アルゴリズム 42－1
（1994．11．18）

グラフの辺多重度の増加を許さない k-辺連結化問題に対する効率的解法

田岡智志 渡辺敏正

広島大学工学部第二類（電気系）
724 東広島市鏡山一丁目 4-1
（電話）0824-24-7662 [渡辺]
（ファクシミリ）0824-22-7195
（電子メール）watanabe@huis.hiroshima-u.ac.jp

重みなしの k辺連結化問題 (UW-kECA と略記) とは，与えられた無向グラフ $G=(V,E)$ にそれを付加して得られるグラフ $G'=(V,E\cup E')$ が k辺連結となるような最小辺集合 E' を求める問題である。本稿では、(1) G が 4 辺連結単純グラフであるときに、G'も単純グラフとなるような辺集合 E'を求める、UW-5ECA に対する $O(|V|\log|V|+|E|)$ 時間アルゴリズムを与える。次に、(2) $3\leq\lambda\leq 4$、且つ G が λ辺連結多重グラフの場合に、G'の辺多重度を増加させないような辺集合 E'を求める、UW-$(\lambda+1)$ECA に対する $O(|V|\log|V|+|E|)$ 時間アルゴリズムを提案する。

キーワード: k-辺連結化問題, アルゴリズム, 多重辺, グラフ, 辺連結度

Efficient Algorithms for the Edge-Connectivity Augmentation Problem of Graphs without Increasing Edge-Multiplicity

Satoshi Taoka and Toshimasa Watanabe

Department of Circuits and Systems, Faculty of Engineering, Hiroshima University,
4-1 Kagamiyama, 1 Chome, Higashi-Hiroshima, 724 Japan
Phone: +81-824-24-7662 (Watanabe) Facsimile: +81-824-22-7195
E-mail: watanabe@huis.hiroshima-u.ac.jp

The unweighted k-edge-connectivity augmentation problem (UW-kECA for short) is described as follows: "Given a graph $G=(V,E)$, find an edge set E' of minimum cardinality such that $G'=(V,E\cup E')$ is k-edge-connected." The paper proposes an $O(|V|\log|V|+|E|)$ algorithm for finding a solution E' to UW-$(\lambda+1)$ECA with the following restriction (1) or (2): (1) $\lambda=4$, G is λ-edge-connected simple graph with $|V|\geq 6$ and G' is also simple; (2) $3\leq\lambda\leq 4$, G is λ-edge-connected multigraph, and adding E' does not increase multiplicity of edges in G.

Key words: k-edge-connectivity augmentation problems, algorithms, multiple edges, graphs, edge-connectivity

1 Introduction

The unweighted *k-edge-connectivity augmentation problem* (UW-kECA for short) is described as follows: "Given a λ-edge-connected graph $G = (V, E)$, find an edge set E' of minimum cardinality such that $G' = (V, E \cup E')$ is $\lambda+\delta$-edge-connected and $\lambda+\delta = k$." We often denote G' as $G + E'$, and E' is called a *solution* to the problem. Let UW-kECA(*,**) denote UW-kECA with the following restriction (i) and (ii) on G and E', respectively: (i) * is set to S if G is required to be simple, and remaining * means G may be a multiple graph; (ii) ** is set to MA if creation of new multiple edges in constructing G' is allowed, and is set to SA otherwise. As for UW-kECA, UW-kECA(*,MA) has mainly been discussed so far. See [3, 5, 7, 11, 17, 18, 19, 20] for the results.

The subject of the present paper is UW-$(\lambda + 1)$ECA(*,SA) with $\lambda = 3$ or 4. This paper proposes an $O(|V| \log |V| + |E|)$ algorithm for each of them.

As known related results, UW-kECA(S,SA) in which G has no edges was first discussed in [6], where the problem more general than UW-kECA(S,SA) is considered. An $O(|V| + |E|)$ algorithm for UW-2ECA(S,SA) can be obtained by slightly modifying the one given in [3] for UW-2ECA(*,MA). As for UW-3ECA(*,SA), [20] proposed an $O(|V| + |E|)$ algorithm for UW-3ECA(*,MA), and showed that if $|V| \geq 4$ then this algorithm finds an optimum solution to UW-3ECA(*,SA). [13] proposed the following two algorithms: (1) an $O(|V| \log |V|+|E|)$ algorithm for solving UW-$(\lambda+1)$ECA(S,SA), where $\lambda = 3$, G is simple and $|V| \geq 5$; (2) an $O(|V|^2 + |E|)$ algorithm for solving UW-$(\lambda+1)$ECA(S,SA), where $\lambda = 3$, G is simple and $|V| \geq 6$. The early version of UW-4ECA(S,SA) has been reported in [12].

A central concept in solving UW-kECA is a *t-component* of G: a maximal set of vertices such that G has at least t edge-disjoint paths between any pair of vertices in the set [19]. A t-component whose degree (the number of edges connecting vertices in the set to those outside of it) is equal to the edge-connectivity of G is called a *leaf*. Although UW-$(\lambda+1)$ECA(*,SA) with $\lambda = 3$ or 4 can be solved almost similarly to general UW-kECA(*,MA), the only difference is that the augmenting step has to choose a pair of leaves, each containing a vertex such that they are not adjacent in G. (Such a pair of leaves is called a *nonadjacent pair*.) This requires addition of another characteristic or a process in finding solutions by means of structural graphs. A structural graph is introduced in [9] and is used as a useful tool that reduces time complexity in finding a solution to UW-kECA(*,MA) in [7, 11].

In addition to structural graphs, [13] adopts the operation, called *edge-interchange*, in finding a solution, where it was introduced in [17, 18] in order to reduce time complexity of [19]. A set of two nonadjacent pairs of leaves is called a *D-combination* if they are disjoint. The augmenting step in solving UW-$(\lambda+1)$ECA(S,SA) in [13] is to repeat both choosing a nonadjacent pair of leaves and enlarging a $(\lambda + 1)$-component by means of edge-interchange (or an analogous operation). Hence obtaining an optimum solution requires finding maximum number of nonadjacent pairs such that any set of two distinct nonadjacent pairs is a D-combination, and it is reduced to finding a maximum matching of a certain graph $R(G)$, called a *leaf-graph*, constructed from G. The main point of UW-$(\lambda+1)$ECA(S,SA) with $\lambda = 3$ or 4 is that there exists a solution E' if $R(G)$ has a maximum matching M: this is not always the case with UW-kECA(S,SA) for $k \geq 6$.

We can avoid obtaining a maximum matching, which is a time-consuming process, except the case where the number of leaves is small. A structural graph of G can be constructed in $O(\lambda^2|V| \log |V|/\lambda+|E|)$ time by the results in [7]. Since it is a tree if $\lambda = 3$ (or if λ is odd in general), finding a solution by edge-interchange operation can be done in $O(|V|)$ time. Based on these observations, an $O(|V| \log |V| + |E|)$ algorithm for solving UW-4ECA(S,SA) was proposed in [13]. On the other hand a structural graph of G is a cactus if $\lambda = 4$ (or if λ is even in general). [13] showed that finding a solution to UW-5ECA(S,SA) can be done in $O(|V|^2 + |E|)$ time.

The results of the present paper are stated as follows: (1) an $O(|V| \log |V| + |E|)$ algorithm for UW-5ECA(S,SA) with $\lambda = 4$ is proposed, reducing $O(|V|^2 + |E|)$ time complexity shown in [13]; (2) $O(|V| \log |V| + |E|)$ algorithms for UW-$(\lambda + 1)$ECA(*,SA) with $\lambda = 3$ or 4 are proposed by extending those algorithms for UW-$(\lambda + 1)$ECA(S,SA) with $\lambda = 3$ or 4, respectively.

When $\lambda = 4$, the procedure proposed in Section 7 finds an edge set E'_s such that if it is not a solution for G then $F(G + E'_s)$ is a tree or a cactus that can be equivalently transformed into a tree. This can be done in $O(|V| \log |V| + |E|)$ time including construction of $F(G + E'_s)$. Since the resulting structural graph is a tree, the algorithm, which is proposed in [13] for $\lambda = 3$, finds a solution E''_s for $G + E'_s$ in $O(|V| \log |V| + |E|)$ time. We can show $E'_s \cup E''_s$ is a solution of G in Section 7. Hence we find a solution of G for UW-$(\lambda + 1)$ECA(S,SA) in $O(|V| \log |V| + |E|)$ time when $\lambda = 4$.

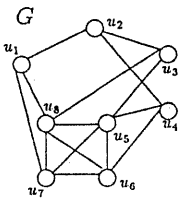

Figure 1: A simple graph G with $\lambda(G) = 3$ and $|LF(G)| = 4$.

2 Preliminaries

2.1 Basic Definitions

Technical terms not specified here can be identified in [1, 4, ?, 16]. An *undirected graph* $G = (V(G), E(G))$ consists of a finite and nonempty set of vertices $V(G)$ and a finite set of undirected edges $E(G)$; an edge e incident upon two vertices u, v in G is denoted by $e = (u, v)$ unless any confusion arises. For disjoint sets S, $S' \subset V(G)$, we denote $(S, S'; G) = \{(u, v) \in E(G) | u \in S \text{ and } v \in S'\}$, where it is often written as (S, S') if G is clear from the context. We denote $d_G(S) = |(S, \overline{S}; G)|$. This is called the *degree* of S (in G). If $S = \{v\}$ then $d_G(\{v\})$ is denoted simply as $d_G(v)$ and is the total number of edges (v, v'), $v' \neq v$, incident upon v. A path between vertices u and v is often called a (u, v)-*path*. For two vertices u, v of G, let $\lambda(u, v; G)$, or simply $\lambda(u, v)$, denote the maximum number of pairwise edge-disjoint paths between u and v.

For a set $X \subseteq V(G)$, let $G[X]$ denote the subgraph having X as its vertex set and $\{(u, v) \in E(G) | u, v \in X\}$ as its edge set. $G[X]$ is called the *subgraph* of G *induced* by X (or the *induced subgraph* of G by X). *Deletion* of $X \subseteq V(G)$ from G is to construct $G[V(G) - X]$, which is often denoted as $G - X$. If $X = \{v\}$ then we often denote $G - v$ for simplicity. Deletion of $Q \subseteq E(G)$ from G defines a spanning subgraph of G, denoted by $G - Q$, having $E(G) - Q$ as its edge set. If $Q = \{e\}$ then we denote $G - e$. For a set E' of edges such that $E' \cap E(G) = \emptyset$, let $G + E'$ denote the graph $(V(G), E(G) \cup E')$. If $E' = \{e\}$ then we denote $G + e$.

Let $S \subseteq V(G) \cup E(G)$ be any minimal set such that $G - S$ has more components than G. S is called a *separator* of G, or in particular a (X, Y)-separator if any vertex of X and any one of Y are disconnected in $G-S$. If $X = \{u\}$ or $Y = \{v\}$ then it is denoted as a (u, Y)-separator or a (X, v)-separator. A *minimum* (X, Y)*-separator* S of G is a (X, Y)-separator of minimum cardinality. Such S is often called a (X, Y)-cut if $S \subseteq E(G)$. It is known that a (u, v)-cut S has $|S| = \lambda(u, v; G)$. A *minimum separator* S of G is a separator of minimum cardinality among all separators of G, and if $S \subseteq E(G)$ then $|S|$ is called the *edge-connectivity* (denoted by $\lambda(G)$) of G; particularly we call such $S \subseteq E(G)$ a *minimum cut* (of G). G is said to be *k-edge-connected* if $\lambda(G) \geq k$. A *k-edge-connected component* (k-component, for short) of G is a subset $S \subseteq V(G)$ satisfying the following (a) and (b): (a) $\lambda(u, v; G) \geq k$ for any pair $u, v \in S$; (b) S is a maximal set that satisfies (a). Let $\Gamma_G(k)$ denote the set of all k-components of G. In a graph G with $\lambda(G) = \lambda$, a $(\lambda + 1)$-component S with $d_G(S) = \lambda + 1$ is called a *leaf* $(\lambda + 1)$*-component* of G. It is known that $\lambda(G) \geq k$ if and only if $V(G)$ is a k-component. Note that distinct k-components are disjoint sets. Each 1-component is often called a *component*. Let $\lceil x \rceil$ ($\lfloor x \rfloor$, respectively) denote the minimum integer not smaller (the maximum one not greater) than x.

A *cactus* is an undirected connected graph in which any pair of cycles share at most one vertex. A *structural graph* $F(G)$ of G with $\lambda(G) = \lambda$ is a representation of all minimum cuts of G. We use the term "nodes of $F(G)$" to distinguish them from vertices of G. $F(G)$ is an edge-weighted cactus of $O(|V|)$ nodes and edges such that each tree edge (an edge which is a bridge in $F(G)$) has weight $\lambda(G)$ and each cycle edge (an edge included in any cycle) has weight $\lambda(G)/2$. Particularly if λ is odd then $F(G)$ is a weighted tree. (Examples of G and $F(G)$ will be given in Figs. 1 and 3.) Each vertex in G maps to exactly one node in $F(G)$, and $F(G)$ may have some other nodes, call empty nodes, to

which no vertices of G are mapped. Let $\epsilon(G) \subseteq V(F(G))$ denote the set of all empty nodes of $F(G)$. Note that any minimum cut of G is represented as either a tree edge or a pair of two cycle edges in the same cycle in $F(G)$, and vice versa. Let ρ: $V(G) \rightarrow V(F(G)) - \epsilon(G)$ denote this mapping. We use the following notations $\rho(X) = \{\rho(v)|v \in X\}$ for $X \subseteq V$, and $\rho^{-1}(Y) = \{u \in V|\rho(u) \in Y\}$ for $Y \subseteq V(F(G))$. $\rho(\{v\})$ or $\rho^{-1}(\{v\})$ is written as $\rho(v)$ or $\rho^{-1}(v)$, respectively, for notational simplicity. For $v \in V(F(G))$, if summation of weight of all edges that are incident to v in $F(G)$ equals to λ, then v is called a *leaf node* (that is a degree-1 vertex in a tree or a degree-2 vertex in a cycle). It is shown that $F(G)$ can be constructed in $O(nm)$ time [9] or in $O(\lambda^2 n \log(n/\lambda) + m)$ time [7], where $m = |E(G)|$ and $n = |V(G)|$.

For two vertices v, v' of a tree T, any vertex u such that both a (u, v)-path and a (u, v')-path exist is called a *common ancestor* of v and v'. Suppose that all vertices of T are numbered in the order they are visited in a depth-first search starting from a vertex that is not a leaf. Let $dfn(v)$ denote this number for each v of T. The vertex v with $dfn(v) = \max\{dfn(u)|u$ is a common ancestor of v and $v'\}$ is called the *lowest common ancestor* (of v and v') [2].

Lemma 2.1 *[5] For distinct two sets $X, Y \subset V$, we have*

$$d(X) + d(Y) = d(X - Y) + d(Y - X) + 2|(X - Y, Y - X)| \text{ and } (2.1)$$

$$d(X) + d(Y) = d(X \cap Y) + d(X \cup Y) + 2|(V - X \cup Y, X \cap Y)|. \quad (2.2)$$

2.2 λ-Components and Leaf-Graphs [13]

Let $\lambda(G) = \lambda > 0$. Let X_1, X_2 be distinct $(\lambda + 1)$-components of G. The pair $\{X_1, X_2\}$ are called an *adjacent pair* (denoted as $X_1 \chi X_2$) if any two vertices $w \in X_1$ and $w' \in X_2$ are adjacent in G, or called a *nonadjacent pair* (denoted as $X_1 \overline{\chi} X_2$) otherwise. Let $LF(G) = \{X|X$ is a leaf of $G\}$ and $V' = \{v|v$ represents a leaf of $G\}$. Let $L(v)$ denote the $(\lambda + 1)$-component corresponding to $v \in V'$ and let u denote a representative of $L(v)$, where $u \in L(v)$, and we choose u from $L(v)$ whenever necessary. Let $R(G) = (V', E')$ be defined by $E' = \{(v, v')|v, v' \in V'$ and $L(v) \overline{\chi} L(v')\}$, and it is called the *leaf-graph* of G.

Property 2.1 *$R(G)$ is simple.*

Let L_i, $i = 1, 2, 3, 4$, be distinct leaves of G. A set of two nonadjacent pairs $\{\{L_1, L_2\}, \{L_3, L_4\}\}$ is called a *D-combination* if they are disjoint (that is, $\{L_1, L_2\} \cap \{L_3, L_4\} = \emptyset$). In general, for $2t$ distinct leaves L_i, $i = 1, \ldots, 2t$ with $t \geq 2$, of G, t nonadjacent pairs $\{L_1, L_2\}, \ldots, \{L_{2t-1}, L_{2t}\}$ are called a *D-set* of G if any two of them are disjoint. Let $L_1 \chi \{L_2, L_3\}$ denote that both $L_1 \chi L_2$ and $L_1 \chi L_3$ hold. A D-combination $\{\{L_1, L_2\}, \{L_3, L_4\}\}$ is called an *I-combination* (denoted as $\{L_1, L_2\} \angle \{L_3, L_4\}$) if one of $L_1 \chi \{L_3, L_4\}$ and $L_2 \chi \{L_3, L_4\}$ holds. We first show some basic results on $R(G)$ and leaves of G.

Proposition 2.1 *Suppose G is simple. Either $|L| = 1$ or $|L| \geq \lambda + 2$ holds for $\forall L \in LF(G)$.*

Proposition 2.2 *Suppose G is simple. If $\{L_1, L_2\} \subseteq LF(G)$ is an adjacent pair then $|L_1| = |L_2| = 1$.*

Proposition 2.3 *$d_{R(G)}(v) \geq \max\{|V'| - (\lambda + 1), 0\}$ for any $v \in V'$.*

Proposition 2.4 *Suppose $\lambda = 3$. Let $Y = \{L_1, L_2, L_3, L_4\} \subseteq LF(G)$, where all elements are distinct. Then (1) Y contains at least one nonadjacent pair; (2) if $\{L_1, L_2\}$ is a nonadjacent pair and we have a pair $\{L_5, L_6\} \subseteq LF(G) - Y$ then there is another nonadjacent pair $\{W_1, W_2\} \subseteq \{L_3, L_4, L_5, L_6\}$.*

2.3 Examples

Let $G = (V, E)$ with $|V| \geq \lambda + 2$ and $\lambda(G) = \lambda$ be any given simple graph for $\lambda = 3, 4$. Let OPT(M) or OPT(S) denote the cardinality of an optimum solution to UW-$(\lambda + 1)$ECA(*,MA) or to UW-$(\lambda + 1)$ECA(S,SA) for G, respectively. For $\lambda = 3$, we give an example such that OPT(S)=OPT(M)+1. Fig. 1 shows a graph G with $|LF(G)| = 4$. $R(G)$ are shown in Fig. 2. A structural graph $F(G)$ of G is shown in Fig. 3. $\{(u_1, u_3), (u_2, u_4)\}$ is a solution to UW-4ECA(*,MA), while $E' = \{(u_1, u_6), (u_2, u_5), (u_3, u_4)\}$ is a solution to UW-4ECA(S,SA) and OPT(S)=3=OPT(M)+1.

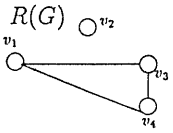

Figure 2: The leaf-graph $R(G)$ of G in Fig. 1.

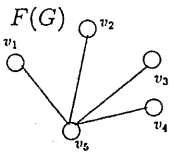

Figure 3: A structural graph $F(G)$ of G in Fig. 1, where all edge-weights are 3 and are not written. In this case leaves L_i in $LF(G)$ the graph G shown in Fig. 1 are represented as nodes v_i of $F(G)$ for $i = 1, \ldots, 5$: it may happen that $F(G)$ has a node to which no corresponding leaf of $LF(G)$ exists.

3 Maximum Matchings of Leaf-Graphs

One of requirements in finding an optimum solution to UW-$(\lambda+1)$ECA with $\lambda \geq 1$ is to obtain a largest D-set. Hence, in this section, the cardinality of a maximum D-set is investigated by considering a maximum matching M of $R(G)$.

Let M denote a maximum matching of $R(G)$ in the following discussion unless otherwise stated, where we assume that $\lambda(G) = \lambda$ with $\lambda \geq 1$. Put $V(M) = \{u, v|(u, v) \in M\}$.

Proposition 3.1 *[13] $|M|$ satisfies one of the following (1) and (2):*

(1) if λ is odd and $|V'| = 2\lambda$ then

$$\lfloor |V'|/2 \rfloor - 1 \leq |M| \leq \lfloor |V'|/2 \rfloor;$$

(2) otherwise

$$\max\{0, \min\{|V'| - \lambda, \lfloor |V'|/2 \rfloor\}\} \leq |M| \leq \lfloor |V'|/2 \rfloor.$$

Corollary 3.1 *[13] Suppose $|V'| = 2\lambda$ and $\lambda = 2m + 1$. If $|M| = \lfloor |V'|/2 \rfloor - 1$ then $G = (V, E)$ is a complete bipartite graph with $V = X \cup Y$, $X \cap Y = \emptyset$, $|X| = |Y| = \lambda$ and $E = \{(x, y)|x \in X, y \in Y\}$.*

Proposition 3.2 *If $|V'| = \lambda$ then $1 \leq |M| \leq \lfloor |V'|/2 \rfloor$.*

4 Augmentation by Edge-Interchange

We explain an operation called edge-interchange which was originally introduced in [17, 18] for an efficient augmentation. It is also used in [12, 13, 15, 14]. Let $LF(G) = \{Y_1, \ldots, Y_q\}$ denote the class of all leaves of G and choose $y_i \in Y_i$ as a representative of Y_i. Let

$$Y = \{y_i|Y_i \in LF(G)\}, \quad q \geq 2,$$

and let $r = \lceil q/2 \rceil$. We denote $V(e) = \{u, v\}$ for an edge $e = (u, v)$ and $V(F) = \bigcup_{e \in F} V(e)$ for an edge set F.

We can easily prove the next proposition.

Proposition 4.1 *If there is an attachment F for G such that $V(F) = Y \subseteq S$ for some $S \in \Gamma_{G+F}(\lambda + 1)$ then $S = V(G)$.*

4.1 Attachments

We have $d_G(Y_i) \geq \lambda$ and $\lambda(y_i, y_j; G) = \lambda$ for $\forall y_i, y_j \in Y$ $(i \neq j)$. A edge set F is called an *attachment* (for G) if and only if the following (1) through (4) hold:

(1) $V(F) \subseteq Y$,

(2) $F \cap E(G) = \emptyset$,

(3) $V(e) \neq V(e')$ $(\forall e, e' \in F, e \neq e')$, and

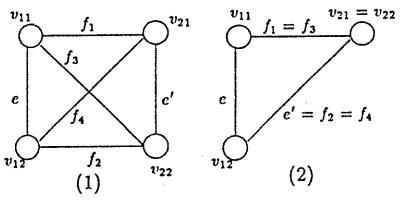

Figure 4: The edges e, e' and f_i, $1 \le i \le 4$: (1) $v_{21} \neq v_{22}$; (2) $v_{21} = v_{22}$.

(4) F has at most one pair f, f' such that $|V(f) \cap V(f')| = 1$ if q is odd, and has no such pair otherwise.

Let F be any attachment for G. For each $e = (u, v) \in F$, $G + F$ has a new $(\lambda + 1)$-component, denoted by $\alpha(e, G + F)$, containing $V(e)$.

We will show that we can find a minimum attachment $Z(\lambda + 1) = \{e_1, \ldots, e_r\}$ such that $\lambda(G + Z(\lambda + 1)) = \lambda + 1$, where $r = \lceil q/2 \rceil$. Although there are two cases: $r = 1$ and $r \ge 2$, we discuss only the latter case in the following. (Note that if $r = 1$ then we immediately obtain the desired attachment F.)

4.2 Finding a minimum attachment

Suppose that there are an attachment F for G and vertices $v_{ij} \in Y - V(F)$, $1 \le i, j \le 2$, where v_{11}, v_{12}, v_{21} are distinct, and if v_{22} is equal to one of the other three then we assume that $v_{22} = v_{21}$ (see Fig. 4). We use the following notations:

$$L = G + F, \quad e = (v_{11}, v_{12}), \quad e' = \begin{cases} (v_{21}, v_{22}) & \text{if } v_{21} \neq v_{22} \\ (v_{12}, v_{21}) & \text{if } v_{21} = v_{22} \end{cases}$$

$$\alpha(e) = \alpha(e, L + \{e, e'\}), \quad \alpha(e') = \alpha(e', L + \{e, e'\}),$$

$$f_1 = (v_{11}, v_{21}), \quad f_2 = (v_{12}, v_{22}), \quad f_3 = (v_{11}, v_{22}), \quad f_4 = (v_{12}, v_{21}),$$

where we set $f_1 = f_3$ and $e' = f_2 = f_4$ if $v_{21} = v_{22}$.

$$\alpha(f_i) = \begin{cases} \alpha(f_i, L + \{f_1, f_2\}) & \text{if } 1 \le i \le 2 \\ \alpha(f_i, L + \{f_3, f_4\}) & \text{if } 3 \le i \le 4 \end{cases}$$

(Note that $e, e', f_i \notin E(L), 1 \le i \le 4$.) We have Case I: $\alpha(e) \cap \alpha(e') = \emptyset$; Case II: $\alpha(e) \cap \alpha(e') \neq \emptyset$ (that is, $\alpha(e) = \alpha(e')$). For Case I, we will show that there are two edges f, f' with $V(f) \cup V(f') = V(e) \cup V(e')$ such that

$$V(e) \cup V(e') \subseteq \alpha(f, L + \{f, f'\}) = \alpha(f', L + \{f, f'\}).$$

That is, we can add two edges so that one $(\lambda+1)$-component containing $V(e) \cup V(e')$ may be obtained. Finding and adding such a pair of edges f, f' is called *edge-interchange* (with respect to $V(e_1) \cup V(e_2)$).

Suppose that $\alpha(e) \cap \alpha(e') = \emptyset$. Note that $v_{21} \neq v_{22}$ in this case. Let K be any fixed $(\alpha(e), \alpha(e'))$-cut of $L + \{e, e'\}$, and let B_i, $1 \le i \le 2$, denote the two sets of $L + \{e, e'\}$ such that $B_1 \cup B_2 = V$, $B_2 = V - B_1$, $K = (B_1, B_2; L + \{e, e'\})$, $\alpha(e) \subseteq B_1$ and $\alpha(e') \subseteq B_2$. $|K| = \lambda = \lambda(v_1, v_2; L'')$ for $\forall v_i \in B_i$, $1 \le i \le 2$, where L'' denotes L, $L + e$, $L + e'$ or $L + \{e, e'\}$. K is a (v_1, v_2)-cut of L. Suppose that f and f' satisfy either (i) or (ii):

(i) $f = f_1$, $f' = f_2$, or (ii) $f = f_3$, $f' = f_4$,

where $\{f, f'\} \cap E(L) = \emptyset$.

The next proposition shows a property of edge-interchange.

Proposition 4.2 *[13, 17, 18] If $\alpha(e) \cap \alpha(e') = \alpha(f_1) \cap \alpha(f_2) = \emptyset$ then $\alpha(f_3) \cap \alpha(f_4) \neq \emptyset$, that is, $\alpha(f_3) = \alpha(f_4)$.*

Such a pair f_3, f_4 of Proposition 4.2 are called an *edge-interchange pair* of L.

Corollary 4.1 *[13] Let f_3, f_4 be the two edges of Proposition 4.2, $L' = L + \{f_3, f_4\}$ and f be either f_3 or f_4. Then $L' - f$ has no λ-cut separating $V(f_3)$ from $V(f_4)$. That is, if $L' - f$ has a λ-cut K separating a vertex of $V(f_3)$ from another one of $V(f_4)$ then K separates $\{u\}$ from $\{v\} \cup V(f')$ and $V(f')$ is not separated by K, where $V(f) = \{u, v\}$ and $\{f'\} = \{f_3, f_4\} - \{f\}$.*

If $\lambda > 0$ then repeating edge-interchange finds a sequence of edges, $e_1, \ldots, e_r$ ($r = \lceil q/2 \rceil \ge 1$), as follows:

$$V(e_{j-1}) \cap V(e_j) = \emptyset, \quad 2 \le j \le r - 1,$$

$$V(e_{r-1}) \cap V(e_r) = \begin{cases} \emptyset & \text{if } q \text{ is even,} \\ \{y_{q-1}\} & \text{if } q \text{ is odd,} \end{cases}$$

and, for each e_i, $1 \le i \le r - 2$, there is an edge $e_i' = (y_j, y_k)$ with $y_j \in Y_j$ and $y_k \in Y_k$ such that e_i and e_i' are an edge-interchange pair, where $G_0 = G$, and $G_{i+1} = G_i + e_{i+1}$, $0 \le i \le r - 1$. The details are given in the statement of algorithms shown later. For notational convenience, we denote

$$e_i = (y_{2i-1}, y_{2i}) \text{ with } y_{2i-1} \in Y_{2i-1} \cap Y \text{ and } y_{2i} \in Y_{2i} \cap Y$$

$$\text{for } 1 \le i \le r = \lceil q/2 \rceil,$$

where if q is odd then we set $Y_{q+1} = Y_{q-1}$.

Proposition 4.3 *[13, 17, 18] $Z(\lambda + 1) = \{e_1, \ldots, e_r\}$ is a minimum attachment such that $\lambda(G') = \lambda + 1$, where $G' = G + Z(\lambda + 1)$.*

From Corollary 4.1, another important property of edge-interchange is obtained.

Proposition 4.4 *[13, 17, 18] G_i has a leaf containing $\alpha(e_i, G_i)$ if and only if the number of leaves of G_{i-1} is three.*

Remark 4.1 *[13] Let f, f' be the two new edges to be added to $L = G + F$ such that*

$$V(f) \cap V(f') = \emptyset, V(f) \cup V(f') = \{v_{11}, v_{12}, v_{21}, v_{22}\} \text{ and}$$

$$v_{ij} \in Y - V(F), \ 1 \le i, j \le 2$$

as in Proposition 4.2. Suppose that we are going to check whether $\alpha(f, G_i + \{f, f'\}) \cap \alpha(f', G_i + \{f, f'\}) = \emptyset$ or not. A maximum flow algorithm can be used. Note that we have only to apply the algorithm to $G + \{f, f'\}$ (not to $G_i + \{f, f'\}$) or to $G + \{g, g'\}$. Thus this can be done in $O(\phi(|V|, |E| + 2))$ time, and we assume that a maximum flow algorithm for H can be done in $\phi(|V|, |E|)$ time. [10] introduced a sparse graph $G'' = (V, E'')$ for a given graph $G = (V, E)$ such that the following (1) through (3) hold for any $u, v \in V$:

(1) $\lambda(u, v; G'') = \lambda$ if $\lambda(u, v; G) \ge \lambda$;

(2) $\lambda(u, v; G'') = \lambda(u, v; G)$ if $\lambda(u, v; G) < \lambda$;

(3) $E'' \subseteq E$ and $|E''| \le \lambda(|V| - 1)$.

[10] showed that G'' can be obtained in $O(|V| + |E|)$ time. By utilizing the results in [4], above checking operation can be done in $O(\lambda^2 |V|)$ time.

4.3 Edge-interchange operation on a structural graph if λ is odd

The subject has been discussed in [13], and the results are summarized in the following.

Let f, f' be the two new edges such that $V(f) \cap V(f') = \emptyset$, and $V(f) \cup V(f') = \{v_{11}, v_{12}, v_{21}, v_{22}\}$ as in Proposition 4.3. Suppose that we are going to check whether $\alpha(f, G_i + \{f, f'\}) \cap \alpha(f', G_i + \{f, f'\}) = \emptyset$ or not. We will show that, if $\lambda = \lambda(G) = 3$ (or odd, in general), finding $Z(\lambda + 1)$ of Proposition 4.3 can be done in $O(|V|)$ time by using a structural graph $F(G)$, which is a tree in this case, where we assume that $F(G)$ has already been available. Clearly $F(G)$ also has q leaf nodes and may be considered as $F(G_0)$. By executing a depth-first search starting from a node that is not a leaf node, we can assign all leaf nodes $v_1, \ldots, v_q$ of $F(G)$ as $1, \ldots, q$ in the order they are visited, where we assume that $q \ge 4$. Let $dfn(v_i)$ denote this number assigned to v_i and let T denote the depth-first tree that is treated as a directed tree or an undirected tree interchangeably. When the search returns from a node v to its parent u we obtain a value $\min(v)$, where $\min(v)$ is the minimum of $dfn(v')$ among all leaf nodes v' in the subtree rooted at v of T (see Fig. 5), where we set $\min(v) = 0$ for any leaf node v. The values $dfn(v)$ and $\min(v)$ for all nodes v of $F(G)$ can be obtained in $O(|V(F(G))|)$ time, which is $O(|V|)$. Let v', v'' be any leaf nodes of T with $dfn(v') < dfn(v'')$. We trace back from v'' toward the root of T up to the first node w such that $dfn(v') \ge \min(w)$. Then w is the lowest common ancestor of v' and v'', and the (v', v'')-path of T can be obtained.

Let $f = (a, b)$ and $f' = (c, d)$ for simplicity, where $\{a, b, c, d\} = \{v_{11}, v_{12}, v_{21}, v_{22}\}$. Let P_{ab} (P_{cd}, respectively) be the path between a and b (between c and d) in $F(G_i)$ ($i = 0$ initially). Note that $\alpha(f, G_i + \{f, f'\}) \cap \alpha(f', G_i + \{f. f'\}) = \emptyset$ if and only if $V(P_{ab}) \cap V(P_{cd}) = \emptyset$. The checking operation consists of two procedures: the first one is to

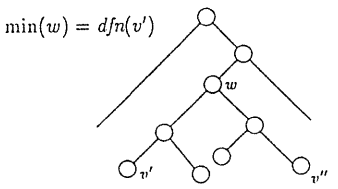

Figure 5: A schematic explanation of leaf nodes and their lowest common ancestors.

detect whether $V(P_{ab}) \cap V(P_{cd}) = \emptyset$ or not; the second one shrinks $V(P_{ab}) \cup V(P_{cd})$ into a node in order to construct $F(G_i + \{f, f'\}) = F(G_{i+1})$ whenever $V(P_{ab}) \cap V(P_{cd}) \neq \emptyset$. These two procedures can be combined into one procedure, but, for ease of understanding, we explain them separately.

We provide a characteristic vector C of length $|V|$ and a stack S. We first find P_{ab} which consists of a path P_{aw} from a to w plus another one P_{wb} from w to b, where nodes are kept from a to w and then from w to b in S. We set $C(v) = 1$ if only if $v \in V(P_{ab})$. Similarly we continue finding P_{cd}. In this second search, however, we check whether $C(v) = 1$ or not, for each visited node v. Once there is a node v with $C(v) = 1$ then $V(P_{ab}) \cap V(P_{cd}) \neq \emptyset$, and after P_{cd} is found, we proceed to the second procedure. Otherwise we will end up with resetting $C(v) = 0$ by popping each $v \in V(P_{cd})$ out of S.

In the second procedure the two paths P_{ab} and P_{cd} are shrunk into a node x so that $F(G_i + \{f, f'\})$ may be constructed, similarly to [11] and we add two virtual edges $(c, x), (d, x)$ to $F(G_i + \{f, f'\})$. Hence edges in these two paths are not visited again by the subsequence searches.

If $V(P_{ab}) \cap V(P_{cd}) = \emptyset$ then only one more combination, say (a, c) and (b, d), will be checked. Hence each edge of $E(P_{ab}) \cup E(P_{cd})$ are visited at most twice in the first procedure, and similarly in the second procedure. Therefore, edge-interchange operation for all leaf nodes $v_1, \ldots, v_q$ can be done in $O(|V|)$ time since amortizing time spent to search all virtual edges is $O(|V|)$.

5 Solving UW-4ECA(S,SA) with $\lambda = 3$

It is shown in Section 4 that repeating edge-interchange constructs $Z(\lambda + 1)$. Adding $Z(\lambda + 1)$ to G, however, may create multiple edges. Hence we have to choose nonadjacent leaves of G during edge-interchange. This is done in this section by combining the results given in Sections 3 and 4. The main point in this and the following sections is that there exists a solution E' if $R(G)$ has a maximum matching M. [13] has already solved the problem, and the results are summarized in the following.

Let $G = (V, E)$, with $|V| \geq 5$ and $\lambda(G) = 3$, be any given simple graph. Let $LF(G) = \{L_i | 1 \leq i \leq q\}$ and $V' = \{v_1, \ldots, v_q\}$ $(q = |V'|)$, $I = \{1, \ldots, q\}$. An edge set E' of minimum cardinality such that $G + E'$ is a simple graph with $\lambda(G + E') = \lambda + 1$ $(\lambda = 3, 4)$ is called a *solution*.

The following Propositions 5.1 and 5.2 are obtained for general λ.

Proposition 5.1 *[13] Let SOL be a solution for G and M be a maximum matching of $R(G)$. Then*

$$|V'| - |M| \leq |SOL|. \tag{5.1}$$

Proposition 5.2 *[13] Suppose that G is λ-edge-connected. If $q = 2$ then the following (1) or (2) holds.*

(1) If $L_1 \overline{\chi} L_2$ then $|M| = 1$, $E' = \{(u_1, u_2)\}$ is a solution, and $OPT(S) = OPT(M) = 1$.

(2) If $L_1 \chi L_2$ then $|M| = 0$, there is a vertex $x \in V$ such that $E' = \{(u_1, x), (u_2, x)\}$ is a solution, and $OPT(S) = 2 = OPT(M) + 1$.

Proposition 5.3 *[13] If $q = 3$ then there are distinct edges e_1, e_2 such that $E' = \{e_1, e_2\}$ is a solution, and $OPT(S) = OPT(M) = 2$.*

Proposition 5.4 *[13] Suppose that $q \geq 4$, and let $J = \{1, 2, 3, 4\}$, $V_J = \{L_1, L_2, L_3, L_4\}$ and $L_J = \{L_j | j \in J\}$.*

(1) If V_J consists of a D-combination then V_J can be partitioned into a D-combination, say $\{\{L'_1, L'_2\}, \{L'_3, L'_4\}\}$, such that $G' = G + \{(u'_1, u'_2), (u'_3, u'_4)\}$ is a simple graph having a 4-component S that contains every $L'_j \in L_J$, where $u'_j \in L'_j$, $j \in J$.

(2) If V_J does not consist of a D-combination then there are three distinct edges e_1, e_2, e_3 such that $G' = G + \{e_1, e_2, e_3\}$ is a simple graph having a 4-component S that contains every $L_j \in L_J$.

By combination Propositions 5.1 and 5.4, we obtain the following Corollary 5.1.

Corollary 5.1 *[13] If $q = 4$ then the following (1) or (2) holds.*

(1) If V_J consists of a D-combination then $OPT(S) = OPT(M) = 2$.

(2) If V_J does not consist of a D-combination then $OPT(S) = 3 = OPT(M) + 1$. □

Based on these results, we propose an $O(|V| \log |V| + |E|)$ algorithm *GS3* for solving UW-4ECA(S,SA) for G with $\lambda(G) = 3$.

Algorithm *GS3*
/* Input: a simple graph $G = (V, E)$ with $\lambda(G) = 3$ */
/* Output: a solution E' */
begin
1. construct a structural graph $F(G)$ of G; $V' \leftarrow$ {all leaves of $F(G)$};
2. construct $R(G) = (V', E')$; $LF' \leftarrow LF(G)$; $E' \leftarrow \emptyset$;
3. if $|LF'| \geq 7$ then
   begin
   choose nonadjacent pairs $D_1 = \{L_1, L_2\}$, $D_2 = \{L_3, L_4\}$ with $D_1 \cap D_2 = \emptyset$ and $D_1 \cup D_2 \subseteq LF'$;
   goto Step 6
   end;
4. if $5 \leq |LF'| \leq 6$ then
   find a maximum matching M of $R(G)$
   else /* $|LF'| \leq 4$ */ goto Step 8;
5. $D_1 \leftarrow \{L_1, L_2\}$ $((v_1, v_2) \in M$ and $L_i = L(v_i)$ for $i = 1, 2)$;
   $D_2 \leftarrow \{L_3, L_4\}$ $((v_3, v_4) \in M - \{(v_1, v_2)\})$;
   and $L_i = L(v_i)$ for $i = 3, 4$);
   /* edge-interchange is going to be done */
6. find a D-combination $\{\{L'_1, L'_2\}, \{L'_3, L'_4\}\}$, whose union is $D_1 \cup D_2$, such that two edges $f_1 = (u'_1, u'_2)$, $f_2 = (u'_3, u'_4)$ with $u'_j \in L'_j$, $j = 1, 2, 3, 4$, satisfy that $G + \{f_1, f_2\}$ is a simple graph having a 4-component S with $L'_i \subseteq S$, $i = 1, 2, 3, 4$;
   construct $F(G + \{f_1, f_2\})$ by shrinking S into one vertex x;
   $F(G) \leftarrow F(G + \{f_1, f_2\}) + \{(v'_3, x), (v'_4, x)\}$;
7. $LF' \leftarrow LF' - \{L'_1, L'_2\}$;
   $R(G) \leftarrow R(G) - \{v'_1, v'_2\}$ (with $L'_i = L(v_i)$ for $i = 1, 2$);
   $E' \leftarrow E' \cup \{f_1\}$; goto Step 3;
8. Find a solution E'' for $G'' = (V, E \cup E')$ by using Propositions 5.2 and 5.3 and Corollary 5.1; $E' \leftarrow E' \cup E''$
end.

Theorem 5.1 *[13] The algorithm GS3 correctly finds a solution E' to UW-4ECA(S,SA) for any given G with $\lambda(G) = 3$ in $O(|V| \log |V| + |E|)$ time.*

6 Solving UW-5ECA(S,SA) with $\lambda = 4$

In this section, let $G = (V, E)$, with $|V| \geq 6$ and $\lambda(G) = 4$, be any given graph. [13] has already proposed an $O(|V|^2 + |E|)$ algorithm for the problem. The results are stated in this section, and then an improved algorithm will be given in the next section. The discussion proceeds almost analogously to the previous section; there are, however, some differences.

Proposition 6.1 *[13] Let $M' = \{(v_{2i-1}, v_{2i}) | 1 \leq i \leq |M'|\} \subseteq M$ and suppose $|M'| \geq \lceil \lambda/2 \rceil + 1$. Then one of the following (1) and (2) holds.*

(1) There are indices i, j with $1 \leq i, j \leq |M'|$ and $i \neq j$ such that $\{L_{2i-1}, L_{2i}\} \not\parallel \{L_{2j-1}, L_{2j}\}$, where $L_j = L(v_j)$ for $v_j \in V'$.

(2) For each $(v_{2i-1}, v_{2i}) \in M'$, $G' = G + \{(u_{2i-1}, u_{2i})\}$ is a simple graph having a $(\lambda + 1)$-component X, with $L_{2i-1} \cup L_{2i} \subseteq X$, such that X is not a leaf $(\lambda + 1)$-component.

In the subsequent Sections 6.1 and 6.2, we consider a solution in the case with $|LF(G)| \leq 6$, and then we propose an algorithm *GS4* in Section 6.3 by handling the case with $|LF(G)| \geq 7$.

6.1 Solutions when $|LF(G)| \leq 6$ and $|M| = \lfloor |LF(G)|/2 \rfloor$

The following Propositions 6.2 through 6.6 consider the case where $|LF(G)| \leq 6$ and $|M| = \lfloor |LF(G)|/2 \rfloor$.

Proposition 6.2 *[13] If $LF(G) = \{L_1, L_2\}$ and $L_1 \overline{\chi} L_2$ (that is, $|M| = 1$) then $\{(u_1, u_2)\}$ is a solution for G, showing that $OPT(S) = OPT(M) = |M|$. If $LF(G) = \{L_1, L_2, L_3\}$ and $L_1 \overline{\chi} L_2$ then $\lceil |M| = \lfloor 3/2 \rfloor = 1$ and) there is a solution $\{e_1, e_2\}$ for G such that $V(\{e_1, e_2\}) \subseteq L_1 \cup L_2 \cup L_3$, showing that $OPT(S) = OPT(M) = |M| + 1$.*

Proposition 6.3 *[13] Suppose $|LF(G)| \geq 4$ and $\{L_1, L_2, L_3, L_4\} \subseteq LF(G)$. If $(u_1, u_4) \in E$ with $u_i \in L_i$ for $i = 1, 4$ then both of the following 4-cuts $(S, \overline{S})$ and $(T, \overline{T})$ do not exist: a 4-cut $(S, \overline{S})$ with $L_1 \cup L_2 \subseteq S$ and $L_3 \cup L_4 \subseteq \overline{S}$; a 4-cut $(T, \overline{T})$ with $L_1 \cup L_3 \subseteq T$ and $L_2 \cup L_4 \subseteq \overline{T}$.*

A minimum cut is called an *edge-increasing cut* (EI-cut, for short) with respect to a pair $\{L_1, L_2\}, \{L_3, L_4\}$ if the pair $\{L_1, L_2\}$ and $\{L_3, L_4\}$ is an I-combination and if $(S, \overline{S})$ with $L_1 \cup L_2 \subseteq S$ and $L_3 \cup L_4 \subseteq \overline{S}$ is a minimum cut.

Proposition 6.4 *[13] Suppose $|LF(G)| \geq 4$, $\{L_1, L_2, L_3, L_4\} \subseteq LF(G)$, and $L_i \overline{\chi} L_{i+1}$ for $i = 1, 3$. Then the following (1) or (2) holds.*

(1) If G has no EI-cut with respect to a pair $\{L_1, L_2\}, \{L_3, L_4\}$ then there is a set $\{e_1, e_2\}$ such that $G + \{e_1, e_2\}$ has a 5-component containing $\{u \in L_i | 1 \leq i \leq 4\}$: therefore if $|LF(G)| = 4$ then $OPT(S) = OPT(M) = |M|$.

(2) Otherwise, there is a set $\{e_1, e_2, e_3\}$ such that $G + \{e_1, e_2, e_3\}$ has a 5-component containing $\{u \in L_i | 1 \leq i \leq 4\}$, and adding at most two edges to G does not create such a 5-component: therefore if $|LF(G)| = 4$ then $\{e_1, e_2, e_3\}$ is a solution for G, and $OPT(S) = OPT(M) + 1 = |M| + 1$.

Proposition 6.5 *[13] Let $LF(G) = \{L_i | 1 \leq i \leq 5\}$. Then $\lceil |M| = \lfloor 5/2 \rfloor = 2$ and) there exists a solution $\{e_i | 1 \leq i \leq 3\}$ for G, showing that $OPT(S) = OPT(M) = |M| + 1$.*

Proposition 6.6 *[13] Let $LF(G) = \{L_i | 1 \leq i \leq 6\}$. Then $\lceil |M| = \lfloor |LF(G)|/2 \rfloor = 3$ and) there is a solution $\{e_i | 1 \leq i \leq 3\}$ for G, showing that $OPT(S) = OPT(M) = |M|$.*

6.2 Solutions when $|LF(G)| \leq 6$ and $|M| = \lfloor |LF(G)|/2 \rfloor - 1$

The following Proposition 6.7 is obtained when $|LF(G)| \leq 6$ and $|M| = \lfloor |LF(G)|/2 \rfloor - 1$.

Proposition 6.7 *[13] Let $LF(G) = \{L_i | 1 \leq i \leq k\}$. If $2 \leq |LF(G)| = k \leq 6$ and $|M| = \lfloor |LF(G)|/2 \rfloor - 1$, then there is a solution $\{e_i | 1 \leq i \leq |LF(G)| - |M|\}$ for G, and $OPT(S) = OPT(M) + 1$.*

6.3 An algorithm *GS4* and its time complexity

In this subsection, we propose an algorithm *GS4* which finds a solution E' for G with $\lambda(G) = 4$, and show its correctness and time complexity.

Algorithm *GS4*
/* Input: a simple graph $G = (V, E)$ with $\lambda(G) = 4$ */
/* Output: a solution E' for G */
begin
1. construct a structural graph $F(G)$ of G; $V' \leftarrow$ {all leaves of $F(G)$};
2. construct $R(G) = (V', E')$; $LF' \leftarrow LF(G)$;
3. $E' \leftarrow \emptyset$; $E'' \leftarrow \emptyset$;
4. if $|LF'| \geq 7$ then
   begin
   if $|LF'| \neq 8$ then $k \leftarrow 3$
   else $k \leftarrow 4$;
   find a matching $M' = \{(v_{2i-1}, v_{2i}) | 1 \leq i \leq k\}$
   with $\{L(v) | v \in V(M')\} \subseteq LF'$;
   /* use Proposition 6.1 in the following */
   if there are two edges $e_1 = (u_1, u_2), e_2 = (u_3, u_4)$ such that
   $e_1, e_2 \in M$ and $\{L_1, L_2\} \not\subseteq \{L_3, L_4\}$ then
   begin
   $D_1 \leftarrow \{L_1, L_2\}$; $D_2 \leftarrow \{L_3, L_4\}$; goto Step 6
   end
   else
   begin
   choose an edge $f_2 = (u_3, u_4)$ with $(v_3, v_4) \in M'$;
   $\Delta \leftarrow \{L_3, L_4\}$; goto Step 7
   end
   end;
5. if $2 \leq |LF'| \leq 6$ then goto Step 8;

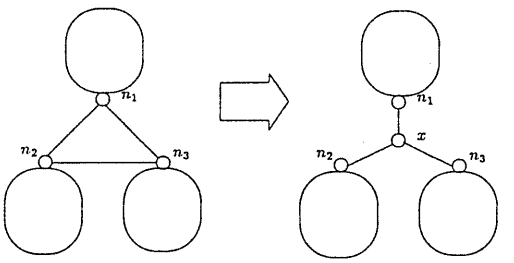

Figure 6: Replacing a cycle whose length is three in $F(G)$.

/* edge-interchange is going to be done */
6. Find a D-combination $\{\{L'_1, L'_2\}, \{L'_3, L'_4\}\}$, whose union is $D_1 \cup D_2$, such that two edges $f_1 = (u'_1, u'_2), f_2 = (u'_3, u'_4)$ with $u'_j \in L'_j$, $j = 1, 2, 3, 4$, satisfy that $G + \{f_1, f_2\}$ is a simple graph having a 5-component S with $L'_i \subseteq S$, $i = 1, 2, 3, 4$;
   $\Delta \leftarrow \{L'_3, L'_4\}$;
7. $LF' \leftarrow LF' - \Delta$; $R(G) \leftarrow R(G) - \Delta$; $E' \leftarrow E' \cup \{f_2\}$; goto Step 4;
8. Find a solution E'' for $G'' = (V, E \cup E')$ by using Propositions 6.2 through 6.7; $E' \leftarrow E' \cup E''$
   end.

We can prove the following theorem similarly to Theorem 5.1: the increase in time complexity is caused mainly because we have to handle a structural graph $F(G)$ that is a cactus when $\lambda(G) = 4$.

Theorem 6.1 *The algorithm GS4 correctly finds a solution E' to UW-5ECA(S,SA) for any given G with $\lambda(G) = 4$ in $O(|V|^2 + |E|)$ time.*

In Section 7, we will reduce the time-complexty of GS4 from $O(|V|^2 + |E|)$ to $O(|V| \log |V| + |E|)$.

7 Edge-interchange operation on a structural graph when $\lambda = 4$

When $\lambda = 4$, the procedure to be proposed in this section finds an edge set E'_s such that if it is not a solution for G then $F(G + E'_s)$ itself is a tree or is a cactus that can be equivalently transformed into tree. This transformation can be done in $O(|V| \log |V| + |E|)$ time including construction of $F(G + E'_s)$. By utilizing the resulting structural graph, the algorithm proposed in Section 4.3 finds a solution E''_s for $G + E'_s$ in $O(|V|)$ time. We can show $E'_s \cup E''_s$ is a solution for G. Hence we find a solution for G in $O(|V| \log ||V)$ time when $\lambda = 4$.

In this section we assume $\lambda = 4$. Let $F(G) = (N, A)$ be a structural graph of G. Note that $LF(G) = \{\rho^{-1}(n) | n$ is a leaf node$\}$. For any cutvertex $u \in G$ and each component S of $G - u$, $S \cup \{u\}$ is called a *u-block* of G.

We obtain the following proposition for a structural graph.

Proposition 7.1 *Let C be any cycle of $F(G)$ whose length is exactly three, and $V(C) = \{n_1, n_2, n_3\}$. A graph $F' = (N', A')$ remains to be a structural graph of G, where $N' = N \cup \{x\}$, $A' = A \cup \{(n_k, x) | 1 \leq k \leq 3\} - E(C)$ with each (n_k, x) having weight λ and x is an empty node with $x \notin N$ (see Fig. 6).*

(Proof) Omitted. □

By Proposition 7.1, we can obtain a structural graph F which has no cycle of length three in $O(|N|) = O(|V|)$ time, since $F(G)$ is a cactus. In the rest of this section, we assume $F(G)$ has no cycle of length three.

Now, we consider how to find a solution to UW-5ECA(S,SA) with $\lambda(G) = 4$ in $O(|V| \log |V| + |E|)$ time. The proposed procedure is based on the next proposition.

Proposition 7.2 *Let C be any cycle of length at least four in $F(G)$, n_1, n_2 be nonadjacent nodes in $V(C)$. Let N_j be the union of those n_i-blocks of $F(G)$ none of which contains $V(C) - \{n_i\}$. Then, for any $n'_j \in N_j$ for $j = 1, 2$, $\rho^{-1}(n'_1) \overline{\chi} \rho^{-1}(n'_2)$.*

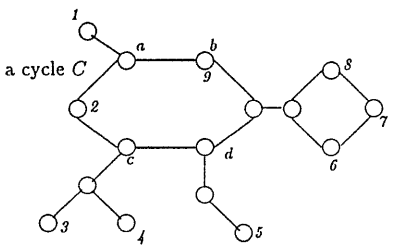

Figure 7: A situation of Proposition 7.3, where the numbers next to nodes are given by *NUM_F*.

(Proof) $F(G) - N_1 \cup N_2$ has exactly two connected components N_3 and N_4. Let $V_i' = \{v \in \rho^{-1}(n') | n' \in N_i\}$ for $i = 1, \ldots, 4$. G has two λ-cuts $K_1 = (V_1' \cup V_3', \overline{V_1' \cup V_3'})$ and $K_2 = (V_1' \cup V_4', \overline{V_1' \cup V_4'})$, crossing each other in G. By the proof of Proposition 4.2(see [13]), we have $(V_1', V_2'; G) = \emptyset$, showing that $\rho^{-1}(n_1') \overline{\chi} \rho^{-1}(n_2')$. □

We can obtain the following corollary from Proposition 7.2.

Corollary 7.1 *For any $n_1, n_2 \in N$, if $\rho^{-1}(n_1) \chi \rho^{-1}(n_2)$ then, for any cycle C of $F(G)$, there is an elementary path P between n_1 and n_2 in $F(G)$ contains at most one edge of C.*

We first show the following procedure *NUM_F* which assigns a number to each leaf node in $F(G)$.

Procedure *NUM_F*;
begin
1. Assingn colors to edges of $F(G)$, with the same color to all edges of each elementary cycle and different colors to separate cycles;
2. Choose any node n of $F(G)$;
3. Execute a depth first search, starting at n and assigning the number $\beta(n')$ to each leaf node n' in the order of first visit by the depth first search, in the following manner:
   in any node v is visited first by way of a cycle edge $e = (u, v)$ of color $C(e)$ then visit any other adjacent node w and the edge (v, w) of color different from $c(e)$, before visiting the other edge (v, u') of color $C(e)$.

end;

Clearly *NUM_F* runs in $O(|V|)$ times.

Proposition 7.3 *Let C be any cycle of $F(G)$ whose length is at least four, and N_1' and N_2' be the components in $F(G) - \{(a, b), (c, d)\}$, where $(a, b), (c, d) \in E(C)$ and the four vertices a, b, c and d are all distinct (see Fig. 7). Then there is at least one pair of leaves $n_1 \in N_1'$, $n_2 \in N_2'$ such that $(\beta(n_1) + \lfloor q/2 \rfloor) \equiv \beta(n_2) (\mathrm{mod} q)$ and $\rho^{-1}(n_1) \overline{\chi} \rho^{-1}(n_2)$, where q is the total number of leaves of $F(G)$.*

(Proof) Let N_i be the set of all leaves in N_i' for $i = 1, 2$. Assume $a, c \in N_1'$, $b, d \in N_2'$ and $|N_1| \leq |N_2|$ without loss of generality. We have three possible cases: (i) $N_1 = \{1, \ldots, p'\} \cup \{p, \ldots, q\}$ with $1 \leq p' < p \leq q$, (ii) $N_1 = \{p, \ldots, q\}$ with $1 \leq p \leq q$, or (iii) $N_1 = \{1, \ldots, p'\}$ with $1 \leq p' \leq q$. For any $n \in N_1$, let

$$\gamma(n) \equiv (\beta(n) + (q - p) + 1) (\mathrm{mod} q). \text{ for (i) and (ii);}$$

$$\gamma(n) = \beta(n).$$

Note that $\{\gamma(n) | n \in N_1\} = \{1, \ldots, |N_1|\}$.

Suppose $\rho^{-1}(n_1) \chi \rho^{-1}(n_2)$ for any pair $n_1 \in N_1$ and $n_2 \in N_2$ with $\gamma(n_2) = \gamma(n_1) + \lfloor q/2 \rfloor$. Clearly, $n_2 \in N_2$ since $|N_1| \leq |N_2|$. By Corollary 7.1, it suffies to consider the case under the length of C is four.

For any $x \in \{a, b, c, d\}$, let N^x be the set of leaf nodes in $N - B_x$, where B_x is the x-block containing $V(C) - \{x\}$. We may assume that N^a has a leaf node n with $\gamma(n) = 1$ without loss of generality. Let $M^x = \{\gamma(n) | n \in N^x\}$ for any $x \in \{a, b, c, d\}$. Then $M^a = \{i | 1 \leq i \leq |N^a|\}$, $M^c = \{i | |N^a| + 1 \leq i \leq |N^a| + |N^c|\}$ It follows from an assumption and Corollary 7.1 that $\{n | n' \in N^a, \gamma(n') + \lfloor q/2 \rfloor = \gamma(n)\} \subseteq N^b$ and $\{n | n' \in N^c, \gamma(n') + \lfloor q/2 \rfloor = \gamma(n)\} \subseteq N^d$. This means that $\max\{\gamma(n) | n \in N^b\} < \min\{\gamma(n) | n \in N^d\}$. On the other hand we have $\min\{\gamma(n) | n \in N^b\} > \max\{\gamma(n) | n \in N^d\}$ by the method assigning $\beta(n)$ and the definition of $\gamma(n)$, a cntradiction. □

Any pair of leaf nodes n_1, n_2 mentioned in Proposition 7.3 is called an admissible pair of $F(G)$.

We show the following procedure *EDGE_FIND_E* which finds a edge set E_s' which is either a solution or a edge set by whose addition, one of structural graphs of the resulting graph is a tree. where the procedure is based on Algorithm Aug_1 in [11].

Procedure *EDGE_FIND_E*
/* Input: a simple graph $G = (V, E)$ which $\lambda(G)$ is even */
/* Output: a subset E_s' of edges */
begin
1. Construct a structural graph $F(G)$ of G; $V' \leftarrow$ {all leaf nodes of $F(G)$}
2. Assign the number $\beta(n')$ to each leaf node n' in $F(G)$ by *NUM_F*;
3. $E_s' = \{(v_1, v_2) | \{n_1, n_2\}$ is an addmissible pair of $F(G)$, $1 \leq \beta(n_1) \leq \lceil q/2 \rceil$, $v_i \in \rho^{-1}(n_0)$ for $i = 1.2$, and $(v_1, v_2) \notin A\}$;

end

Proposition 7.4 *If any cycle of $F(G)$ has length at least four, then $F(G + E_s')$ is a tree. If has any cycle of length three then it is changed to an equivalent tree by repeated application of Proposition 7.1.*

(Proof) Omitted.

□

We can easily prove the following proposition.

Proposition 7.5 *Let $(S, \overline{S})$ be any λ-cut in G, and $(N_1, \overline{N_1}; F(G)) = \{f_1, f_2\}$ be the cut representing $(S, \overline{S})$. Assume that N_1 contains no more leaf nodes than $\overline{N_1}$. Then for any leaf node $n_1 \in N_1$, $\overline{N_1}$ has a leaf node n_2 with $\beta(n_2) \equiv \beta(n_1) + \lfloor q/2 \rfloor (\mathrm{mod} q)$.*

Proposition 7.6 *Let X be a leaf of G with $\beta(\rho(X)) = \lceil q/2 \rceil$. Let Y_1 (Y_2, respectively) be the leaf of G such that $\beta(\rho(Y_1)) = 1$ ($\beta(\rho(Y_2)) = q$). If, for some $(u, v) \in E_s'$, a $(\lambda + 1)$-component S containing $\{u, v\}$ of $G + E_s'$ is a leaf then q is odd, $x \in X$, $y \in Y_i$ and $X \chi Y_j$, where $x \in \{u, v\}$, and $\{i, j\} = \{1, 2\}$.*

(Proof) Assume that the $(\lambda + 1)$-component S is a leaf. Clearly, $(S, \overline{S}; G) = (S, \overline{S}; G + E_s')$. Let $(N_1, \overline{N_1}; F(G))$ be either a bridge or a cutpair of $F(G)$ representing $(S, \overline{S}; G)$. From Proposition 7.5, there must exist a pair of leaf nodes n_1, n_2 such that $n_1 \in N_1$ and $\beta(n_2) \equiv \beta(n_1) + \lfloor q/2 \rfloor (\mathrm{mod} q)$. For any such pair, we have $\rho^{-1}(n_1) \chi \rho^{-1}(n_2)$ in G, otherwise $(S, \overline{S}; G)$ is no longer a λ-cut of $G + E_s'$. If either q is even or $\beta(n_1) \neq \lceil q/2 \rceil$ then $\rho^{-1}(n_1)$ is a $(\lambda + 1)$-component in $G + E_s'$ since E_s' has no edge included upon a vertex in $\rho^{-1}(n_1)$, contradicts the fact that S is a leaf since $\rho^{-1}(n_1) \subset S$. □

Next corollary is contraposition of Proposition 7.6.

Corollary 7.2 *If either (q is even) or (q is odd and $u, v \notin X$) then the $(\lambda + 1)$-component S of $G + E_s'$ is not a leaf.*

We obtain the following proposition from Proposition 7.3.

Proposition 7.7 *If $|LF(G + E_s')| \geq 7$ then $E' = E_s' \cup E_s''$ is a solution for G, where E_s'' is a solution for $G + E_s'$.*

(Proof) Clearly, $\lambda(G + E') = \lambda + 1$. We only show that $|E'|$ is minimum. Let q' be the number of leaves of $G + E_s'$. By Proposition 7.6 and Corollary 7.2, if the $(\lambda + 1)$-component S containing $\{u, v\}$ is not a leaf in G for any $(u, v) \in E_s'$, then $q' = q - 2|E_s'|$; otherwise$q' = q - (2|E_s'| - 1)$. By Theorem 6.1, $|E_s''| = \lceil q'/2 \rceil$. Hence $|E'| = \lceil q/2 \rceil$ in each of two cases. □

If $|LF(G)| \geq 7$ and $|LF(G + E_s')| \leq 6$ then we select $\lceil (7 - |LF(G + E_s')|)/2 \rceil$ edges in E_s', delete the edges from E_s' and denote the resulting edge set as E_s''. Clearly, $|LF(G + E_s'')|$ is seven or eight. We find a solution for $G + E_s''$ by using Algorithm GS_4 in Section 6.3. This is done in $O(|V|)$ time. Hence, if $|LF(G)| \geq 7$ and $|LF(G + E_s')| \leq 6$ then we find a solution for G in $O(|V| \log |V| + |E|)$ time. We obtain the following proposition which can be shown similarly to the proof of Proposition 7.7.

Proposition 7.8 *If $|LF(G)| \geq 7$ and $|LF(G + E_s')| \leq 6$ then $E_s'' \cup E_s'''$ is a solution for G, where E_s''' is a solution for $G + E_s'$.*

IF $|LF(G)| \leq 6$ then we find a solution for G in $O(|V|\log|V|+|E|)$ time by using Algorithm *GS4* since Algorithm *GS4* does not have Step 5 in this case.

From the above discussion, we can obtain the following theorem.

Theorem 7.1 *There is an algorithm that correctly finds a solution E' to UW-5ECA(S,SA) for any given G with $\lambda(G) = 4$ in $O(|V|\log|V|+|E|)$ time.*

8 UW-$(\lambda+1)$ECA(*,SA) with $\lambda = 3, 4$

We will show that UW-$(\lambda+1)$ECA(*,SA) with $\lambda = 3$ or 4 can be solved by the algorithms proposed in Section 6 and 7, slightly modifying them for handling G that may be a multiple graph.

Proposition 8.1 *Suppose $L_1 \chi L_2$ for $L_1, L_2 \in LF(G)$. Then the following (1) or (2) holds:*

(1) if $|\Gamma_G(\lambda+1)| \geq 3$ then $|L_1||L_2| \leq \lfloor \lambda/2 \rfloor$;

(2) if $|\Gamma_G(\lambda+1)| = 2$ then $|L_1||L_2| \leq \lambda$.

(Proof) First, we consider the case (1). Suppose $|L_1||L_2| \geq \lfloor \lambda/2 \rfloor + 1$. If $\lambda = 2m$ ($= 2m-1$, respectively) then $|L_1||L_2| \geq m+1$ ($\geq m$). Since $L_1 \chi L_2$, the definition of an adjacent pair implies that G has $|L_1||L_2|$ edges between L_1 and L_2. We have $d_G(L_i) = \lambda$ $(i = 1, 2)$, and

$$|(L_1 \cup L_2, V - L_1 \cup L_2)| = 2\lambda - 2|L_1||L_2| \leq \lambda - 1,$$

which contradicts the fact that G is λ-edge-connected.

Next, we consider the case (2). If we assume $|L_1||L_2| > \lambda$ then $d_G(L_i) > \lambda$ $(i = 1, 2)$, contradicting the fact that L_1 and L_2 are leaves in G. □

Next, we consider a solution to UW-$(\lambda+1)$ECA(*,SA) with $\lambda = 3, 4$. Clealry we obtain the following poposition.

Proposition 8.2 *If there is a solution for G then there does not exist a leaf $L \in LF(G)$ with $L = \{u\}$ and $\{(u, u')|u' \in V - \{u\}\} \subseteq E$.*

Proposition 8.3 *Suppose $\lambda = 3, 4$, $|LF(G)| \geq 3$ and $L_1 \overline{\chi} L_2$ for a pair of leaves $L_1, L_2 \in LF(G)$. Let $G' = G + \{(u_1, u_2)\}$ with $u_i \in L_i$, $i = 1, 2$ and suppose G' has a leaf (denoted by L') with $L_1 \cup L_2 \subseteq L'$. Then the following (1) or (2) holds:*

(1) in case $|\Gamma_G(\lambda+1)| \geq 4$ and $|LF(G)| \geq 3$, we have $L' \overline{\chi}_{G'} L_3$ for any $L_3 \in LF(G) - \{L_1, L_2\}$;

(2) in case $|\Gamma_G(\lambda+1)| = |LF(G)| = 3$, we have $L' \chi_{G'} L_3$ if and only if $\lambda = 4$, $L_3 = \{u\}$ and $X = \{(u, u')|u' \in V - \{u\}\} \subseteq E$.

(Proof) We have $|L'| \geq 2$ since $L_1 \cup L_2 \subseteq L'$. First, we consider (1). $|L'||L_3| \geq 2$, so if $\lambda = 3$ then, by Proposition 8.1(1), clearly $L' \overline{\chi}_{G'} L_3$. We assume $\lambda = 4$. Suppose $L' \chi_{G'} L_3$. Since $|\Gamma_G(\lambda+1)| \geq 4$, G' has more than three vertices. Hence, by Proposition 8.1(1), since $|L'| \geq 2$ and $L' \chi_{G'} L_3$, we have $|L'| = 2$ and $|L_3| = 1$. This means $|L_i| = 1$ $(i = 1, 2)$. Let $Z = L_1 \cup L_2$ then $(Z, \overline{Z}; G)$ is a 4-cut since L' is a leaf. $d_G(L_1) = d_G(L_2) = 4$, $|(Z, \overline{Z}; G)| = 4$ and $|L_i| = 1$ $(i = 1, 2)$, so $(u_1, u_2) \in E$, with $u_i \in L_i$ for $i = 1, 2$, exists and $L_1 \chi L_2$, which contradicts the supposition with $L_1 \overline{\chi} L_2$.

Next, we consider (2). Then $\lambda = 4$ since if $|\Gamma_G(\lambda+1)|$ is odd then it does not hold that $d_G(X)$ is odd for any $X \in \Gamma_G(\lambda+1)$. We will show only necessity since clearly sufficiency holds. Suppse $|\Gamma_G(\lambda+1)| = |LF(G)| = 3$ and $L' \chi_{G'} L_3$. Then $E = \{(u_i, u_j), (u'_i, u'_j)|1 \leq i < j \leq 3\}$ in G, and $\Gamma_{G'}(\lambda+1) = \{L', L_3\}$, where $u_i, u'_i \in L_i$ and $u_j, u'_j \in L_j$. By Proposition 8.1(2), $|L'||L_3| \leq 4$ holds. Hence $|L'| \geq 2$ and $|L_3| \leq 2$. If we assume $|L_3| = 2$ then $|L'| = 2$, $d_G(L_1) = d_G(L_2) = 4$, $|(Z, \overline{Z}; G)| = 4$ for $Z = L_1 \cup L_2$, and $|L_i| = 1$ $(i = 1, 2)$. Hence there is $(u_1, u_2) \in E$ with $u_i \in L_i$ for $i = 1, 2$ and $L_1 \chi L_2$, contradicting a supposition. Hence $L_3 = \{u\}$ and $\{(u, u')|u' \in L'\} = \{(u, u')|u' \in V - \{u\}\} \subseteq E$ since $LF(G') = \{L', L_3\}$ and $L' \chi_{G'} L_3$. □

If G is a multiple graph without leaves L mentioned in Proposition 8.2 then, by using Proposition 8.3, the discussion similar to Sections 3, 5 and 6 when G does not has the leaf L of Proposition 8.2 is possible. (It should be mentioned that the proof of Proposition 5.2 requires some modification in handling UW-$(\lambda+1)$ECA(*,SA): the result is the same.) Hence we obtain the following theorem.

Theorem 8.1 *The algorithm GS3 (GS4, respectively) can be used in finding a solution to UW-$(\lambda+1)$ECA(*,SA) with $\lambda(G) = 3$ (with $\lambda(G) = 4$) in $O(|V|\log|V|+|E|)$ time.*

The paper has proposed an $O(|V|\log|V|+|E|)$ algorithm for solving UW-4ECA(*,SA) with $\lambda(G) = 3$ (UW-5ECA(S,SA) and UW-5ECA(*,SA) $\lambda(G) = 4$). The paper is a first step toward UW-kECA(S,SA) and UW-kECA(*,SA), which are left for future research.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. On finding lowest common ancestors of trees. *SIAM J. Comput.*, 5:115–132, 1976.

[3] K. P. Eswaran and R. E. Tarjan. Augmentation problems. *SIAM J. Comput*, 5:653–655. 1976.

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

[5] A. Frank. Augmenting graphs to meet edge connectivity requirements. *SIAM J. Discrete Mathematics*, 5(1):25–53, 1992.

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

[7] H. N. Gabow. Applications of a poset representation to edge connectivity and graph rigidity. In *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1991.

[8] T. C. Hu. *Integer Programming and Network Flows.* Addison-Wesley, Reading, Mass, 1969.

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

[10] H. Nagamochi and T. Ibaraki. A linear time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7:583–596, 1992.

[11] D. Naor, D. Gusfield, and C. Martel. A fast algorithm for optimally increasing the edge connectivity. In *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 698–707, 1990.

[12] D. Takafuji, S. Taoka, and T. Watanabe. Simplicity-preserving augmentation to 4-edge-connect a graph. *IPSJ SIG Notes*, AL-33-5:33–40, May 1993.

[13] S. Taoka, D. Takafuji, and T. Watanabe. Simplicity-preserving augmentation of the edge-connectivity of a graph. *Tech. Rep. of IEICE of Japan*, COMP93-73:49–56, January 1994.

[14] S. Taoka and T. Watanabe. Minimum augmentation to k-edge-connect specified vertices of a graph. In *Lecture Notes in Computer Science*, pages 217–225. Springer-Verlag, Berlin, 1994. (Proc. 5th International Syposium on Algorithms and Computation(ISAAC'94)).

[15] S. Taoka and T. Watanabe. Smallest augmentation to k-edge-connect all specified vertices in a graph. *IPSJ SIG Notes*, AL-38-3:17–24, March 1994.

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

[17] T. Watanabe. An efficient way for edge-connectivity augmentation. Tec. Rep. ACT-76-UILU-ENG-87-2221, Coordinated Science Lab., University of Illinois at Urbana, Urbana, IL 61801, April 1987. Also presented at Eighteenth Southeastern International Conference on Combinatorics, Graph Theory, Computing, No.15, Boca Raton, FL, U.S.A., February 1987.

[18] T. Watanabe. A simple improvement on edge-connectivity augmentation. Tech. Rep. CAS87-203, pp. 43–48, IEICE of Japan, 1987.

[19] T. Watanabe and A. Nakamura. Edge-connectivity augmentation problems. *J. Comput. System Sci.*, 35(1):96–144, 1987.

[20] T. Watanbe and M. Yamakado. A linear time algorithm for smallest augmentation to 3-edge-connect a graph. *Trans. IEICE of Japan*, E76-A(4):518–531, 1993.