λ-辺連結グラフにおける指定点集合の $(\lambda+1)$-辺連結化のための 2-近似アルゴリズム *FSA+1*

田岡 智志 †, 間島 利也 ‡, 渡邉 敏正 †
† 広島大学大学院 工学研究科
‡ 広島国際大学 社会環境科学部
Email: {taoka,watanabe}@infonets.hiroshima-u.ac.jp, mashima@it.hirokoku-u.ac.jp

[Abstract] 無向グラフ $G=(V,E)$，G の部分グラフ $G'=(V,E')$, 指定点集合 $\Gamma\subseteq V$，コスト関数 c: $E\to\mathbb{R}^+$ が与えられたとき，Γ の k-辺連結化問題 (kECA-SV) を考える．本稿では，$\lambda(V;G')=\lambda(\Gamma;G')$ なる場合の $(\lambda+1)$ECA-SV に対し，$O(\Delta+|V||E|)$ 時間の 2-近似アルゴリズム *FSA+1* を提案する，ここで，$\lambda=\lambda(\Gamma;G')$ とし，Δ は G' の構造グラフ $F(G')$ を構成する計算時間複雑度とする．

A 2-Approximation Algorithm *FSA+1* to $(\lambda+1)$-Edge-Connect a Specified Set of Vertices in a λ-Edge-Connected Graph

Satoshi Taoka†, Toshiya Mashima‡, Toshimasa Watanabe†
† Graduate School of Engineering, Hiroshima University
‡ Faculty of Infrastructural Technologies, Hiroshima International University
Email: {taoka,watanabe}@infonets.hiroshima-u.ac.jp, mashima@it.hirokoku-u.ac.jp

[Abstract] Given an undirected graph $G=(V,E)$, a subgraph $G'=(V,E')$ of G, a specified set $\Gamma\subseteq V$, and a cost function c: $E\to\mathbb{R}^+$, we consider the k-edge-connectivity augmentation problem for Γ (kECA-SV). The paper proposes an $O(\Delta+|V||E|)$ time 2-approximation algorithm *FSA+1* for $(\lambda+1)$ECA-SV for the case when $\lambda(V;G')=\lambda(\Gamma;G')$, where $\lambda=\lambda(\Gamma;G')$ and Δ is the time complexity of constructing a structural graph $F(G')$ of G'.

1 Introduction

[Problem definition] The k-edge-connectivity augmentation problem for a specified set of vertices (W-kECA-SV) is defined as follows: "Given an undirected graph $G=(V,E)$, a spanning subgraph $G'=(V,E')$ of G, a specified set of vertices $\Gamma\subseteq V$ and a cost function c: $E\to\mathbb{R}^+$ (nonnegative real numbers), find a set $E''\subseteq E-E'$ of edges, each connecting distinct vertices of V, of minimum total cost such that $\lambda(\Gamma;G'+E'')\geq k$ for $G'+E''=(V,E'\cup E'')$," where $\lambda(\Gamma;G'')\geq k$ means that G'' has at least k edge disjoint paths between any pair of vertices in Γ. Costs $c((u,v))$ for $(u,v)\in E$ is denoted as $c(u,v)$ for simplicity. $\lambda(\Gamma;G)$ with $\Gamma=V$ is denoted as $\lambda(G)$. The k-vertex-connectivity augmentation problem for a specified set of vertices (W-kVCA-SV) is similarly defined if we consider "internally disjoint paths" instead of "edge disjoint paths". For simplicity, let kECA-SV denote W-kECA-SV unless otherwise stated in this paper. kECA-SV with $\Gamma=V$ is denoted simply as kECA, called the k-edge-connectivity augmentation problem. The problem is called the unweighted version, denoted by UW-kECA-SV, if G has k multiple edges connecting u and v for any $u,v\in V$ and any edge cost is unity.

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

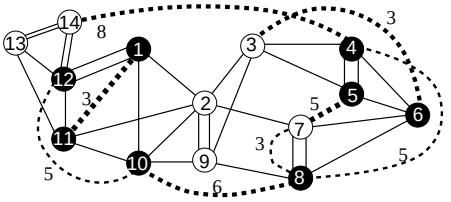

Figure 1: $G=(V,E)$ and a spanning subgraph $G'=(V,E')$ with $\lambda(V;G')=\lambda(\Gamma;G')=4$, where black vertices are in Γ, solid lines are edges in E', (fine or bold) dotted lines are ones in $E-E'$, numbers shown beside edges are costs, and bold dotted lines denote an optimum solution E^*, of total cost 25, for 5ECA-SV.

Table 1: Summary of NP-completeness results, where K_n is a complete graph of n vertices

Problem	Ref.	G	G'	Costs	Comments
2ECA	[4]	K_n	tree	1 or 2	
3ECA	[27]	K_n	$\lambda(G')\geq 2$	1 or 2	
kECA	[26]	K_n	no edges		$k\geq 2$

Example 1 *Fig. 1 shows an example of* 5*ECA-SV, where* $\lambda(V;G')=4(=\lambda)$. *Also shown is an optimum solution* E^*.

[Applications and related results] The problem has application to designing robust networks: a k-edge-connected network can survive $k-1$ communication lines' failure.

Some related results on kECA and kECA-SV are summarized in Tables 1 and 2.

Table 2: Results concerning optimum, approximate or heuristic solutions, where $n = |V|$ and $m = |E|$

Problem	Ref.	PR	Time	Space	Comments
2ECA	[4]	2 if $\lambda(G') = 1$ 3 if $\lambda(G') = 0$	$\Theta(n^2)$	$\Theta(n^2)$	
2ECA	[14]	2 if $\lambda(G') = 1$	$O(m + n \log n)$	$O(n + m)$	
3ECA	FS[27]	2	$O(n^3)$	$\Theta(n^2)$	minimum-cost arborescence
kECA	[15]	2	$O(kn(m + \log n) \log n)$	$O(n + m)$	packing arborescence
3ECA	[28]	unbounded	$O(n^3 \log n)$	$\Theta(n^2)$	max weight matching*
kECA	[24, 25]	unbounded	$O(n^3 \log n)$	$\Theta(n^2)$	max weight matching*
2ECA-SV	[23]	2	$\Theta(n^2)$	$\Theta(n^2)$	$\lambda(G') \geq 1$, reduction to 2ECA[4]
3ECA-SV	[29]	unbounded	$O(n^3 \log n)$	$\Theta(n^2)$	$\lambda(G') = 2$, max weight matching*
LECA**	[12]	2	$O(\|V\|^{10}\|E\|^7)$		
LECA**	[8]	$2k - 1$ if $k \geq 2$ 2 if $k = 1$	$O(k^2 n^2 + kn\sqrt{m \log \log n})$		
LECA**	[10]	$2H(k)$	$O(n \min\{kn, m\}(k + n) + n^2 \rho)$		$H(k) = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$

* Four heuristic algorithms are proposed, and the one using maximum weight matching has highest capability.
** LECA (local edge-connectivity augmentation problem) contains kECA-SV as a subproblem.

[Subjects and main results] In this paper, $\lambda(\Gamma; G')$ is denoted as λ for short. For $(\lambda + 1)$ECA-SV with $\lambda(G') = \lambda$, (1) and (2) are the main results, where we assume $\lambda \geq 1$ since if $\lambda = 0$ then the problem is optimally solved by using an algorithm for finding a minimum-cost spanning tree. (1) The recognition version of $(\lambda + \delta)$ECA is NP-complete even if $\delta = 1$, G' is λ-edge-connected and edge costs are either 1 or 2. (This part is omitted because of space limitation in this paper.) (2) A 2-approximation algorithm *FSA+1* for $(\lambda + 1)$ECA-SV with $\lambda(G') = \lambda(\Gamma; G')$ is proposed, based on a minimum-cost arborescence algorithm. Its time complexity (space complexity, respectively) is $O(\Delta + |V||E|)$ ($O(|V|^2 + |E|)$) if λ is even, or $O(\Delta + |E|)$ ($O(|V| + |E|)$) if λ is odd, where Δ is time complexity of finding a structural graph of given graph G'.

(The early version of this paper appeared in [16, 24].)

2 Preliminaries

Technical terms not specified here can be identified in [2, 21].

An (*undirected*) *graph* $G = (V(G), E(G))$ is often denoted as $G = (V, E)$, and an (undirected) edge between u and v is denotes as (u, v). A directed graph $\overline{G} = (V(\overline{G}), A(\overline{G}))$ is often denoted as $\overline{G} = (V, A)$, and a directed edge from u to v is denoted as $\langle u, v \rangle$. The *degree* of a vertex v is denoted as $d_G(v)$, or simply $d(v)$: v is often called a degree-$d(v)$ vertex. A path from u to v in G is referred to as a (u, v)-path. We consider a pair of multiple edges as a cycle of length 2. A *connected component* (is simply called a *component*). Let $\lambda(u, v; G)$ or simply $\lambda(u, v)$ denote the maximum number of edge-disjoint (u, v)-paths of G. We represent as $\lambda(S; G) = \min\{\lambda(u, v; G) \mid u, v \in S\}$ for a set of vertices $S \subseteq V$. If $S = V$ then we simply represent as $\lambda(G)$, which is called the *edge-connectivity* of G.

Let $S \subseteq V \cup E$ be any minimal set such that $G - S$ (a graph obtained by deleting all element of S from G) is disconnected where deleting a vertex v removes all edges incident to v. S is called a *separator* of G, or in particular a (u, v)-separator if u and v are disconnected in $G - S$. A *minimum separator* S of G is a separator of minimum cardinality among those of G, and $|S|$ is the edge-connectivity (denoted by $\lambda(G)$) of G in case $S \subseteq E$; particularly such $S \subseteq E$ is called a *minimum cut* (of G). (For a minimum separator S, $|S|$ is the vertex-connectivity $\kappa(G)$ if $S \subseteq V$.) It is called also a *k-cut* if $|S| = k$. If $|S| = 1$ then the element of S is called a *cutpoint* in case $S \subseteq V$ or a *bridge* in case $S \subseteq E$. A minimum cut $S \subseteq E$ is often denoted as $(X, Y; G)$, where $X \cup Y$ is a partition of V such that $S = \{(u, v) \in E \mid u \in X, v \in Y\}$. If no confusion occurs then $(X, Y; G)$ is simply represented as (X, Y). We say that a cut $(X, \overline{X})$ *separates* Γ if and only if $X \cap \Gamma \neq \emptyset$ and $\overline{X} \cap \Gamma \neq \emptyset$: $S = (X, \overline{X})$ is called a Γ-cut or a k-Γ-cut if $|S| = k$. A minimum Γ-cut is defined similarly to a minimum cut.

A directed path from u to v is referred as a $\langle v_1, v_n \rangle$-path, and we say that v is reachable from u. An *arborescence* is a directed acyclic graph with one specified vertex, called the *root*, having no entering edges, all other vertices having exactly one entering edge and they are reachable from the root. A *minimum-cost arborescence* is an arborescence of minimum total cost. For an arborescence with the root r, a graph consisting of edges $\langle u, v \rangle$ such that the edges $\langle v, u \rangle$ are contained in the arborescence is called a *reverse arborescence* rooted at r. If there is a $\langle u, v \rangle$-path in a (reverse) arborescence, u is an *ancestor* of v and v is a *descendant* of u. For a reverse arborescence $\overline{G'_d}$, u is called a leaf if there are no ancestors of u in $\overline{G'_d}$. Suppose that u is not an ancestor of v and v is not an ancestor of u in a reverse arborescence with the root r. Then a *nearest common descendant* of u and v is a common descendant t of u and v such that it is the nearest among all such common descendants. A *cactus* is an undirected connected graph in which any pair of cycles share at most one vertex: each shared vertex is a cutpoint. An edge of a cactus is called a *cycle edge* if it is contained in a cycle; otherwise it is called

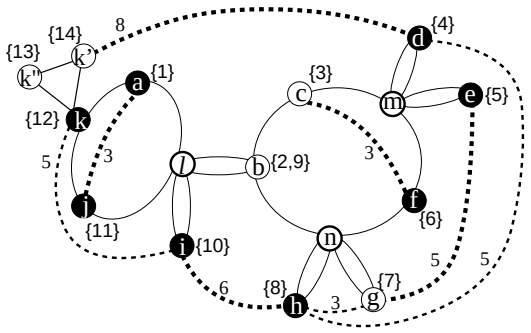

Figure 2: A graph G_s (to be constructed in Step 2 of *FSA+1*) and a structural graph G'_s $(= F(G'))$ of G', where black vertices are in $\Gamma_s = \rho(\Gamma)$, edges in G'_s are denoted by solid lines with weight 2, and (fine or bold) dotted lines are edges in $E_s - E'_s$ (bold dotted lines are edges in E^*_s constructed from an optimum solution E^*), and $\rho^{-1}(u)$ for each $u \in V_s$ is shown in braces

a *tree edge*. A *leaf* of a cactus is either a vertex v with $d(v) = 1$ or v' with $d(v') = 2$ included in a cycle.

A *structural graph* $F(G)$ of a given graph $G = (V, E)$ with edge-connectivity $\lambda(G) = \lambda$ is a representation of all minimum cuts of G (Fig. 2). $F(G)$ is an edge-weighted cactus of $O(|V|)$ nodes and edges such that each tree edge has weight λ and each cycle edge has weight $\lambda/2$. Particularly if λ is odd then $F(G)$ is a weighted tree. There is a one-to-one function $\rho : V(G) \to V(F(G))$, and any vertex of $V(F(G)) - V(G)$ is called an *empty node* to which $V(G)$ has no corresponding vertices (see l, m, n in Fig. 2). It is shown that $F(G)$ can be constructed in $O(|V||E|)$ time [13] or $O(|E|+\lambda^2|V|\log(|V|/\lambda))$ time [5]. Note that if λ is even then replacing each tree edge by a pair of multiple edges of weight $\lambda/2$ preserves the properties of structural graphs and makes their handling easy because the resulting graphs have no bridges. This graph, as well as a tree in the case where λ is odd, is called a *modified cactus*. In this paper, $F(G)$ denotes a modified cactus unless otherwise stated. Note that $\lambda(F(G)) = 1$ if λ is odd and $\lambda(F(G)) = 2$ if λ is even.

3 A 2-Approximate Algorithm for W-kECA-SV with $\lambda(G') = \lambda(\Gamma; G')$

In this section, we propose a 2-approximation algorithm *FSA+1* for $(\lambda+1)$ECA-SV with $\lambda(G') = \lambda(\Gamma; G') = \lambda$.

$(\lambda+1)$ECA-SV for G' with $\lambda(G') = \lambda(\Gamma; G')$ can be reduced to 2ECA-SV or 3ECA-SV for $F(G')$. Since $\lambda(F(G')) = 1$ if $\lambda(\Gamma; G')$ is odd and $\lambda(F(G')) = 2$ if $\lambda(\Gamma; G')$ is even, we consider 2ECA-SV or 3ECA-SV for $F(G')$ by setting $\theta \leftarrow 2$ if $\lambda(\Gamma; G')$ is odd, and $\theta \leftarrow 3$ if $\lambda(\Gamma; G')$ is even.

For W-2ECA-SV, [23] proposed an $\Theta(|V|^2)$ time 2-approximation algorithm for the case with $\lambda(G') = 1$. [23] proposed four heuristic algorithms for W-3ECA-SV and UW-3ECA-SV for the case with $\lambda(G') = 2$. These four algorithms are outlined as follows. First, a given graph G' with $\lambda(G') = \theta - 1$ is transformed into a graph G'_s such that a minimum solution to θECA-SV for G', G and c is a minimum solution to θECA for G'_s, G_s and c_s, and vice versa. For such a graph G'_s W-2ECA (UW-3ECA or W-3ECA, respectively) is solved by means of an $\Theta(|V|^2)$ time algorithm of [4] (an $O(|V|+|E|)$ time algorithm of [30] or some heuristic algorithms proposed in [24, 25, 28]). Then a 2-approximate solution (an optimum one or a heuristic one) is obtained. The algorithm of [4] is based on a minimum-cost arborescence algorithm [7].

Our algorithm *FSA+1* is based on a minimum-cost arborescence algorithm [7] and utilizes the algorithm of [14], instead of [4], for solving 2ECA. The algorithm *FSA+1* is outlined as follows. First, we construct a structural graph $G'_s = (V_s, E'_s)$ $(= F(G'))$ of $G' = (V, E')$ by the algorithm in [5, 13] (see G and G' of Fig. 1 and G_s of Fig. 2). According to the construction, we obtain a graph $G_s = (V_s, E_s)$, a specified vertex set $\Gamma_s = \rho(\Gamma)$, a cost function $c_s : E_s - E'_s \to \mathbf{R}^+$ and a backpointer $b : E_s - E'_s \to E$ (see Fig. 2). Secondly, G'_s is changed into a simple graph by deleting multiplicity, and then a tree G'_b will be constructed as follows (see Fig. 3): for each cycle C that is remaining in this simple graph, a new vertex v_C is added, each vertex on C and v_C are connected by an edge, and then all edges of C are deleted. Let G'_b denote the resulting tree. Next, we choose some vertex $r \in \Gamma_s$ as the root of G'_b, and direct every edge of G'_b toward r. Note that r becomes a leaf in $\overline{G'_d}$ to be constructed later. Let $\overline{G'_b} = (V_b, A'_b)$ denote the resulting directed graph (see Fig. 6). Some modification of G_s and c_s will be done. From G_s, $\overline{G'_b}$ and c_s, we obtain $\overline{G_b} = (V_b, A_b)$ and $\overline{c_b} : A_b \to \mathbb{R}^+$ that are a supergraph containing $\overline{G'_b}$ and the associated cost function, respectively. And then, we construct a maximal subgraph $\overline{G'_d} = (V_d, A'_d)$, which is a tree, from $\overline{G'_b} = (V_b, A'_b)$ such that $\Gamma_s \subseteq V_d \subseteq V_b$, $A'_d \subseteq A'_b$, and $u \in \Gamma_s$ for any leaf u of $\overline{G'_d}$. According to the construction, we obtain a supergraph containing $\overline{G'_d}$, $\overline{G_d}$, the associated cost function $\overline{c_d} : A_d \to \mathbb{R}^+$ and a backpointer b_d from $\overline{G_b}$ and $\overline{c_b}$ (see Fig. 7). Finally we find a minimum-cost arborescence $T_m = (V_b, A''_b)$ of $\overline{G_d}$ with respect to $\overline{c_d}$ (see Fig. 8), and an approximate solution of G' is obtained from $A''_b - A'_d$.

[Description of *FSA+1*] We first present Procedure *REMAKE* that changes a modified cactus G'_s $(= F(G'))$ into a spanning tree G'_b.

Procedure *REMAKE*;
/* Input : $G'_s = (V_s, E'_s)$. Output : $G'_b = (V_b, E'_b)$. */

1. If λ is odd then $G'_b \leftarrow G'_s$ and stop.
2. Delete multiplicity of edges in G'_s, making it simple. Then find all cycles (each being a 2-ecc of G'_s) by a depth-first-search.
3. For each cycle C, add a dummy vertex w_C, connect w_C to every vertex on C and delete all edges (u, v)

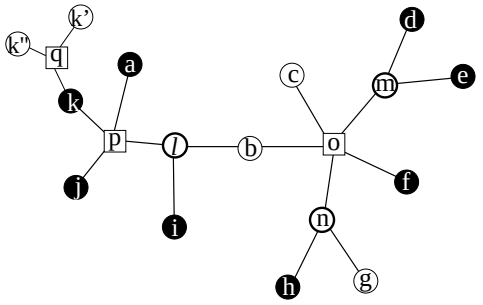

Figure 3: A tree G'_b constructed from G'_s in Fig. 2 by procedure *REMAKE*. A square denotes a dummy vertex.

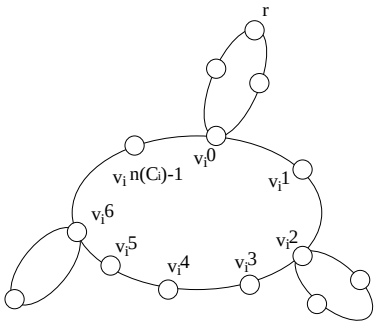

Figure 4: A circuit $C_i = \{(v_i^0, v_i^1), (v_i^1, v_i^2), ..., (v_i^{n(Ci)-1}, v_i^0)\}$ in G'_s.

of C. Let $G'_b = (V_b, E'_b)$ be the resulting graph and W be the set of dummy vertices in G'_b (Fig. 3). □

FSA+1 chooses a vertex $r \in \Gamma_s$ called the root of G'_d. Suppose G'_s has at least one cycles whose length is more than 2. Let C_i, $1 \leq i \leq h$, be cycles in G'_s. For each cycle C_i, let $v_i^0 \in V(C_i)$ denote the vertex which is nearest from the root r, and v_i^0 is called the *starting vertex* of C_i. Note that v_i^0 is the first visited vertex in C_i when we execute the depth first search starting from r in G'_s. We number the vertices of C_i as $v_i^0, v_i^1, v_i^2, \ldots, v_i^{n(C_i)-1}$ clockwise, where $n(C_i)$ denotes the length of C_i (see Fig. 4). Let $L_i(j,k)$ denote the set of vertices v_i^j through v_i^k clockwise on C_i for $j \leq k$. For any vertex $u \in V_s$, let us denote $A_i(j,k;u) = \{\langle v_i^l, u\rangle \mid v_i^l \in L_i(j,k)\}$ (see Fig. 5(3)), where all self-loops are deleted. Note that if $0 < j \leq k < n(C_i)$ then $v_i^0 \notin L_i(j,k)$. These notations are used in constructing $\overline{G_b}$ and $\overline{G_d}$.

Algorithm *FSA+1*;

/* Input: A graph $G = (V, E)$, a subgraph $G' = (V, E')$ of G, a specified set $\Gamma \subseteq V$ with $\lambda(\Gamma; G') = \lambda(V; G') = \lambda$, and a cost function $c : E \to \mathbb{R}^+$. */

/* Output: An edge set $E'' \subseteq E - E'$ such that $\lambda(\Gamma; G' + E'') \geq \lambda + 1$. */

1. Construct a structural graph $G'_s = (V_s, E'_s)$ (= $F(G')$) of G' by the algorithm in [5, 13].
2. Construct $G_s = (V_s, E_s)$, c_s and Γ_s as follows. First, $E_s \leftarrow E'_s$. Then, for each edge $(x, y) \in E - E'$, set $E_s \leftarrow E_s \cup \{(u, v)\}$, $c_s(u, v) \leftarrow c(x, y)$ and $b(u, v) \leftarrow (x, y)$, where $u = \rho(x)$ and $v = \rho(y)$ (see Fig. 2). Let $\Gamma_s = \rho(\Gamma) = \{\rho(v) \mid v \in \Gamma\}$.
3. Construct a tree $G'_b = (V_b, E'_b)$ from G'_s by using *REMAKE* (see Fig. 3). /* The set W of dummy vertices in G'_b is obtained. */
4. If G'_b has any leaf $v \in \Gamma_s$ then choose v as the root r. Otherwise, choose any leaf $v \in V_b$, find any vertex $w \in \Gamma_s$ that is nearest from v in G'_b, and let w be the root r.
5. Construct a directed tree $\overline{G'_b} = (V_b, A'_b)$ from G'_b by directing each edge of E'_b toward r.
6. Define a directed graph $\overline{G_b} = (V_b, A_b)$, a cost function $\overline{c_b} : A_b \to \mathbb{R}^+$ and backpointers $\overline{b_b} : A_b \to E_s$ as follows (see Figs. 5 and 6):

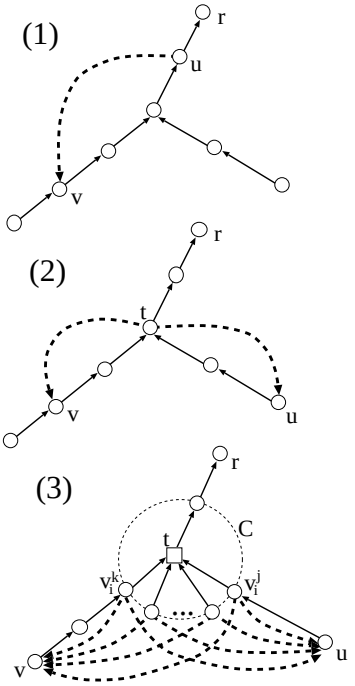

Figure 5: Schematic explanation of a set of edges $A^c(e)$ of *FSA+1*: (1) Case 6.2(a), (2) Case 6.2(b1), (3) Case 6.2(b2).

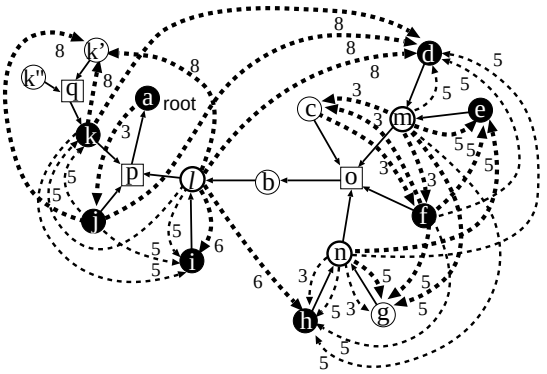

Figure 6: A graph $\overline{G_b} = (V_b, A_b)$ and a directed tree $\overline{G'_b} = (V_b, A'_b)$, where solid arrows are edges in A'_b with $\overline{c_b}(e) = 0$ for any $e \in A'_b$, and (fine or bold) dotted lines are edges in $A_b - A'_b$ (bold dotted lines corresponds to edges in E^*_s).

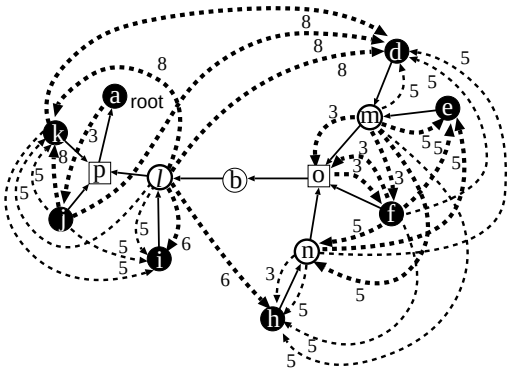

Figure 7: A graph $\overline{G_d} = (V_d, A_d)$ and a directed tree $\overline{G'_d} = (V_d, A'_d)$, where all self-loops are omitted, solid arrows are edges in A'_d ($\overline{c_d}(e) = 0$ for any $e \in A'_d$) and dotted lines are edges in $A_d - A'_d$ (bold dotted lines are corresponding to edges in E^*_s).

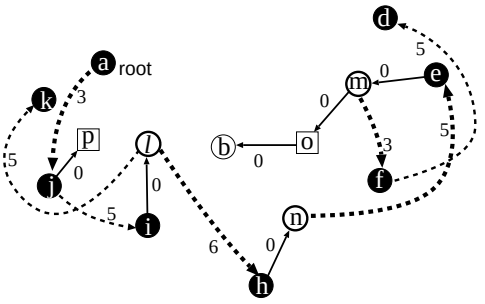

Figure 8: A minimum-cost arborescence $T_m = (V_d, A_m)$ of total cost 32 given by *FSA+1*. $E''_s = E_s - (E'_s \cup \{(d,k'),(g,h)\}$ (see Fig. 2) is obtained from T_m at Step 9 of *FSA+1* and an approximate solution $E'' = E - (E' \cup \{(4,14),(7,8)\})$ of Fig. 1 is obtained from E''_s at Step 10 of *FSA+1*, where $c_s(E''_s) = c(E'') = 27$.

6.1. $A_b \leftarrow A'_b$. For each $e \in A'_b$, $\overline{c_b}(e) \leftarrow 0$ and $\overline{b_b}(e) \leftarrow \emptyset$.

6.2. For each edge $e = (u,v) \in E_s - E'_s$, execute the following (1) and (2).

(1) construct a set $A^c(e)$ of directed edges by executing the following (a) or (b) (see Fig. 5).

(a) If one of $\{u,v\}$, say u, is an descendant of the other, v, in $\overline{G'_b}$, then $A^c(e) \leftarrow \{\langle u,v\rangle\}$ (Fig. 5(1)).

(b) Otherwise (u is not an descendant of v and v is not an descendant of u in $\overline{G'_b}$), find a nearest common descendant t of u and v, and execute either (b1) or (b2). (Note that (b2) is executed only if λ is even.)

(b1) If $t \notin W$ then $A^c(e) \leftarrow \{\langle t,u\rangle, \langle t,v\rangle\}$ (Fig. 5(2)).

(b2) If $t = w_{C_i} \in W$ (that is, t is a dummy vertex) then $A^c(e) \leftarrow A_i(j,k;u) \cup A_i(j,k;v)$, where we assume that the $\langle u,t\rangle$-path and the $\langle v,t\rangle$-path of $\overline{G'_b}$ pass through v^j_i and v^k_i with $0 < j < k < n(C_i)$, respectively (Fig. 5(3)).

(2) $\overline{c_b}(e') \leftarrow c_s(e)$ and $\overline{b_b}(e') \leftarrow e$ for each $e' \in A^c(e)$, and then $A_b \leftarrow A_b \cup A^c(e)$ (see Fig. 6).

7. (1) Construct a maximal subgraph $\overline{G'_d} = (V_d, A'_d)$ consisting of those edges on the $\langle u,r\rangle$-path of $\overline{G'_b}$ for any $u \in \Gamma_s$ (see Fig. 7). (Note that $\overline{G'_d}$ is a reverse arborescence rooted at r.)
(2) Let $I = V_b - V_d$. Let X_i ($1 \le i \le n$) denote the vertex set of any maximal tree in $\overline{G'_b} - A'_d$ such that $|X_i| \ge 2$ and $X_i \cap V_d = \{x_i\}$. ($I = \{c,g,k',k'',q\}, X_1 = \{k,k',k'',q\}$ with $x_1 = k$, $X_2 = \{c,o\}$ with $x_2 = o$, $X_3 = \{q,n\}$ with $x_3 = n$ in Fig. 6.) Define $\overline{G_d} = (V_d, A_d)$, $c_d : E_d \to \mathbb{R}^+$ and $\overline{b_d} : E_d \to E_b$ as follows.
Set $A_d \leftarrow A'_d$. For each $\langle u,v\rangle \in A_b - A'_b$, (i) if $\{u,v\} \cap I \neq \emptyset$ and $u' \neq v'$ then $A_d \leftarrow A_d \cup \{\langle u',v'\rangle\}$, $c_d(u',v') \leftarrow c_b(u,v)$, $\overline{b_d}(u',v') \leftarrow \langle u,v\rangle$, where ($u'$ or v' denotes x_i or x_j) if u or v is in some X_i or X_j, respectively, or ($u' = u$ and $v' = v$) otherwise; (ii) otherwise $A_d \leftarrow A_d \cup \{\langle u,v\rangle\}$, $c_d(u,v) \leftarrow c_b(u,v)$, $\overline{b_d}(u,v) \leftarrow \langle u,v\rangle$.

8. Find a minimum-cost arborescence $T_m = (V_d, A_m)$ rooted at r in $\overline{G_d}$ with respect to $\overline{c_d}$ (see Fig. 8).

9. Construct a solution $E''_s = \{\overline{b_b}(\overline{b_d}(e)) \in E_s - E'_s \mid e \in A_m, \overline{b_b}(\overline{b_d}(e)) \neq \emptyset\}$ (with multiplicity deleted) such that $\lambda(\Gamma_s; G'_s + E''_s) = \theta$.

10. Construct a solution $E'' = \{b(u,v) \in E - E' \mid (u,v) \in E''_s\}$ (with multiplicity deleted) such that $\lambda(\Gamma; G' + E'') = \lambda + 1$. □

Remark 3.1 *When we change a graph having multiple edges into a simple one, we choose an edge $e = (u,v)$ of minimum cost among those connecting u and v for any pair of vertices u, v.* □

Remark 3.2 *If the problem is 2ECA-SV with $\lambda(G) = 1$ and $\Gamma = V$ then* FSA+1 *is similar to the 2-approximation algorithm of [14].* □

[Correctness of *FSA+1*] We show series of necessary results, some of which are stated without proofs because of space limitation.

For a set E'' or E''_s of edges, each connecting distinct vertices of V or of V_s, respectively, let

$$c(E'') = \sum_{(u,v)\in E''} c(u,v) \quad \text{or}$$

$$c_s(E''_s) = \sum_{(u,v)\in E''_s} c_s(u,v).$$

The following lemma shows that it suffices to consider $(\lambda+1)$ECA-SV for a cactus $G'_s = (V_s, E'_s)$ (instead of G') from the properties of a structural graph of G'.

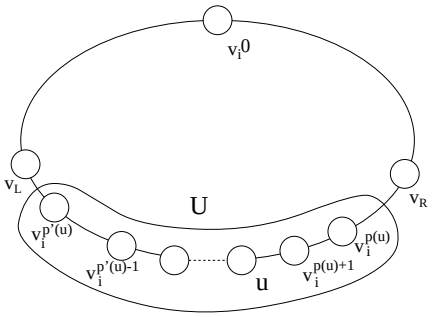

Figure 9: A set U of vertices in G'_s.

Lemma 3.1 *If $\lambda(\Gamma; G' + E'') \geq \lambda + 1$ for some E'' then there is E''_s with $c_s(E''_s) \leq c(E'')$ such that E''_s is a solution to G'_s and Γ_s. For a set E''_s, let $b(E''_s) = \{b(u,v) \mid (u,v) \in E''_s\}$ with multiplicity deleted. If E''_s is a solution to G'_s and Γ_s then $\lambda(\Gamma; G' + b(E''_s)) \geq \lambda + 1$ and $c_s(E''_s) \geq c(b(E''_s))$.* □

Clearly, $\overline{G'_d}$ satisfies the following Properties 3.1, 3.2 and 3.3 from its construction.

Property 3.1 *If u is an descendant of v in $\overline{G'_d}$, u is reachable from v in $\overline{G'_d}$.* □

Property 3.2 *Any pair of dummy vertices in $\overline{G'_d}$ are not adjacent in $\overline{G'_d}$.* □

Property 3.3 *If $(X, Y; G'_s)$ is any minimum Γ-cut which is either a bridge or a 2-cut such that $\{(x,y),(x,z)\} = (X, Y; G'_s)$ then there is a Γ_s-bridge $(x', y') = (X', Y'; G'_d)$ with $X' - W = X - I$ and $Y' - W = Y - I$. Conversely if $(X', Y'; G'_d)$ is any Γ_s-bridge then there is a minimum Γ-cut $(X, Y; G'_s)$ such that $X' - W = X - I$ and $Y' - W = Y - I$, and $(X, Y; G'_s)$ is either a bridge or a 2-cut such that $\{(x,y),(x,z)\} = (X, Y; G'_s)$.* □

Backpointers $\overline{b_b}$ and $\overline{b_d}$, respectively, have the following Properties 3.4 and 3.5 from its construction.

Property 3.4 *If there is any directed edge $\langle x, y\rangle \in A_b - A'_b$ in $\overline{G_b}$ and $(x', y') = \overline{b_b}(\langle x, y\rangle) \in E_s$ then $y \in \{x', y'\}$.* □

Property 3.5 *If there is any directed edge $\langle x, y\rangle \in A_d - A'_d$ in $\overline{G_d}$ and $\langle x', y'\rangle = \overline{b_d}(\langle x, y\rangle) \in E_b$ then x (y, respectively) is a descendant of x' (y') in $\overline{G'_b}$.* □

For $e \in E_s - E'_s$, let $A^c_d(e) = \{e' \in A_d \mid \overline{b_d}(e')$ is included in $A^c(e)\}$. Let $A^c(Z) = \bigcup_{e \in Z} A^c(e)$ and $A^c_d(Z) = \bigcup_{e \in Z} A^c_d(e)$. Lemma 3.2 (Lemma 3.3, respectively) shows strongly connectedness of Γ_s in $\overline{G'_b} + A^c(Z)$ (V_d in $\overline{G'_d} + A^c_d(Z)$), where $Z \subseteq E_s$ is any set of edges such that $\lambda(\Gamma_s; G'_s + Z) \geq \theta$. As an example of such Z, see E^*_s (bold dotted lines) in Fig. 2 and the edges (bold dotted lines in Fig. 7) corresponding to E^*_s.

Lemma 3.2 *If $\lambda(\Gamma_s; G'_s + Z) \geq \theta$ for a set $Z \subseteq E_s$ then Γ_s is strongly connected in $\overline{G'_b} + A^c(Z)$.*

Proof : Suppose that $\Gamma_s \subseteq V_b$ is not strongly connected in $\overline{G'_b} + A^c(Z)$, while $\lambda(\Gamma_s; G'_s + Z) \geq \theta$ holds. Clearly, r is reachable from any vertex $v \in V_b$ through edges in $\overline{G'_b}$. Let T_v denote the subtree induced by the set of all ancestors (including v itself) of v in the reverse arborescence $\overline{G'_b}$. Suppose $u \in V_s$ is the nearest vertex from r in $\overline{G'_b}$ such that $V(T_u) \cap \Gamma_s \neq \emptyset$ and u is not reachable from r in $\overline{G'_b} + A^c(Z)$. Let u_p denote the vertex with $\langle u, u_p\rangle \in A'_b$. Note that any inner vertex of $\langle u_p, r\rangle$-path in $\overline{G'_b}$ is reachable from r. We select the vertex set $U \subseteq V_b$ containing u as in the following (a1) or (a2).

(a1) If $u_p \notin W$ then $U = \{u\}$.

(a2) If $u_p = w_{C_i} \in W$ then $U = L_i(p(u), p'(u))$ with $0 < p(u) \leq p'(u) < n(C_i)$ as shown in Fig. 9, where U is a maximal vertex set such that $u \in U$, $v^0_i \notin U$ and any vertex in U is not reachable from r. From the maximality of U, $v_R = v_i^{p(u)-1}$ and $v_L = v_i^{p'(u)+1}$ are reachable from r in $\overline{G'_b} + A^c(Z)$, where superindices of vertices in U are mod $n(C_i)$.

Let $D(U)$ denote the union of $V(T_a)$ over all $a \in U$. Clearly $u \in D(U)$, $u_p \notin D(U)$, and any vertex in $D(U)$ is not reachable from r in $\overline{G'_b} + A^c(Z)$. There is a $(\theta-1)$-Γ_s-cut in G'_s separating $D(U) - W$ from $V_s - (D(U) \cup W)$. Let K be such any $(\theta-1)$-cut. If $u_p \notin W$ then K is a bridge or a 2-cut consisting of multiple edges; if $u_p \in W$ then K is a 2-cut consisting of two edges in C_i. Because K is not a $(\theta-1)$-cut separating Γ_s in $G'_s + Z$, there is an edge $e = (s, v) \in Z$ such that $s \notin D(U) - W$ and $v \in D(U) - W$. Suppose that $v \in V(T_x)$ for some vertex $x \in U$. Let t be the nearest common descendant of v and s in $\overline{G'_b}$, where t may be equal to s. Then $t \notin D(U)$ and t is reachable from r in $\overline{G'_b} + A^c(Z)$. We select a vertex $z \in V_s$ as in the following (b1) or (b2).

(b1) The case with $t \notin W$. Then $z \leftarrow t$ (Fig. 10).

(b2) The case with $t \in W$. If $t \neq u_p$ then $z \leftarrow u'$ (Fig. 11), where u' is the parent of t on the path from u to t in $\overline{G'_b}$. If $t = u_p \in W$ then we set z to either v_R or v_L by the following rule (Fig. 12). Let $t = w_{C_i}$ and $v^j_i \in V(C_i)$ be the vertex in the $\langle s, t\rangle$-path in $\overline{G'_b}$. Then set $z \leftarrow v_R$ if $0 < j < p(u)$, or $z \leftarrow v_L$ if $p'(u) < j < n(C_i)$.

For such a vertex z, we have $z \notin W$ and $z \notin D(U)$, and z is reachable from r. From the construction of $A^c(e)$, there is an edge $\langle z, v\rangle \in A^c(e) \subseteq A^c(Z)$. This means that $v \in D(U)$ is reachable from r by using $\langle z, v\rangle$, a contradiction. □

Lemma 3.3 *If $\lambda(\Gamma_s; G'_s + Z) \geq \theta$ for a set $Z \subseteq E_s$ then $\overline{G'_d} + A^c_d(Z)$ is strongly connected.*

Proof : Let $u \in V_d$ be any leaf in $\overline{G'_d}$. Clearly $u, r \in \Gamma_s$ holds. Then $\overline{G'_d} + A^c_d(Z)$ is strongly connected because, from Lemma 3.2, Γ_s is strongly connected in $\overline{G'_b} + A^c(Z)$, that is, there is a directed cycle containing both of u and r. □

Lemma 3.4 *$\lambda(\Gamma_s; G'_s + E''_s) \geq \theta$ for E''_s in Step 9 of* FSA+1.

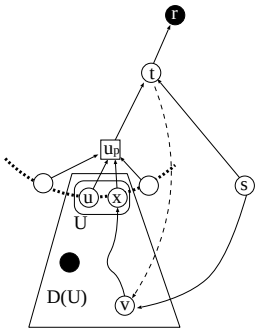

Figure 10: The case with $t \notin W$ in the proof of Lemma 3.2

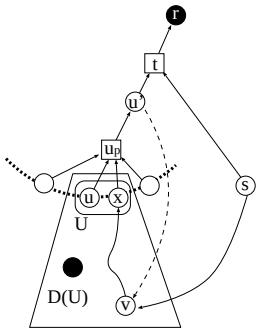

Figure 11: The case with $t \in W$ and $t \neq u_p$ in the proof of Lemma 3.2

Proof : Suppose a $(\theta-1)$-Γ_s-cut K exists in $G'_s + E''_s$. Then K is a $(\theta-1)$-Γ_s-cut of G'_s and is a bridge or a 2-cut. Let $S \subseteq V_s$ be defined from K by the following (i) or (ii).

(i) If K is a bridge e_1 in G'_s then $S \leftarrow \{u\}$, where $e_1 = (u, u_d) \in E'_b$ and $\langle u, u_d\rangle \in A'_b$.

(ii) If K is a 2-cut $\{e_1, e_2\}$ included in the same cycle in C_i of G'_s then $S \leftarrow L_i(j,k)$, where we assume that $e_1 = (v_i^{j-1}, v_i^j)$, $e_2 = (v_i^k, v_i^{k+1})$ with $0 < j \leq k < n(C_i)$, and their superindices are mod $n(C_i)$.

Let $D(S) \subseteq V_b$ be the union of $V(T_a)$ over all vertices $a \in S$ in G'_b. Clearly $\lambda(\Gamma_s; G_s) \geq \theta$ holds. From Lemmas 3.2 and 3.3, $\overline{G_d}$ $(= \overline{G'_d} + A^c_d(E_s - E'_s))$ is strongly connected. Therefore we can find a minimum-cost arborescence $T_m = (V_d, A_m)$ in Step 8 of *FSA+1*. T_m has an edge $\langle w, v\rangle \in A_m$ such that $w \notin D(S) - I$ and $v \in D(S) - I$. Let $\langle w', v'\rangle = \overline{b_d}(\langle w, v\rangle)$ and $(w'', v'') = \overline{b_b}(\langle w', v'\rangle)$. From Properties 3.4 and 3.5, v (w, respectively) is a descendant of v' (w') in $\overline{G'_b}$, and $v' = v'' \in V_s$. Then, by the construction of $A^c((w'', v'))$, we have $w'' \notin D(S)$ and $v' \in D(S)$. Because $(w'', v') \in E''_s$, this contradicts that K is a $(\theta-1)$-cut separating Γ_s in $G'_s + E''_s$. □

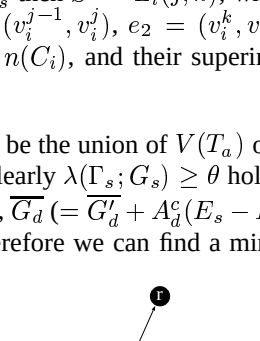

Figure 12: The case with $t \in W$ and $t = u_p$ in the proof of Lemma 3.2

The next lemma shows the performance ratio of *FSA+1*.

Lemma 3.5 *Let E'' be an approximate solution for $(\lambda+1)$ECA-SV by* FSA+1 *and E^* be an optimum solution for $(\lambda+1)$ECA-SV. Then $c(E'') \leq 2c(E^*)$.*

Proof : Let E^*_s denote the set of edges of G'_s corresponding to E^* of G'. By Lemma 3.1, E^*_s is an optimum solution such that $\lambda(\Gamma_s; G'_s + E^*_s) \geq \theta$. We will show that $c_s(E''_s) \leq 2c_s(E^*_s)$. From Lemma 3.3, $\overline{G'_d} + A^c_d(E^*_s)$ is strongly connected and, therefore, it contains an arborescence $T^* = (V_d, A^*)$ rooted at r. Since T_m is a minimum-cost arborescence in $\overline{G_d}$, we have $\overline{c_d}(A_m) \leq \overline{c_d}(A^*)$.

Because T^* is an arborescence, T^* contains at most two edges of $A^c_d(e)$ for each $e \in E^*_s$. All edges of $A^c_d(e)$ have the same cost $c_s(e)$. Hence $\overline{c_d}(A^*) \leq 2c_s(E^*_s)$. Moreover, $c_s(E''_s) \leq \overline{c_d}(A_m)$ holds, meaning that $c_s(E''_s) \leq 2c_s(E^*_s)$. Because $c(E'') = c_s(E''_s)$, we get $c(E'') = c_s(E''_s) \leq 2c_s(E^*_s) = 2c(E^*)$. □

Remark 3.3 *Although Step 6 of* FSA+1 *keeps all edge of A_b, keeping only one minimum-cost edge for each ordered pair $\{u, v\} \in V_b \times V_b$ with $\langle u, v\rangle \in A_b$ (see pairs $\{l, i\}$, $\{n, h\}$, $\{n, g\}$ in Fig. 6) is sufficient for us to find a minimum-cost arborescence. Our algorithm can be easily modified so that Step 8 may be done in $O(|V|^2)$ time if $\lambda(\Gamma; G')$ is even. Even with this improvement, however, overall time complexity of* FSA+1 *is still $O(\Delta + |V||E|)$ if $\lambda(\Gamma; G')$ is even.* □

Theorem 3.1 FSA+1 *is a 2-approximate algorithm for $(\lambda+1)$ECA-SV with $\lambda = \lambda(G') = \lambda(\Gamma; G')$. Its time complexity (space complexity, respectively) is $O(\Delta + |V||E|)$ ($O(|V|^2 + |E|)$) if λ is even, or $O(\Delta + |E|)$ ($O(|V| + |E|)$) if λ is odd.*

Proof : The correctness is proved by Lemma 3.4. By Lemma 3.5, $PR = 2$ is shown. We are going to show that *FSA+1* takes $O(\Delta + |V||E|)$ time if λ is even, and $O(\Delta + |E|)$ if λ is odd. Constructing G'_s, c_s and b can be done in $O(\Delta + |E|)$ time, where Δ is time complexity of obtaining a structural graph of G' at Step 1 of *FSA+1* and $\Delta = \min\{|V||E'|, |E'| + \lambda^2|V|\log(|V|/\lambda)\}$. Step 2 takes $O(|E|)$ time. Because G'_s has $O(|V|)$ vertices and edges, Steps 3, 4, 5 and 6.1 take $O(|V|)$ time. In Step 6.2, it is necessary to find $O(|E|)$ nearest common descendants for each edge in E_s. It is can be done in $O(|E|)$ time by using an algorithm by [11]. In Step 6.2 (a), only one edge is produced. In Step 6.2 (b1), two edges are produced, while, in Step 6.2 (b2), $O(|V|)$ edges may be produced. So, Step 6.2 takes $O(|V||E|)$ time, where if λ is odd then Step 6.2

(b2) is not executed and $O(|E|)$ time is to be spent in Step 6.2. Step 6.2 takes $O(|V||E|)$ ($O(|E|)$, respectively) time if λ is even (odd). Step 7 is done in $O(|V| + |E|)$ time. Finding a minimum-cost arborescence T_m in Step 8 takes $O(m + n \log n)$ time by using the algorithm of [7], where m (n, respectively) is the number of edges (vertices) in $\overline{G_b}$. Because $\overline{G_b}$ has $O(|V|^2 + |E|)$ edges ($O(|E|)$ edges, respectively) if λ is even (odd), Step 8 takes $O(|V| \log |V| + |V|^2 + |E|)$ time ($O(|V| \log |V| + |E|)$). Steps 9 and 10 take $O(|V|)$ time. Thus the time complexity as stated follows. Explanation of space complexity is omitted. □

4 An optimally solvable case

We can prove that $(\lambda+1)$ECA-SV is solvable in $O(\Delta + |E|)$ time if the following condition holds (meaning that Step 6.2 (b) of *FSA+1* is not executed).

Condition 4.1 *For any edge $(u, v) \in E_s - E'_s$, one of $\{u, v\}$, say u, is an descendant of the other, v, in $\overline{G'_b}$ in Step 6.2 of* FSA+1. □

Theorem 4.1 *If Condition 4.1 holds then E'' given by* FSA+1 *is an optimum solution to $(\lambda+1)$ECA-SV for the case with $\lambda(\Gamma; G') = \lambda(G') = \lambda$, and its time complexity is $O(\Delta + |E|)$.*

Proof : Clearly E'' is a solution to the problem. So it is enough to show $c(E'') \leq c(E^*)$. We consider E''_s, E^*_s and T^* in Lemma 3.5. By the construction of $A^c(E^*_s)$, we have $\overline{c_b}(A^c(E^*_s)) = c_s(E^*_s)$ when Condition 4.1 holds. Because, for $e = (u, v) \in E^*_s$, T^* contains exactly one edge $\langle u, v\rangle$ with $\{\overline{b_d}(\langle u, v\rangle)\} = A^c(e)$, we have $\overline{c_d}(A(T^*)) \leq \overline{c_b}(A^c(E^*_s)) = c_s(E^*_s)$. Thus $c(E'') = c_s(E''_s) \leq \overline{c_d}(A_m) \leq \overline{c_d}(A(T^*)) \leq c(E^*)$.

□

References

[1] K. Eswaran and R. Tarjan, "Augmentation problems," *SIAM J. Comput.*, Vol. 5, pp. 653–655, 1976.

[2] S. Even, *Graph Algorithms*, Pitman, London, 1979.

[3] A. Frank, "Augmenting graphs to meet edge connectivity requirements," *SIAM J. Discrete Mathematics*, Vol. 5, No. 1, pp. 25–53, 1992.

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

[5] H. Gabow, "Applications of a poset representation to edge connectivity and graph rigidity," in *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pp. 812–821, 1991.

[6] H. Gabow, "Efficient splitting off algorithms for graphs," in *Proc. 26th ACM Symposium on Theory of Computing*, pp. 696–705, 1994.

[7] H. Gabow, Z. Galil, T. Spencer, and R. Tarjan, "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs," *Combinatorica*, Vol. 6, No. 2, pp. 109–122, 1986.

[8] H. Gabow, M. Goemans, and D. Williamson, "An efficient approximation algorithm for the survivable network design problem," *Mathematical Programming*, Vol. 82, pp. 13–40, 1998.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[10] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, É. Tardos, and D. P. Williamson, "Improved approximation algorithms for network design problems," in *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*. pp. 223–232, ACM Press, 1994.

[11] D. Harel and R. Tarjan, "Fast algorithm for finding nearest common ancestors," *SIAM J. Comput.*, Vol. 13, pp. 338–355, 1984.

[12] K. Jain, "A factor 2 approximation algorithm for the generalized Steiner network problem," *Combinatorica*, Vol. 21, pp. 39–60, 2001.

[13] A. V. Karzanov and E. A. Timofeev, "Efficient algorithm for finding all minimal edge cuts of a nonoriented graph," *Cybernetics*, pp. 156–162, March-April 1986, Translated from Kibernetika, 2 (1986), 8–12.

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

[15] S. Khuller and U. Vishkin, "Biconnectivity approximations and graph carvings," *Journal of the ACM*, Vol. 41, No. 2, pp. 214–235, 1994.

[16] T. Mashima and T. Watanabe, "Approximation Algorithms for the k-Edge-Connectivity Augmentation Problem," in *1995 IEEE International Symposium on Circuits and Systems*, pp. 155–158, May 1995.

[17] H. Nagamochi, S. Nakamura, and T. Ibaraki, "A simplified $\tilde{O}(nm)$ time edge-splitting algorithm in undirected graphs," *Algorithmica*, Vol. 26, pp. 50–57, 2000.

[18] D. Naor, D. Gusfield, and C. Martel, "A fast algorithm for optimally increasing the edge connectivity," *SIAM J. Comput.*, Vol. 26, No. 4, pp. 1139–1165, Aug. 1997.

[19] S. Taoka and T. Watanabe, "Minimum augmentation to k-edge-connect specified vertices of a graph," in *Lecture Notes in Computer Science 834(D-Z du and X-S Zhang(Eds.): Algorithms and Computation)*, pp. 217–225. Springer-Verlag, Berlin, 1994, (Proc. 5th International Symposium on Algorithms and Computation(ISAAC'94)).

[20] S. Taoka and T. Watanabe, "Efficient augmentation to $(\sigma + 1)$-edge-connect specified vertices of a graph," preparing to publish, 2002.

[21] R. Tarjan, *Data Structures and Network Algorithms*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, 1983.

[22] S. Ueno, Y. Kajitani, and H. Wada, "Minimum augmentation of trees to k-edge-connected graph," *Networks*, Vol. 18, pp. 19–25, 1988.

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

[24] T. Watanabe, T. Mashima, and S. Taoka, "The k-edge-connectivity augmentation problem of weighted graphs," in *Lecture Notes in Computer Science 650, Algorithms and Computation*, T. I. , Y. Inagaki, K. Iwama, T. Nishizeki, and M. Yamashita, Eds. Springer-Verlag, Berlin, 1992, (Proc.3rd International Syposium on Algorithms and Computation(ISAAC'92)).

[25] T. Watanabe, T. Mashima, and S. Taoka, "Approximation algorithms for minimum-cost augmentation to k-edge-connect a multigraph," in *Proc. 1993 IEEE International Symposium on Circuits and systems*, pp. 2556–2559, May 1993.

[26] T. Watanabe and A. Nakamura, "Edge-connectivity augmentation problems," *Journal of Computer and System Sciences*, Vol. 35, No. 1, pp. 96–144, 1987.

[27] T. Watanabe, T. Narita, and A. Nakamura, "3-edge-connectivity augmentation problems," in *Proc. 1989 IEEE International Symposium on Circuits and Systems*, pp. 335–338, 1989.

[28] T. Watanabe, S. Taoka, and T. Mashima, "Approximation algorithms for the 3-edge-connectivity augmentation problem of graphs," in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 424–429, Dec. 1992.

[29] T. Watanabe, S. Taoka, and T. Mashima, "Minimum-cost augmentation to 3-edge-connect all specified vertices in a graph," in *Proc. 1993 IEEE International Symposium on Circuits and Systems*, pp. 2311–2314, May 1993.

[30] T. Watanabe and M. Yamakado, "A linear time algorithm for smallest augmentation to 3-edge-connect a graph," *IEICE Trans. Fundamentals*, Vol. E76-A, No. 4, pp. 518–531, 1993.