A $(2-2/|L|)$-Approximation Algorithm *R2VS* or *R2ES* to 2-Vertex- or 2-Edge-Connect Specified Vertices in a Graph

Makoto Tamura, Satoshi Taoka and Toshimasa Watanabe

Graduate School of Engineering, Hiroshima University
1-4-1, Kagamiyama, Higashi-Hiroshima, 739-8527 Japan
Phone : +81-824-24-7662 (Watanabe), -7666 (Taoka), -7661 (Tamura)
Facsimile : +81-824-22-7028
E-mail : {tamura, taoka, watanabe}@infonets.hiroshima-u.ac.jp

[Abstract] The 2-vertex-connectivity (2-edge-connectivity, respectively) augmentation problem for specified vertices, 2VCA-SV (2ECA-SV) for short, is defined as follows: "Given an undirected graph $G=(V,E)$, a spanning subgraph $G'=(V,E')$ of G, specified vertices $S\subseteq V$, and a weight function $w:E\rightarrow R^+$ (nonnegative real numbers), find a set $E''\subseteq E-E'$ with the minimum total weight, such that $G'+E''=(V,E'\cup E'')$ has at least two internally disjoint (edge disjoint) paths between any pair of vertices in S. In this paper, we propose a $(2-2/|L|)$-approximation algorithm *R2VC* or *R2EC* for 2VCA-SV or 2ECA-SV, respectively, if G' has a connected component containing S, where L is the set of leaves of a certain tree constructed from G' and S. Its time or space complexity is $O(|V||E|+|V|^2\log|V|+|L||V|^2)$ or $O(|V|^2)$, respectively.

1 Introduction

[Problem definitions] For a positive integer k, the *k-vertex-connectivity* (*k-edge-connectivity*, respectively) *augmentation problem for specified vertices*, kVCA-SV (kECA-SV) for short, is defined as follows: "Given an undirected graph $G=(V,E)$, a spanning subgraph $G'=(V,E')$ of G, specified vertices $S\subseteq V$, and a weight function $w:E\rightarrow R^+$ (nonnegative real numbers), find a set $E''\subseteq E-E'$ with the minimum total weight, such that $\kappa(S;G'+E'')\geq k$ ($\lambda(S;G'+E'')\geq k$) for $G'+E''=(V,E'\cup E'')$," where $\kappa(S;G'')\geq k$ ($\lambda(S;G'')\geq k$) means that G'' has at least k internally disjoint (edge disjoint) paths between any pair of vertices in S. We assume that $\kappa(S;G)\geq k$ ($\lambda(S;G)\geq k$) without loss of generality. If $S=V$ then kVCA-SV (kECA-SV) is denoted simply as kVCA (kECA), which is called the k-vertex-connectivity (k-edge-connectivity) augmentation problem. In this paper, we exclusively consider 2VCA-SV and 2ECA-SV.

By "an *r-approximation algorithm*" we mean that it produces a solution whose total weight is no more than r times the optimum. This solution is called "an *r-approximate solution*". It is also said that the performance ratio of the algorithm is r, and this statement is simply represented as $PR=r$.

[Related results] First, we state existing results for 2VCA and 2ECA. For 2VCA (2ECA, respectively), [1, 2] ([3]) proposed a linear time algorithm finding an optimum solution when G is a complete graph and any edge weight is unity. On the other hand, [4] proved that 2VCA and 2ECA are NP-hard if distinct edge weights exist. For 2VCA and 2ECA , [5] devised $O(|E|+|V|\log|V|)$ time 2-approximation algorithms when G' is connected. Moreover, for 2VCA (2ECA (and even for kECA for any $k\geq 2$), respectively), [6] ([7]) showed an $O(|V|^3|E|)$ ($O(|V|(|E|+|V|\log|V|)\log|V|)$) time 2-approximation algorithm for any given graph G'.

Next, known results for 2VCA-SV (2ECA-SV, respectively) are summarized. Since 2VCA (2ECA) is NP-hard, so is 2VCA-SV (2ECA-SV). For 2VCA-SV (2ECA-SV) [8] ([9]) proposed an $O(|V||E|+|V|^2\log|V|)$ ($O(|V|^2)$) time 2-approximation algorithm when $\kappa(S;G')=1$ ($\lambda(V;G')=1$), that is, G' has a connected component containing S . In [8] ([9]) 2VCA-SV (2ECA-SV) is reduced to 2VCA (2ECA), which is to be solved by the approximation algorithm of [5] ([4]). Some approximation algorithms proposed for more general problems containing 2VCA-SV (2ECA-SV) as a subproblem can be used. If the $O(|V|^3+|V||E|\alpha(|V|,|V|))$ ($O(|V|^2\log|V|)$) time algorithm proposed in [10] ([11]) is applied then $PR=3$ ($PR=3-3/|S|$), where α is the inverse of Ackermann's function (see [12]). In particular, if $\kappa(S;G')=1$ ($\lambda(S;G')=1$) then $PR=2$ ($PR=2-2/|S|$). And, [13] ([14]) proposed a polynomial time ($O(|V|^{10}|E|^7)$ time) 2-approximation algorithm based on a linear programming technique. (In [13] the exact time complexity is not shown.)

[Main results] In this paper, we propose a $(2-2/|L|)$-approximation algorithm *R2VS* or *R2ES* for 2VCA-SV or 2ECA-SV, respectively, for the case when $\kappa(S;G')=\lambda(S;G')=1$, where L is the set of leaves of a certain tree (called a path tree) constructed from G' and S. Its time or space complexity is $O(|V||E|+|V|^2\log|V|+|L||V|^2)$ or $O(|V|^2)$, respectively.

2 Preliminaries

2.1 Basic Definitions

A *graph* $G=(V,E)$ consists of a finite and nonempty set of vertices V and a finite set of edges E. V and E are often written as $V(G)$ and $E(G)$, respectively. A directed graph is often written as $\overrightarrow{G}=(V,\overrightarrow{E})$. In an undirected graph, an edge with endvertices u, v is denoted by (u, v). In a directed graph, a direct edge which leaves u and enters v is denoted by $\langle u, v\rangle$. In the two cases, vertices u and v are said to be *adjacent*. For $\langle u, v\rangle$, u is called the *parent* of v and v is called the *child* of u. For two sets P and Q, let $P - Q = \{x \in P \mid x \notin Q\}$. For $V_1 \subset V$ and $E_1 \subset E$, let $G - (V_1 \cup E_1) = (V - V_1, E - (E_1 \cup E(V_1)))$, where $E(V_1) = \{(u, v) \in E \mid \{u, v\} \cap V_1 \neq \emptyset\}$. $G - \{x\}$ is simply denoted as $G - x$.

For an undirected graph (a directed graph, respectively) G and vertices $u, v \in V$, an undirected path (a directed path) from u to v is denoted by $P(u, v; G)$ ($P\langle u, v; G\rangle$). If no confusion occurs then $P(u, v; G)$ ($P\langle u, v; G\rangle$) is simply represented as $P(u, v)$ ($P\langle u, v\rangle$).

A directed graph G is *weakly connected* or a *weakly connected graph* if and only if, for any two vertices u and v, there is a weakly directed path (that is, if any directed edge is replaced by an undirected one then it is a path) from u to v in G. And, G is *strongly connected* or a *strongly connected graph* if and only if, for any two vertices u and v, there is a directed path from u to v.

An undirected graph G is *connected* or a *connected graph* if, for any two vertices u and v, there is a path from u to v. A *connected component* of a graph is the vertex set of a maximal connected subgraph.

If two paths P, P' do not share any vertex except endpoints (any edge, respectively) then they are called *internally disjoint* (*edge disjoint*). Let $\kappa(u, v; G)$ ($\lambda(u, v; G)$) denote the maximum number of pairwise internally disjoint (edge disjoint) paths. Let us denote $\kappa(S; G) = \min\{\kappa(u, v; G) \mid u, v \in S\}$ and $\lambda(S; G) = \min\{\lambda(u, v; G) \mid u, v \in S\}$ for a set of vertices $S \subseteq V$. If $S = V$ then we simply represent as $\kappa(G)$ ($\lambda(G)$). A *k-vertex-connected graph* (*k-edge-connected graph*) is a graph such that $\kappa(G) \geq k$ ($\lambda(G) \geq k$). That is to say, the graph has at least k internally (edge) disjoint paths between any pair of vertices.

A *k-vertex-connected components* (*k-edge-connected components*, respectively) of G is a maximal subset $Q \subseteq V$ with $\kappa(Q; G) \geq k$ ($\lambda(Q; G) \geq k$). In particular, 2-vertex-connected components are often called *blocks*. A vertex v (An edge e) is called a *cutvertex* (a *bridge*) of G if the number of connected components of $G - v$ ($G - e$) is greater than that of G.

A *tree* is an undirected connected acyclic graph. A *leaf* in a tree is a vertex to which only one edge is incident. An *arborescence* is a weakly connected acyclic directed graph such that it has only one specified vertex r, called the *root*, having no entering edges, and for any vertex v except r, there is a path $P\langle r, v\rangle$ and v has exactly one entering edge. For an arborescence with the root r, a graph consisting of edges $\langle u, v\rangle$ such that the oppositely directed edges $\langle v, u\rangle$ are contained in the arborescence is called a *reverse arborescence* with the root r, where a root in reverse arborescence is a vertex having no leaving edges. If there is $P\langle u, v\rangle$ in a (reverse) arborescence, then we say that u is an *ancestor* of v and v is a *descendant* of u. For a reverse arborescence with the root r, suppose that u is not a descendant of v and that v is not a descendant of u. Then a *nearest common descendant* of u and v is a common descendant of u and v such that it is the nearest among all such common descendants. A cutvertex (leaf, respectively) in a directed tree means a cutvertex (leaf) in the undirected tree which is obtained by replacing each edge $\langle u, v\rangle$ by an undirected one (u, v).

A real number assigned to each edge is called the (edge) *weight* of the edge, and a function assigning each edge a weight is called a *weight function*. A graph with edge weights is called a *weighted graph*. In a weighted directed graph G, a *minimum arborescence* is a spanning arborescence whose total weight is the minimum among all arborescences of G. A *shortest path* from u to v of an undirected graph G is an undirected path $P(u, v; G)$ whose total sum of weights is the minimum among all such paths of G.

2.2 A block-cutvertex-tree

As an instance of 2VCA-SV, suppose that $G' = (V, E')$ is a graph such that $|V| \geq 3$ and let S be the set of specified vertices (see Fig. 1). We focus on blocks and cutvertices of G', where if $|V' \cap S| \leq 1$ for any connected component V' in G' then we regard V' as a block of G' (such as the connected component $\{o, p\}$ in Fig. 1).

We construct a *block-cutvertex-forest* $T_B = (V_B, E'_B)$ from G' as follows (see Fig. 2). Any block or any cutvertex in G' is represented as a new vertex, called a block-vertex or a cutvertex, respectively. Let V_{b_B} or V_{c_B} be the set of block-vertices or of cutvertices (given as new vertices), respectively, and denote as $V_B = V_{b_B} \cup V_{c_B}$. Let E'_B be the set of edges (u, v) such that u is a block-vertex and v represents an individual cutvertex contained in the block corresponding to u in G'.

T_B is a forest such that block-vertices and cutvertices appear alternately in each tree of T_B. T_B is called a *block-cutvertex-tree* if G' is connected. Note that any leaf of T_B is a block-vertex. A block-cutvertex-forest of G' can be constructed in $O(|V| + |E'|)$ time [15].

For each vertex u of T_B, let $\alpha^{-1}(u)$ be the set of vertices of G' satisfying the following (i) or (ii). (i) If u is a cutvertex of T_B then $\alpha^{-1}(u) = \{u'\}$, where u' is a cutvertex of G' that corresponds to u. (ii) If u is a block-vertex of T_B then $\alpha^{-1}(u) = B_u - V_c$, where B_u is a block corresponding to u (see sets represented as $\{\ldots\}$ in Fig. 2) and V_c is the set of cutvertices of G'. For any pair of vertices $u, v \in V_B$, let E''_B be the set of edges (u, v) such that there exists an edge whose endvertices belong to $\alpha^{-1}(u)$ and $\alpha^{-1}(v)$ in $G - E'$. Let $E_B = E'_B \cup E''_B$ and $G_B = (V_B, E_B)$ (Fig. 2). Each vertex $u \in V_B$ is often denoted as $\alpha(x)$ for some $x \in \alpha^{-1}(u)$.

Let T_P be any subtree of T_B, and let T_X denote T_B or T_P. Let V_{c_X} be the set of cutvertices of T_X. For T_X and a set of edges E''', we write as $\kappa'(T_X + E''') \geq 2$ to mean that, for any cutvertex $v \in V_{c_X}$, $\kappa((T_X + E''') - v) \geq 1$.

For a set of vertices $S' \subseteq V(T_X)$, let us denote as $\kappa'(S'; T_X + E''') \geq 2$ if and only if, for any cutvertex $v \in V_{c_X}$, $\kappa(S'; (T_X + E''') - v) \geq 1$. Note that there may exist a block vertex $v \in V_{b_B}$ such that $\kappa(S'; (T_X + E''') - v) = 0$ even if $\kappa'(S'; T_X + E''') \geq 2$. In [4, 5] ([9], respectively) it has already been proved that if $\kappa'(T_X + E''') \geq 2$ ($\kappa'(S'; T_X + E''') \geq 2$) then we can obtain a set of edges E'' from E''' such that $\kappa(G' + E'') \geq 2$ ($\kappa(S; G' + E'') \geq 2$).

2.3 A path-tree

Suppose that $\kappa(S; G') = 1$. Let $S_b = \{u \in V_{b_B} \mid (\alpha^{-1}(u) - V_c) \cap S \neq \emptyset\}$, $S_c = \{v \in V_{c_B} \mid \alpha^{-1}(v) \subseteq S\}$ and $S_B = S_b \cup S_c$. Let T'_B be a subgraph consisting of those edges on $P(u, v; T_B)$ for any pair of $u, v \in S_B$ (denoted by thick bold lines in Fig. 2). Clearly, T'_B is a tree. Let $T_P = (V_P, E'_P)$ be the tree constructed from T'_B by deleting all leaves $v \in V_{c_B} \cap S_B$ (see c_6 in Fig. 2). T_P is called the *path-tree* (of G')[9]. Let us partition V_P as $V_P = V_{b_P} \cup V_{c_P}$, where V_{b_P} and V_{c_P} are sets of block-vertices and of cutvertices of T_P, respectively. Let $S_P = (S_B - C_L) \cup \{x \in V_{b_P} \mid x$ is adjacent to $u \in C_L$ in $T'_B\}$, where C_L is the set of cutvertices deleted in constructing T_P. (In Fig. 2, c_6 is deleted from S_B and b_3 is added to S_P.) Note that any leaf of T_P is contained in $S_P \cap V_{b_P}$ (see Fig. 3).

3 The 2-vertex-connectivity augmentation problem for specified vertices (2VCA-SV)

3.1 The proposed algorithm *R2VS* for $\kappa(S; G') = 1$

The algorithm *2-ABIS* of [16] or the algorithm *R2VS* to be proposed is a 2-approximation or $(2 - 2/|L|)$-approximation one, respectively, for 2VCA-SV when $\kappa(S; G') = 1$. *2-ABIS* utilizes the algorithm of [5] for solving 2VCA, while *R2VS* repeats *2-ABIS* as follows: it repeats selecting each leaf of a given path tree T_P and executing the algorithm of [5] to solve 2VCA, and then selects the best solution among those obtained. The proposed algorithm *R2VS* is stated in the following.

[Algorithm *R2VS*]

Input: An undirected graph $G = (V, E)$, a spanning subgraph $G' = (V, E')$ of G with $\kappa(S; G') = 1$ (Fig. 1), a set of specified vertices $S \subseteq V$, and a weight function $w : E \rightarrow R^+$ (nonnegative real numbers)

Output: A set of edges $E'' \subseteq E - E'$ with $\kappa(S; G' + E'') \geq 2$.

1. Construct a block-cutvertex-forest $T_B = (V_B, E'_B)$ (Fig. 2) of G' and a path-tree $T_P = (V_P, E'_P)$ (Fig. 3) from T_B. Let E''_B be the set of edges (u, v) such that $u, v \in V_B$ and there is an edge $(u', v') \in E - E'$ with $u' \in \alpha^{-1}(u)$ and $v' \in \alpha^{-1}(v)$. Let $E_B = E'_B \cup E''_B$, $G_B = (V_B, E_B)$ and $w_B((u, v)) = min\{w((u', v')) \mid (u', v') \in E - E',\ u' \in \alpha^{-1}(u),\ v' \in \alpha^{-1}(v)\}$ for any $(u, v) \in E''_B$, where $w_B((u, v))$ may be undefined for some (u, v).
2. Set $E_P \leftarrow E'_P$ and construct $G_P = (V_P, E_P)$ as follows: for any pair of vertices $u, v \in V_P$ with $(u, v) \notin E'_P$, if $G_B - E'_P$ has a path $P(u, v)$ such that $(V(P(u, v)) - \{u, v\}) \cap V_P = \emptyset$ then set $E_P \leftarrow E_P \cup \{(u, v)\}$ and let $w_P((u, v))$ be the shortest length of such paths with respect to the weight function w_B, where we set $w_B((u, v)) \leftarrow 0$ for any $(u, v) \in E'_B - E'_P$. (See $E_P - E'_P$ whose edges are denoted by dotted lines in Fig. 3.)
3. Let $L = \{\rho_1, \cdots, \rho_{|L|}\}$ be the set of leaves of T_P. Set $E'' \leftarrow \emptyset$ and $i \leftarrow 1$ initially and repeat the following Steps 4 through 8.
4. Select a leaf $\rho_i \in L$ as the root and construct from T_P a reverse arborescence $\overrightarrow{T'_i} = (V_P, \overrightarrow{E'_i})$ with the root ρ_i (see solid arrows in Figs. 4 and 7).
5. Set $\overrightarrow{E_i^+} \leftarrow \emptyset$ initially, and define $\overrightarrow{E_i^+}$ and $w_i : \overrightarrow{E_i^+} \rightarrow R^+$ by executing the following (1) through (3) for each edge $(u, v) \in E_P$.
   - (1) If $\langle u, v\rangle \in \overrightarrow{E'_i}$ then set $\overrightarrow{E_i^+} \leftarrow \overrightarrow{E_i^+} \cup \{\langle u, v\rangle\}$ and $w_i(\langle u, v\rangle) \leftarrow 0$; otherwise execute the following (2) or (3).
   - (2) If u is an ancestor of v in $\overrightarrow{T'_i}$ then set $\overrightarrow{E_i^+} \leftarrow \overrightarrow{E_i^+} \cup \{\langle v, u\rangle\}$ and $w_i(\langle v, u\rangle) \leftarrow w_P((u, v))$.
   - (3) Otherwise (that is, any one of $\{u, v\}$ is not an ancestor of the other), set $\overrightarrow{E_i^+} \leftarrow \overrightarrow{E_i^+} \cup \{\langle t, u\rangle, \langle t, v\rangle, \langle u, v\rangle, \langle v, u\rangle\}$ and $w_i(\langle t, u\rangle) = w_i(\langle t, v\rangle) = w_i(\langle u, v\rangle) = w_i(\langle v, u\rangle) \leftarrow w_P((u, v))$, where t is the nearest common descendant of u and v in $\overrightarrow{T'_i}$. (For example, in Fig. 4, if $u = b_7$ and $v = b_8$ then $t = b_6$).
6. $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i^+}$ initially, and construct $\overrightarrow{E_i}$ as follows (see Figs. 5 and 8): for each edge $\langle u, v\rangle \in \overrightarrow{E_i^+} - \overrightarrow{E'_i}$ such that u is a cutvertex and v is an ancestor of u in $\overrightarrow{T'_i}$, set $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i} \cup \{\langle u_v, v\rangle\} - \{\langle u, v\rangle\}$ and $w_i(\langle u_v, v\rangle) \leftarrow w_i(\langle u, v\rangle)$, where u_v is the parent (a block vertex) of u on $P(v, u)$ in $\overrightarrow{T'_i}$. (For example, in Fig. 5, $\langle c_2, b_8\rangle \in \overrightarrow{E_i^+} - \overrightarrow{E'_i}$ is changed to $\langle b_6, b_8\rangle \in \overrightarrow{E_i}$, where $u_v = b_6$ if $u = c_2$.)
7. Find a minimum arborescence $\overrightarrow{T''_i} = (V_P, \overrightarrow{A_i})$ with the root ρ_i in $\overrightarrow{G_i} = (V_P, \overrightarrow{E_i})$ (see Figs. 6 and 9). Set $\overrightarrow{E''_i} \leftarrow \overrightarrow{A_i} - \overrightarrow{E'_i}$. Construct $E''_i \subseteq E - E'$ by replacing each edge of $\overrightarrow{E''_i}$ by the corresponding undirected edge of G, where multiple edges are changed to a simple one.

8. If $E'' = \emptyset$ or $w(E'') > w(E_i'')$ then $E'' \leftarrow E_i''$. Set $i \leftarrow i+1$. If $i \leq |L|$ then go back to Step 4. □

The correctness and the performance ratio of the algorithm *R2VS* are going to be shown later. Here we consider its time complexity. The running time spent by all steps except the loop from Steps 4 through 8 is $O(|V||E|+|V|^2\log|V|)$ from the known results (see [8] for example). The loop is repeated $|L|$ times and each iteration of the loop takes at most $O(|\overrightarrow{E_i}|+|V_P|\log|V_P|)$ time [5]. Since $|V_P|$ is $O(|V|)$ and $|\overrightarrow{E_i}|$ is $O(|V|^2)$, the total time spent by the loop from Steps 4 through 8 is $O(|L||V|^2)$ time. Thus, time complexity of the algorithm is $O(|V||E|+|V|^2\log|V|+|L||V|^2)$.

3.2 Correctness and performance ratio of *R2VS*

Theorem 3.1 $\kappa(S;G'+E'') \geq 2$.

(Proof) The theorem follows from the results of [8, 16]. □

We consider the relationship between the total weight of $\overrightarrow{E_i''}$ and that of an optimum solution $E^* \subseteq E - E'$. Let $E_B^* = \{(\alpha(u),\alpha(v)) \mid (u,v) \in E^*\}$. We may assume that there are no multiple edges in E_B^* and that $E_B^* \subseteq E_B - E_B'$. Since E^* is an optimum solution, $\kappa'(S_B;T_B+E_B^*) \geq 2$ and $w_B(E_B^*) = w(E^*)$. Thus, we consider E_B^* instead of E^* in the rest of the section.

Let $\overline{T_B} = (\overline{V_B},\overline{E_B'})$ be any fixed minimal subgraph of T_B such that $E_P' \subseteq \overline{E_B'}$ and $\kappa'(S_P;\overline{T_B}+E_B^*) = 2$. (In Fig. 2, for example, $(b_3,c_6) \in \overline{E_B'}$, while $(c_6,b_9) \notin \overline{E_B'}$: even if there were an edge $(b_{10},c_1) \in E_B'$, it would not in $\overline{E_B'}$.) We partition $E_B^* \cup (\overline{E_B'} - E_P')$ into the three sets, E_B^1, E_B^2 and E_B^3, as follows:

$E_B^1 = \{(u,v) \in E_B^* \cup (\overline{E_B'} - E_P') \mid u,v \in \overline{V_B} - V_P\}$;

$E_B^2 = \{(u,v) \in E_B^* \cup (\overline{E_B'} - E_P') \mid u \in \overline{V_B} - V_P, v \in V_P\}$;

$E_B^3 = \{(u,v) \in E_B^* \cup (\overline{E_B'} - E_P') \mid u,v \in V_P\}$.

Let Z_j denote any connected component of $(\overline{V_B} - V_P, E_B^1)$. Since E^* is an optimum solution, the graph whose vertex set is Z_j and whose edge set is $E_{Z_j} = \{(u,v) \in E_B^1 \mid u,v \in Z_j\}$ is a tree. Let $\mathcal{T}_j$ denote the graph induced by the edge set $E_{Z_j} \cup E_B^2(i)$, where $E_B^2(i) = \{e \in E_B^2 \mid e \text{ is incident to a vertex of } Z_j\}$. $\mathcal{T}_j$ is also a tree and is called a *tent*. Let F_j denote the subtree of T_P such that it consists of all paths $P(u,v;T_P)$, one path for each pair $u,v \in V(\mathcal{T}_j) \cap V_P$. F_j is called a *floor*. Let N_j denote the graph consisting of a tent $\mathcal{T}_j$ and a floor F_j (Fig. 10(1), (2)). N_j is called a *net*. For E_B^3, let us define a tent to be a graph consisting of an individual edge of E_B^3 and its endvertices. A floor and a net are defined similarly (Fig. 10(3), (4)). Let n be the number of nets. For each j, the subgraph of $\overrightarrow{T_i}$ corresponding to F_j is denoted as $\overrightarrow{F_j}$. $\overrightarrow{F_j}$ is also called a floor. $\overrightarrow{F_j}$ has exactly one vertex having no leaving edges. The vertex is called the root of $\overrightarrow{F_j}$ and denoted by r_j. Let x_j be a vertex defined as follows: if r_j is a block vertex then set $x_j \leftarrow r_j$, or if r_j is a cutvertex then select any block-vertex u that is a parent of r_j in $\overrightarrow{F_j}$ and set $x_j \leftarrow u$ (see x_j in Fig. 10).

Lemma 3.1 *For each net N_j, a set of directed edges $\overrightarrow{\mathcal{A}_j} \subseteq \overrightarrow{E_i} - \overrightarrow{E_i'}$ satisfying the following (a) and (b) can be constructed from E_B^*: (a) all vertices of $\overrightarrow{F_j}$ are reachable from x_j in $\overrightarrow{F_j} + \overrightarrow{\mathcal{A}_j}$; (b) If r_j is the leaf of $\overrightarrow{F_j}$ then $w_i(\overrightarrow{\mathcal{A}_j}) \leq (2-2/p_j)w_B(E(\mathcal{T}_j))$ else $w_i(\overrightarrow{\mathcal{A}_j}) \leq 2w_B(E(\mathcal{T}_j))$, where p_j is the number of leaves of F_j.*

(Proof) Let s_j be a vertex defined as follows: if r_j is a leaf of $\overrightarrow{F_j}$ then set $s_j = r_j$, or if r_j is not a leaf of $\overrightarrow{F_j}$ then select any vertex s_j in $\mathcal{T}_j$. For each $\mathcal{T}_j$, we execute the depth-first-search (DFS) by selecting s_j as the starting vertex, and assign the DFS-number to each vertex. Suppose that L_j be the set of leaves of F_j and $p_j = |L_j|$. Note that $L \cap V(F_j) \subseteq L_j$ and that $p_j \geq 2$ since $E(F_j) \neq \emptyset$. Then, leaves of $\overrightarrow{F_j}$ are numbered as $l_1^{(j)}, l_2^{(j)}, \cdots, l_{p_j}^{(j)}$, whose indices denote the order in which they are visited by DFS, where $l_1^{(j)}$ may be often denoted as $l_{p_j+1}^{(j)}$ for notational simplicity. Each of p_j paths $P(l_k^{(j)}, l_{k+1}^{(j)};\mathcal{T}_j)$ $(1 \leq k \leq p_j)$ is called a *bypass* (connecting $l_k^{(j)}$ and $l_{k+1}^{(j)}$). $P(l_k^{(j)}, l_{k+1}^{(j)};\mathcal{T}_j)$ is often represented as $P_k^{(j)}$ for simplicity. Then the following (1) and (2) hold.

(1) For each $e \in E(\mathcal{T}_j)$, there are exactly two bypasses containing e.

(2) For some k' with $1 \leq k' \leq p_j$, there is at least one weakly connected path $P(l_{k'}^{(j)}, l_{k'+1}^{(j)};\overrightarrow{F_j})$ containing r_j and x_j, where $l_{k'+1}^{(j)}$ is an ancestor of x_j in $\overrightarrow{T_i}$.

Let us define a set of directed edges $\overrightarrow{\mathcal{A}_j}$ as in the following (i) and (ii) by appropriately choosing one weakly connected path $P(l_{k'}^{(j)}, l_{k'+1}^{(j)};\overrightarrow{F_j})$ containing r_j and x_j (x_j may be equal to r_j) in $\overrightarrow{F_j}$.

(i) If r_j is a leaf of $\overrightarrow{F_j}$ (see Fig. 10(1), (3)) then $l_1^{(j)} = r_j$. First, let

$$\omega_k^{(j)} = \sum_{(u,v)\in E(P_k^{(j)})} w_B((u,v)), \quad \text{and}$$

$$\omega_{k''}^{(j)} = \max\{\omega_k^{(j)} \mid 1 \leq k \leq p_j\}.$$

(Note that $\omega_{k''}^{(j)} \geq \frac{1}{p_j}\sum_{k=1}^{p_j}\omega_k^{(j)}$.)

If $k'' \neq 1$ and $k'' \neq p_j$ then let

$$\begin{aligned}\overrightarrow{\mathcal{A}_j} \quad = \quad & \{\langle x_j, l_2^{(j)}\rangle, \langle x_j, l_{p_j}^{(j)}\rangle\} \cup \\ & \{\langle l_m^{(j)}, l_{m+1}^{(j)}\rangle \mid 2 \leq m \leq k''-1\} \cup \\ & \{\langle l_{m'+1}^{(j)}, l_{m'}^{(j)}\rangle \mid k''+1 \leq m' \leq p_j-1\}\end{aligned}$$

(Fig. 10(1) assumes $l_{k''} = l_3$)

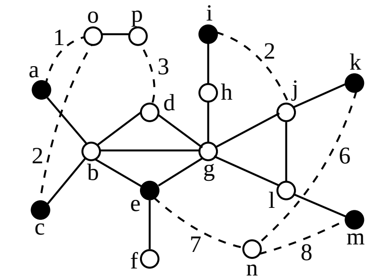

Figure 1: $G = (V, E)$ and $G' = (V, E')$, where solid lines are edges in E', dotted lines are ones in $E - E'$, black vertices are those of $S \subseteq V$, and numbers shown beside edges are weights.

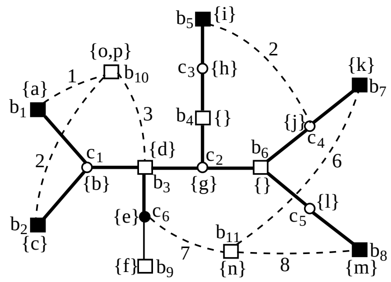

Figure 2: $G_B = (V_B, E_B)$ and a block-cutvertex-forest $T_B = (V_B, E'_B)$ (edges are denoted by solid lines), where dotted lines are edges in $E_B - E'_B$, squares are block-vertices, circles are cutvertices, black squares and circles are vertices in S_B, and $\alpha^{-1}(u)$ for each $u \in V_B$ is shown in braces.

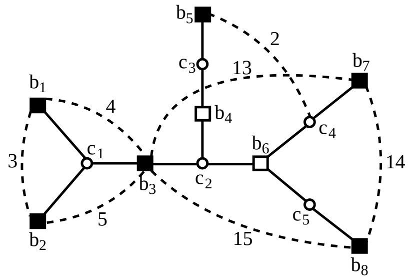

Figure 3: $G_P = (V_P, E_P)$ (to be defined in Step 2 of *R2VS*) and the path-tree $T_P = (V_P, E'_P)$, where dotted lines are edges in $E_P - E'_P$ and black squares and circles are vertices in S_P.

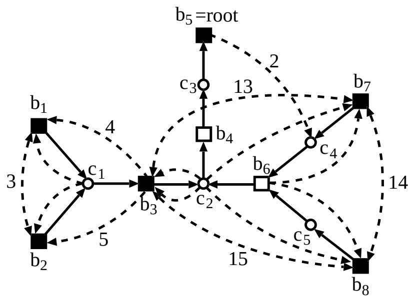

Figure 4: Construction of $\vec{T}_i$ (solid arrows) and $\vec{E}_i^+$ with the root b_5, where dotted lines denote edges appeared in Step 5 (2) or (3)).

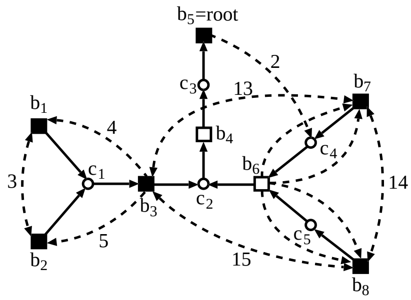

Figure 5: Constructing $\vec{E}_i$ from $\vec{E}_i^+$. In Step 6, directed edges emanating from c_2 in Fig. 4 are transformed into those which start from b_6 with resulting self-loops deleted.

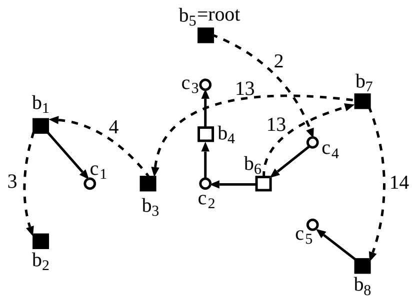

Figure 6: A minimum arborescence $\vec{T}'_i$ (solid lines are edges of $\vec{E}'_i$, and dotted lines are those of $\vec{E}''_i$) with the root b_5. The total weight is 49.

If $k'' = 1$ (see Fig. 10(3)) then let

$$\begin{aligned}\overrightarrow{\mathcal{A}_j} &= \{\langle x_j, l_{p_j}^{(j)}\rangle\} \cup \\ &\quad \{\langle l_{m'+1}^{(j)}, l_{m'}^{(j)}\rangle \mid 2 \leq m' \leq p_j - 1\}\end{aligned}$$

If $k'' \neq 1$ and $k'' = p_j$ then let

$$\begin{aligned}\overrightarrow{\mathcal{A}_j} &= \{\langle x_j, l_2^{(j)}\rangle\} \cup \\ &\quad \{\langle l_m^{(j)}, l_{m+1}^{(j)}\rangle \mid 2 \leq m \leq p_j - 1\}.\end{aligned}$$

(ii) If r_j is not a leaf of $\overrightarrow{F_j}$ then let

$$\begin{aligned}\overrightarrow{\mathcal{A}_j} &= \{\langle x_j, l_{k'+1}^{(j)}\rangle\} \cup \\ &\quad \{\langle l_m^{(j)}, l_{m+1}^{(j)}\rangle \mid 1 \leq m \leq p_j, m \neq k'\}\end{aligned}$$

($l_{k'} = l_8$ and $l_{k'+1} = l_9$ in Fig. 10(2), and $l_{k'} = l_1$ and $l_{k'+1} = l_2$ in Fig. 10(4)).

The following important points hold from the definitions of $\overrightarrow{\mathcal{A}_j}$, G_P and w_P:

(a) all vertices of $\overrightarrow{F_j}$ are reachable from x_j in $\overrightarrow{F_j} + \overrightarrow{\mathcal{A}_j}$;
(b) $\overrightarrow{\mathcal{A}_j} \subseteq \vec{E}_i - \vec{E}'_i$;
(c) $w_P((l_k^{(j)}, l_{k+1}^{(j)})) \leq \omega_k^{(j)}$ for each k ($1 \leq k \leq p_j$).

Now, if r_j is a leaf of $\overrightarrow{F_j}$, the condition (1) mentioned above in this proof and the fact that $w_i(\langle x_j, l_2^{(j)}\rangle) = w_P((l_1^{(j)}, l_2^{(j)}))$ and $w_i(\langle x_j, l_{p_j}^{(j)}\rangle) = w_P((l_1^{(j)}, l_{p_j}^{(j)}))$ (this follows from Steps 5(2) and 6 of *R2VS*) show the following inequality

$$\begin{aligned}w_i(\overrightarrow{\mathcal{A}_j}) &\leq \sum_{k=1, k\neq k''}^{p_j} w_P((l_k^{(j)}, l_{k+1}^{(j)})) \\ &\leq \left(1 - \frac{1}{p_j}\right) \sum_{k=1}^{p_j} \omega_k^{(j)} \\ &= \left(1 - \frac{1}{p_j}\right) \cdot 2 \sum_{(u,v)\in E(\mathcal{T}_j)} w_B((u, v)) \\ &= \left(2 - \frac{2}{p_j}\right) w_B(E(\mathcal{T}_j)).\end{aligned}$$

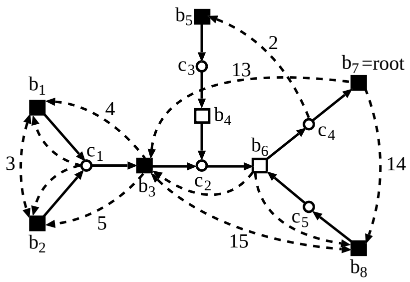

Figure 7: Construction of $\vec{T_i}$ (solid arrows) and $\vec{E_i^+}$ with the root b_7, where dotted lines denote edges appeared in Step 5 (2) or (3)).

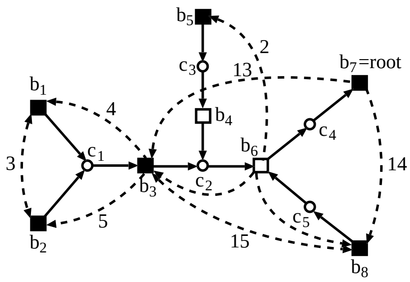

Figure 8: Constructing $\vec{E_i}$ from $\vec{E_i^+}$. In Step 6, directed edges emanating from c_4 in Fig. 7 are transformed into those which start from b_6.

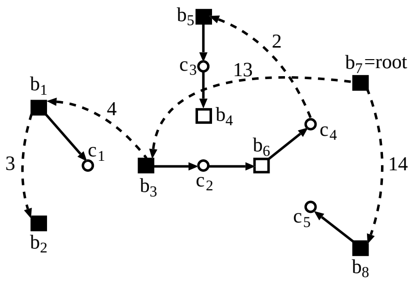

Figure 9: A minimum arborescence $\vec{T_i'}$ (solid lines are edges of $\vec{E_i'}$, and dotted lines are those of $\vec{E_i''}$) with the root b_7. The total weight is 36.

If r_j is not a leaf of $\overrightarrow{F_j}$ then $w_i(\langle x_j, l^{(j)}_{k'+1}\rangle) = w_P((l^{(j)}_{k'}, l^{(j)}_{k'+1}))$ (this follows from Steps 5(3) and 6 of *R2VS*) and, therefore,

$$
\begin{aligned}
w_i(\overrightarrow{\mathcal{A}_j}) &\leq \sum_{k=1}^{p_j} w_P((l^{(j)}_k, l^{(j)}_{k+1})) \\
&\leq \sum_{k=1}^{p_j} \omega^{(j)}_k \\
&\leq 2w_B(E(\mathcal{T}_j)). \qquad \square
\end{aligned}
$$

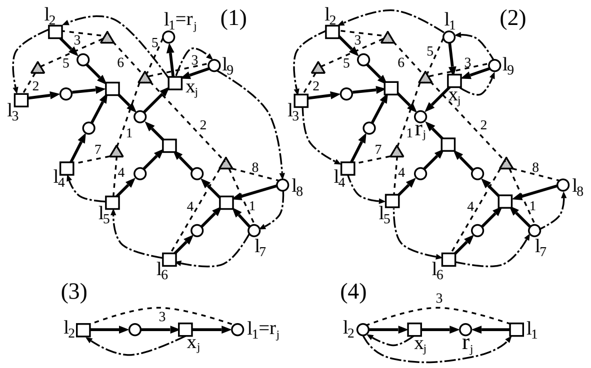

Figure 10: Four examples (1) through (4) of nets N_j, where triangles are vertices in $\overline{V_B} - V_P$, and solid lines, dotted lines and partially broken lines represent a floor $\vec{F_j}$, a tent $\mathcal{T}_j$ and edges of $\vec{\mathcal{A}}_j$, respectively.

Lemma 3.2 *For some h with $1 \leq h \leq |L|$, a set of directed edges $\overrightarrow{B_h} \subseteq \overrightarrow{E_h} - \overrightarrow{E_h'}$ satisfying the following (a) and (b) can be constructed from E_B^*: (a) $\overrightarrow{T_h} + \overrightarrow{B_h}$ is strongly connected; (b) $w_h(\overrightarrow{B_h}) \leq (2-2/|L|)w_B(E_B^*)$.*

(Proof) We prove that the desired set $\overrightarrow{B_h}$ is obtained from E_B^*. First, for each i with $1 \leq i \leq |L|$, we show how to construct a set of directed edges $\overrightarrow{B_i}$ such that $\overrightarrow{T_i} + \overrightarrow{B_i}$ is strongly connected. Initially set $\overrightarrow{B_i} \leftarrow \emptyset$ and $\mathcal{E}_B^* \leftarrow E_B^* \cup (\overline{E_B'} - E_P')$, and assign "*accessible*" to the root ρ_i of the path-tree $\overrightarrow{T_i} = (V_P, \overrightarrow{E_i'})$ and "*nonaccessible*" to the other vertices of $\overrightarrow{T_i}$. Repeat the following pair of procedures **AL1** and **AL2** until "*accessible*" is assigned to every vertex of V_P:

(AL1) Select an accessible block-vertex x satisfying the following (1) and (2):

(1) x is in a net N_j constructed from $\mathcal{E}_B^*$;

(2) in $\overrightarrow{F_j}$, if r_j is a block-vertex then $x \leftarrow r_j$, otherwise x is set to a parent of r_j.

(AL2) For x and N_j, construct $\overrightarrow{\mathcal{A}_j}$ by using Lemma 3.1 (in which x is written as x_j), and set

$$\overrightarrow{B_i} \leftarrow \overrightarrow{B_i} \cup \overrightarrow{\mathcal{A}_j}.$$

Then assign "*accessible*" to all vertices of the floor $\overrightarrow{F_j}$ of the net N_j, and set $\mathcal{E}_B^* \leftarrow \mathcal{E}_B^* - E(\mathcal{T}_j)$.

If we assume that there are no block-vertices x satisfying (1) and (2) of **AL1**, while we have a nonaccessible vertex, then we can easily show a contradiction that $\kappa'(S_B; T_B + E_B^*) \geq 2$ is not satisfied. Hence it is concluded that "*accessible*" is assigned to every vertex eventually, implying that $\overrightarrow{T_i} + \overrightarrow{B_i}$ is strongly connected.

Next we show that there is a direct edge set $\overrightarrow{B_h}$ with $w_h(\overrightarrow{B_h}) \leq (2 - 2/|L|)w(E_B^*)$. Let h be an index such that $w_h(\overrightarrow{B_h}) = min\{w_i(\overrightarrow{B_i}) \mid 1 \leq i \leq |L|\}$. For any $r \in L_j - L$, there is an edge $(r,u) \in E(F_j)$ such that $T_P - (r,u)$ has a connected component T' with $V(T') \cap V(F_j) = \{r\}$ and $L \cap V(T') \neq \emptyset$. Hence there is a reverse arborescence, rooted at some $r' \in L \cap V(T')$, such that any path $P\langle u, r'\rangle$ with $u \in V(F_j)$ passes through r toward r', meaning that r is a root of $\overrightarrow{F_j}$. That is, each vertex in L_j becomes the root of $\overrightarrow{F_j}$ at least once in $\overrightarrow{T_1}, \cdots, \overrightarrow{T_{|L|}}$. Since $E(\mathcal{T}_x) \cap E(\mathcal{T}_y) = \emptyset$ if $x \neq y$, Lemma 3.1 gives us

$$
\begin{aligned}
w_h(\overrightarrow{B_h}) &\leq \frac{1}{|L|}\sum_{i=1}^{|L|} w_i(\overrightarrow{B_i}) \\
&\leq \frac{1}{|L|}\sum_{j=1}^{n}\Big(p_j \cdot \Big(2 - \frac{2}{p_j}\Big) w_B(E(\mathcal{T}_j)) \\
&\qquad + (|L| - p_j) \cdot 2w_B(E(\mathcal{T}_j))\Big)
\end{aligned}
$$

$$= \frac{1}{|L|} \cdot 2(|L|-1)\sum_{j=1}^{n} w_B(E(\mathcal{T}_j))$$
$$= \Big(2 - \frac{2}{|L|}\Big) w_B(E_B^*). \qquad \square$$

Since $\overrightarrow{T_h} + \overrightarrow{B_h}$ is strongly connected, it has a subgraph $T_H = (V_P, \overrightarrow{E_H})$ which is an arborescence rooted at ρ_h. Since $w_h(\langle u, v\rangle) = 0$ for any directed edge $\langle u, v\rangle \in \overrightarrow{E'_h}$, we have $w_h(\overrightarrow{E_H}) \le w_h(\overrightarrow{B_h})$. (Note that $\overrightarrow{B_h}$ may have some edges not contained in $\overrightarrow{E_H}$.) Hence we obtain the next corollary.

Corollary 3.1 *$\overrightarrow{T_h} + \overrightarrow{B_h}$ of Lemma 3.2 contains, as a subgraph, an arborescence $T_H = (V_P, \overrightarrow{E_H})$, with the root ρ_h, such that $w_h(\overrightarrow{E_H}) \le w_h(\overrightarrow{B_h})$.*

We obtain the next theorem from the above discussion.

Theorem 3.2 *If $\kappa(S; G') = 1$ then the proposed algorithm R2VS generates a $(2 - 2/|L|)$-approximate solution to 2VCA-SV in $O(|V||E| + |V|^2 \log |V| + |L||V|^2)$ time, where L is the set of leaves of the path-tree of G'.*

(Proof) Since time complexity of the algorithm has already been given in Section 3.1, we consider the weight of any approximate solution E'' given by *R2VS*. $\overrightarrow{T'_h}$ constructed in Step 7 is a minimum arborescence in a directed graph $(V_P, \overrightarrow{E_h})$ with respect to the weight function w_h. That is, $w_h(\overrightarrow{E''_h}) \le w_h(\overrightarrow{E_H})$. By considering the rooted tree $T_H = (V_P, \overrightarrow{E_H})$ mentioned in Lemma 3.2 and Corollary 3.1, we obtain

$$\begin{aligned} w(E'') &= min\{w(E''_i) \mid 1 \le i \le |L|\} \\ &\le w(E''_h) \le w_h(\overrightarrow{E''_h}) \le w_h(\overrightarrow{E_H}) \\ &\le w_h(\overrightarrow{B_h}) \le (2 - 2/|L|)w(E^*). \quad \square \end{aligned}$$

4 The 2-edge-connectivity augmentation problem for specified vertices (2ECA-SV)

4.1 The proposed algorithm *R2ES* for $\lambda(S; G') = 1$

The algorithm of [9] is a 2-approximation one for 2ECA-SV when $\lambda(G') = 1$ and utilizes the algorithm of [4] for solving 2ECA. On the other hand, our algorithm *R2ES* is a $(2 - 2/|L|)$-approximation one for 2ECA-SV when $\lambda(S; G') = 1$ and utilizes the algorithm of [5] for solving 2ECA in order to improve the time and space complexity. The differences from [5] are the following (1) and (2): (1) we select each leaf of T as the root in Step 4 and execute the algorithm of [5], and then select the best solution ; (2) we have modified construction of $\overrightarrow{E_i}$ in Step 6(3) and add Steps 2, 3 in order to extend 2ECA to 2ECA-SV. The proposed algorithm *R2ES* is stated in the following.

[Algorithm *R2ES*]
Input: An undirected graph $G = (V, E)$, a spanning subgraph $G' = (V, E')$ of G with $\lambda(S; G') = 1$, a set of specified vertices $S \subseteq V$, and a weight function $w : E \to R^+$ (nonnegative real numbers)
Output: A set of edges $E'' \subseteq E - E'$ with $\lambda(S; G' + E'') \ge 2$.

1. Construct a graph $G'_s = (V_s, E'_s)$ from G' by shrinking each 2-edge-connected component of G' into an individual vertex, where any connected component not containing S is regarded as a 2-edge-connected component in this construction. For $u, v \in V_s$, let $E''_s = \{(u, v) \mid (u', v') \in E - E', u' \in \beta^{-1}(u), v' \in \beta^{-1}(v)\}$, $w_s((u, v)) = \min\{w((u', v')) \mid (u', v') \in E - E', u' \in \beta^{-1}(u), v' \in \beta^{-1}(v)\}$, $E_s = E'_s \cup E''_s$ and $G_s = (V_s, E_s)$, where $\beta^{-1}(x)$ is the component represented by $x \in V_s$

2. Let $S' = \{u \in V_s \mid \beta^{-1}(u) \cap S \ne \emptyset\}$ and $T = (V_T, E'_T)$ (called a path-tree) be a subgraph consisting of those edges on $P(u, v; G'_s)$ for any pair of $u, v \in S'$.

3. Set $E_T \leftarrow E'_T$ and construct $G_T = (V_T, E_T)$ as follows: for any pair of vertices $u, v \in V_T$ if $G_s - E'_T$ has a path $P(u, v)$ such that $(V(P(u, v)) - \{u, v\}) \cap V_T = \emptyset$ then set $E_T \leftarrow E_T \cup \{(u, v)\}$ and let $w_T((u, v))$ be the shortest length of such paths with respect to the weight function w_s, where we set $w_s((u, v)) \leftarrow 0$ for any $(u, v) \in E'_s - E'_T$.

4. Let $L = \{\rho_1, \cdots, \rho_{|L|}\}$ be the set of leaves of T. Set $E'' \leftarrow \emptyset$ and $i \leftarrow 1$ initially and repeat the following Steps 5 through 8.

5. Select a leaf ρ_i of T as the root and construct from T a reverse arborescence $\overrightarrow{T_i} = (V_T, \overrightarrow{E'_i})$ with the root ρ_i

6. Set $\overrightarrow{E_i} \leftarrow \emptyset$ initially, and define $\overrightarrow{E_i}$ and $w_i : \overrightarrow{E_i} \to R^+$ by executing the following (1) through (3) for each edge $(u, v) \in E_T$.

   (1) If $\langle u, v\rangle \in \overrightarrow{E'_T}$ then set $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i} \cup \{\langle u, v\rangle\}$ and $w_i(\langle u, v\rangle) \leftarrow 0$; otherwise execute the following (2) or (3).

   (2) If u is an ancestor of v in $\overrightarrow{T_i}$ then set $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i} \cup \{\langle v, u\rangle\}$ and $w_i(\langle v, u\rangle) \leftarrow w_T((u, v))$.

   (3) Otherwise (that is, any one of $\{u, v\}$ is not an ancestor of the other), set $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i} \cup \{\langle t, u\rangle, \langle t, v\rangle, \langle u, v\rangle, \langle v, u\rangle\}$ and $w_i(\langle t, u\rangle) = w_i(\langle t, v\rangle) = w_i(\langle u, v\rangle) = w_i(\langle v, u\rangle) \leftarrow w_T((u, v))$, where t is the nearest common descendant of u and v in $\overrightarrow{T_i}$.

7. Find a minimum arborescence $\overrightarrow{T'_i} = (V_T, \overrightarrow{A_i})$ with the root ρ_i in $\overrightarrow{G_i} = (V_T, \overrightarrow{E_i})$. Set $\overrightarrow{E''_i} \leftarrow \overrightarrow{A_i} - \overrightarrow{E'_i}$.

Construct $E''_i \subseteq E - E'$ by replacing each edge of $\overrightarrow{E''_i}$ by the corresponding undirected edge of G, where multiple edges are changed to a simple one.

8. If $E'' = \emptyset$ or $w(E'') > w(E''_i)$ then $E'' \leftarrow E''_i$. Set $i \leftarrow i+1$. If $i \leq |L|$ then go back to Step 5.

□

Note that, in solving 2ECA-SV, we may restrict paths $P(u,v)$ in Step 3 to those having $(V(P(u,v)) - \{u,v\}) \cap V_T = \emptyset$, even though it is usual to require $E(P(u,v)) \cap E'_T = \emptyset$.

4.2 Correctness and performance ratio of *R2ES*

Theorem 4.1 $\lambda(S; G' + E'') \geq 2$.

(Proof) The theorem follows from the results of [8], [9]. □

We consider the relationship between the total weight of $\overrightarrow{E''_i}$ and that of an optimum solution $E^* \subseteq E - E'$. Let $E^*_s = \{(u,v) \in E_s - E'_s \mid (u',v') \in E^*, u' \in \beta^{-1}(u), v' \in \beta^{-1}(v)\}$. Since E^* is an optimum solution, $\lambda(S'; G'_s + E^*_s) \geq 2$, there are no multiple edges in E^*_s and $w_s(E^*_s) = w(E^*)$. Thus, we consider E^*_s instead of E^* in the rest of the section.

Let $\overline{G'_s} = (\overline{V_s}, \overline{E'_s})$ be any fixed minimal subgraph of G'_s such that $E'_T \subseteq \overline{E'_s}$ and $\lambda(S'; \overline{G'_s} + E^*_s) = 2$. Analogously to Section 3.2, we define a tent $\mathcal{T}_j$, a floor F_j (or $\overrightarrow{F_j}$), a net N_j, a root r_j and $x_j = r_j$ by replacing E^*_B, $\overline{T_B} = (\overline{V_B}, \overline{E'_B})$ and $T_P = (V_P, E'_P)$ with E^*_s, $\overline{G'_s} = (\overline{V_s}, \overline{E'_s})$ and $T = (V_T, E'_T)$, respectively. We can prove Lemma 4.1 and 4.2 and Theorem 4.2 below similarly to Section 3.2.

Lemma 4.1 *For each net N_j, a set of directed edges $\overrightarrow{\mathcal{A}_j} \subseteq \overrightarrow{E_i} - \overrightarrow{E'_i}$ satisfying the following (a) and (b) can be constructed from E^*: (a) all vertices of $\overrightarrow{F_j}$ are reachable from r_j in $\overrightarrow{F_j} + \overrightarrow{\mathcal{A}_j}$; (b) If r_j is the leaf of $\overrightarrow{F_j}$ then $w_i(\overrightarrow{\mathcal{A}_j}) \leq (2 - 2/p_j) w(E(\mathcal{T}_j))$ else $w_i(\overrightarrow{\mathcal{A}_j}) \leq 2w(E(\mathcal{T}_j))$, where p_j is the number of leaves of F_j.* □

Lemma 4.2 *For some h with $1 \leq h \leq |L|$, a set of directed edges $\overrightarrow{B_h} \subseteq \overrightarrow{E_h} - \overrightarrow{E'_h}$ satisfying the following (a) and (b) can be constructed from E^*_s: (a) $\overrightarrow{T_h} + \overrightarrow{B_h}$ is strongly connected; (b) $w_h(\overrightarrow{B_h}) \leq (2 - 2/|L|) w(E^*)$.*

□

Theorem 4.2 *If $\lambda(S; G') = 1$ then the proposed algorithm R2ES generates a $(2 - 2/|L|)$-approximate solution to 2ECA-SV in $O(|V||E| + |V|^2 \log |V| + |L||V|^2)$ time, where L is the set of leaves of the path-tree of G'.*

□

5 Concluding remarks

We have proposed approximation algorithms for 2-vertex- or 2-edge-connectivity augmentation of specified vertices and have shown that $PR = 2 - 2/|L|$ if $\kappa(S; G') = \lambda(S; G') = 1$. The time complexity is $O(|V||E| + |V|^2 \log |V| + |L||V|^2)$. Comparing our algorithms with the other approximation algorithms in [10, 13] or [11, 14] through computational experiments is left for future research.

References

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

[2] T. Hsu and V. Ramachandran: "On finding a smallest augmentation to biconnect a graph", SIAM J. Comput., pp. 889–912 (1983).

[3] K. Eswaran and R. Tarjan: "Augmentation problems", SIAM J. Comput., **5**, pp. 653–655 (1976).

[4] G. N. Fredericson and J. Ja'ja': "Approximation algorithms for several graph augmentation problems", SIAM J. Comput., **10**, pp. 270–283 (1981).

[5] S. Khuller and R. Thurimella: "Approximation algorithms for graph augmentation", Journal of Algorithms, **14**, pp. 214–225 (1993).

[6] M. Penn and H. Shasha-Krupnik: "Improved approximation algorithms for weighted 2- and 3-vertex connectivity augmentation problems", Journal of Algorithms, **22**, pp. 187–196 (1997).

[7] S. Khuller and U. Vishkin: "Biconnectivity approximations and graph carvings", Journal of the ACM, **41**, 2, pp. 214–235 (1994).

[8] M. Tamura, S. Taoka and T. Watanabe: "A 2-approximation algorithm 2-ABIS for 2-vertex-connectivity augmentation of specified vertices in a graph", IEICE Trans. Fundamentals, **E86–A**, 3 (2003). (submitted).

[9] T. Watanabe, Y. Higashi and A. Nakamura: "Construction robust networks by means of graph augmentation problems", Trans. IEICE of Japan, **73-A**, 7, pp. 1242–1254 (1990). (in Japanese) Also see Electronics and Communications in Japan, Part 3, Vol. 74, No. 2, pp. 79–96 (1991).

[10] R. Ravi and D. P. Williamson: "An approximation algorithm for minimum-cost vertex-connectivity problems", Algorithmica, **18**, pp. 21–43 (1997).

[11] P. Klein and R. Ravi: "When cycles collapse: A general approximation technique for constrained two-connectivity problems", Proceedings of the Third MPS Conference on Integer Programming and Combinatorial Optimization, pp. 39–55 (1993).

[12] R. Tarjan: "Data Structures and Network Algorithms", CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia (1983).

[13] L. Fleischer: "A 2-approximation for minimum cost {0,1,2} vertex connectivity", Lecture Notes in Computer Science, **2081**, pp. 115–129 Springer-Verlag, (2001).

[14] K. Jain: "A factor 2 approximation algorithm for the generalized steiner network problem", Combinatorica, **21**, pp. 39–60 (2001).

[15] A. Aho, J. Hopcroft and J. Ullman: "The Design and Analysis of Computer Algorithms", Addison-Wesley, Reading, MA (1974).

[16] M. Tamura, S. Taoka and T. Watanabe: "A 2-approximation algorithm 2-ABIS for 2-vertex-connectivity augmentation of specified vertices in a graph", Proc. of the 15th Karuizawa Workshop on Circuits and Systems, IEICE of Japan, pp. 447–452 (2002). (in Japanese).