journal on Computing*, 1995. To appear.

[17] M. Y. Kao. Data security equals graph connectivity. *SIAM Journal on Discrete Mathematics*, 9:87–100, 1996.

[18] R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 869–941. North Holland, 1990.

[19] J. A. La Poutré and J. Westbrook. Dynamic two-connectivity with backtracking. In *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 204–212, 1994.

[20] F. M. Malvestuto, M. Moscarini, and M. Rafanelli. Suppressing marginal cells to protect sensitive information in a two-dimensional statistical table. In *Proceedings of ACM Symposium on Principles of Database Systems*, pages 252–258, 1991.

[21] 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.

[22] M. Rauch. Improved data structures for fully dynamic biconnectivity. In *Proc. 26th Annual ACM Symp. on Theory of Computing*, pages 686–695, 1994.

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

[24] 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.

[25] S. Taoka and T. Watanabe. Minimum augmentation to k-edge-connect specified vertices of a graph. In *ISAAC'94*, volume LNCS #834, pages 217–225. Springer-Verlag, 1994.

[26] R. E. Tarjan and U. Vishkin. An efficient parallel biconnectivity algorithm. *SIAM J. Comput.*, 14:862–874, 1985.

[27] W. T. Tutte. *Connectivity in Graphs.* University of Toronto Press, 1966.

[28] T. Watanabe, Y. Higashi, and A. Nakamura. An approach to robust network construction from graph augmentation problems. In *Proc. of 1990 IEEE Int'l Symp. on Circuits and Systems*, pages 2861–2864, 1990.

[29] 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.

[30] T. Watanabe, S. Taoka, and T. Mashima. Minimum-cost augmentation to 3-edge-connect all specified vertices in a graph. In *Proc. of 1993 IEEE Int'l Symp. on Circuits and Systems*, pages 2311–2314, 1993.

[31] J. Westbrook and R. E. Tarjan. Maintaining bridge-connected and biconnected components on-line. *Algorithmica*, 7(5/6):433–464, 1992.