Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–514
Volume 1, Number 1, pages 1–8, January 2012

A Simplified Algorithm for Augmenting Edge-Connectivity by One with Bipartition Constraints

Tadachika Oki
Graduate School of Engineering
Hiroshima University
1-4-1 Kagamiyama,
Higashi-Hiroshima,
739-8527 Japan

Satoshi Taoka
Graduate School of Engineering
Hiroshima University
1-4-1 Kagamiyama,
Higashi-Hiroshima,
739-8527 Japan

Toshimasa Watanabe
Graduate School of Engineering
Hiroshima University
1-4-1 Kagamiyama,
Higashi-Hiroshima,
739-8527 Japan

Email: oki@infonets.hiroshima-u.ac.jp Email: taoka@infonets.hiroshima-u.ac.jp Email: watanabe@infonets.hiroshima-u.ac.jp

***Abstract*—The k-edge-connectivity augmentation problem with bipartition constraints (kECABP, for short) is defined by "Given a multigraph $G = (V, E)$ and a bipartition $\pi = \{V_B, V_W\}$ of V with $V_B \cap V_W = \emptyset$, find an edge set E_f of minimum size, consisting of edges that connect V_B and V_W, such that $G_f = (V, E \cup E_f)$ is k-edge-connected, where a multigraph means a graph, with unweighted edges, such that multiple edges may exist." In this paper, we give a simplified algorithm for finding an optimal solution to $(\sigma + 1)$ECABP in $O(|V||E| + |V|^2 \log |V|)$ time when G is σ-edge-connected ($\sigma > 0$), and show that the problem can be solved in linear time when $1 \le \sigma \le 2$. The time complexity of the proposed algorithm is equal to that of an existing algorithm of Oki et al. (2012) (to appear).**

I. Introduction

[Background] The *k-edge-connectivity augmentation problem* (kECA, for short) is defined by "Given a multigraph $G = (V, E)$, find an edge set E_f of minimum cardinality such that $G_f = (V, E \cup E_f)$ is k-edge-connected, where a multigraph means a graph, with unweighted edges, such that multiple edges may exist." We often denote G_f as $G + E_f$, and E_f is called an optimal *solution* to the problem. There are several applications to construction of a fault-tolerant network, and so on. It is called the *k-edge-connectivity augmentation problem with bipartition constraints* (kECABP, for short) when a bipartition $\pi = \{V_B, V_W\}$ of V with $V_B \cap V_W = \emptyset$ is additionally given and we require that E_f consists of edges connecting between V_B and V_W (see Fig. 1).

A bipartite graph is a graph (V, E) such that V is partitioned into two sets V^B and V^W with $V^B \cap V^W = \emptyset$, and any edge $(u, v) \in E$ satisfies a condition ($u \in V^B$ and $v \in V^W$) or ($u \in V^W$ and $v \in V^B$): such a graph is often denoted as $G = (V^B \cup V^W, E)$. If G is bipartite and we set $V_B = V^B$ and $V_W = V^W$ in kECABP then $G_f = G + E_f$ is bipartite.

This problem, denoted as B-kECABP, is a typical subproblem of kECABP, where "B-" means that G is a bipartite graph. There are several applications to security of statistical data stored in a cross tabulated table [5], and so on.

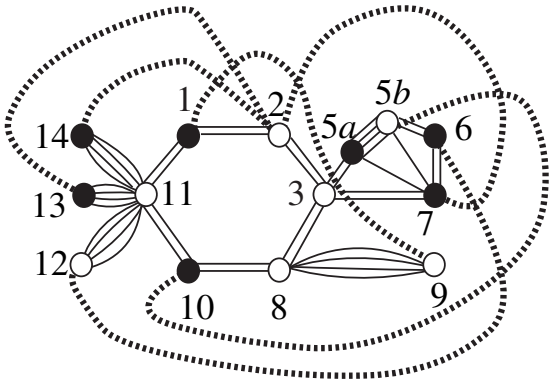

Fig. 1. A graph G with $\lambda(G) = 4$, where a closed circle (an open circle, respectively) represents a vertex which belongs to V_B (V_W). The set of dashed lines is an optimal solution $E_f = \{(1, 9), (2, 7), (2, 13), (2, 14), (5b, 10), (6, 12)\}$ to 5ECABP.

[Existing Results] Many algorithms for kECA have been given. [3] gave a linear time algorithm for 2ECA, and [16], [4], [9] gave polynomial time algorithms for kECA.

[5] gave a linear time algorithm for B-2ECABP, and an $O(\log |V|)$ parallel time algorithm on an EREW PRAM with a linear number of processors. A general problem kECAMP is similarly defined, where r-partition $\pi_M = \{V_1, \ldots, V_r\}$ ($r \ge 2$) of V is given and E_f consists of edges connecting between V_i and V_j ($1 \le i < j \le r$). Several algorithms for kECAMP have been given: [1] gave an $O(|V|(|E|+|V| \log |V|) \log |V|)$ time algorithm. Note that, in [1], a given multigraph is handled as an edge-weighted simple one such that, for any pair of vertices u and v, a simple edge (u, v) with a weight $w((u, v)) = x$ means that there are x multiple edges between u and v.

Let M-kECAMP denote kECAMP in which G is an r-partite graph, where "M-" means that G is a multipartite graph. [2] gave a linear time algorithm for 2ECAMP, and an $O(\log |V|)$ parallel time algorithm on an EREW PRAM with a linear number of processors. [12] gave an $O(|V||E|+|V|^2 \log |V|)$ time

algorithm for $(\sigma+1)$ECABP.

[Our Contributions] Now we describe our contributions of the paper as follows.

Our main contribution of the paper is to give a simplified algorithm S-Sol_$(\sigma+1)$ECABP for finding an optimal solution to $(\sigma+1)$ECABP in linear time when G is σ-edge-connected and a structural graph $F(G)$ of G is given. Note that a structural graph $F(G)$ represents all minimum cuts of G.

The time complexity of S-Sol_$(\sigma+1)$ECABP is $O(|V||E|+|V|^2\log|V|)$ because $F(G)$ can be constructed in $O(|V||E|+|V|^2\log|V|)$ time [10]. It follows that the problem can be solved in linear time when $1 \le \sigma \le 2$ because $F(G)$ can be obtained in $O(|V|+|E|)$ time [8], [13].

The main theorem is as follows and its proof is given in § IV-D.

Theorem 1.1 Algorithm S-Sol_$(\sigma+1)$ECABP finds an optimal solution to $(\sigma+1)$ECABP in $O(|V||E|+|V|^2\log|V|)$ time. Moreover, it runs in $O(|V|+|E|)$ time when $1 \le \sigma \le 2$.

[The Structure of the Paper] The paper is organized as follows. Section II provides some definitions and notations. Section III shows a lower bound on the size of feasible solutions to $(\sigma+1)$ECABP. Section IV gives formal description of S-Sol_$(\sigma+1)$ECABP, and in Section IV-C, its correctness is provided, and in Section IV-D, its time complexity is provided. The concluding remarks are given in Section V.

II. Definitions

In this section, we explain basic terminologies of graphs and structural graphs. Also mentioned is handling of a multigraph: unweighted multigraphs versus edge-weighted simple graphs.

[Basic Terminologies of Graphs] An *undirected graph* is denoted as $G=(V(G),E(G))$, where $V(G)$ and $E(G)$ are often denoted as V and E, respectively. In this paper, only graphs without loops are handled, and the term "a graph" means an undirected multigraph unless otherwise stated. An edge that is incident to two vertices u,v in G is denoted as (u,v). For two graphs $G=(V(G),E(G))$ and $H=(V(H),E(H))$, H is a *subgraph* of G if $V(G)\subseteq V(H)$ and $E(G)\subseteq E(H)$.

Remark 2.1 It should be noted that conceptually a multigraph is used, while the corresponding edge-weighted simple graph is adopted in actual handling of graphs in this paper.

For a set $X\subseteq V$ of G, let $G[X]$ denote the subgraph having X as its vertex set and $\{(u,v)\in E \mid u,v\in X\}$ as its edge set.

For two disjoint sets X, $X'\subset V$, we denote $(X,X';G)=\{(u,v)\in E|u\in X \text{ and } v\in X'\}$, where it is often denoted as (X,X') if G is clear from the context. The *degree* of X (in G) is defined by $d_G(X)=|(X,V-X;G)|$. If $X=\{v\}$ then $d_G(\{v\})$ is the total number of edges incident to v and is called the degree of v (in G): $d_G(\{v\})$ is denoted as $d_G(v)$. For a set $X\subseteq V$, the set $(X,V-X;G)$ is called a k-cut if $|(X,V-X)|=k$.

For an edge set E_f consisting of edges with endvertices in V, let $G+E_f$ denote the graph $(V,E\cup E_f)$. For a given bipartition $\pi=\{V_B,V_W\}$ of V with $V_B\cap V_W=\emptyset$, E_f is said to be *legal* (with respect to π) if E_f consists of edges connecting between different members V_B and V_W of π. The term "with respect to π" is omitted if π is fixed.

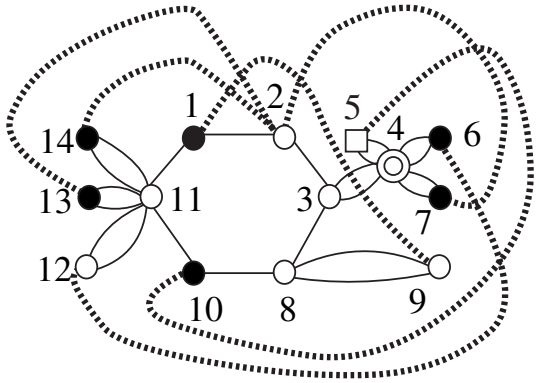

Fig. 2. A structural graph $F(G)$, where a closed circle (an open circle and a square, respectively) represents a vertex which belongs to $BF(G)$ ($WF(G)$ and $HF(G)$). The vertex 4 (double circled) is inserted in constructing $F(G)$. The set of dashed lines denotes an optimal solution $E'=\{(1,9),(2,7),(2,13),(2,14),(5,10),(6,12)\}$ to a structural graph $F(G)$ of G in Fig. 1 and $\lambda(F(G)+E')=\lambda(F(G))+1$.

A *trail* (or a (v_0,v_r)-trail) is a sequence of distinct edges $(v_0,v_1),(v_1,v_2),\ldots,(v_{r-1},v_r)$ in which there may appear the same endvertices, and each of v_0 and v_r is called a *terminal vertex* of the trail. A trail is called a *closed trail* if $r\ge 1$ and $v_0=v_r$. A closed trail is called an *Eulerian closed trail* of G if all edges of G are included. A trail is called a *path* (or a (v_0,v_r)-path) if all vertices $v_0,v_1,\ldots,v_r$ are distinct. A *cycle* consists of a path with $r\ge 1$ and an edge (v_r,v_0).

For two vertices $u,v\in V$, let $\lambda(u,v;G)$ denote the maximum number of edge-disjoint (u,v)-paths between u and v in G. For a subset $\Gamma\subseteq V$, the *edge-connectivity of* Γ in G is defined by $\lambda(\Gamma;G)=\min_{u,v\in\Gamma}\lambda(u,v;G)$. The *edge-connectivity* of G is defined as $\lambda(V;G)$, and is denoted as $\lambda(G)$. G is *k-edge-connected* if $\lambda(G)\ge k$ for a nonnegative integer k. In particular, G is *connected* if G is 1-edge-connected. For a set $S\subseteq V$, S is a *k-edge-connected component* (k-component, for short) of G if S is a maximal vertex set such that $\lambda(u,v;G)\ge k$ holds for any pair of vertices $u,v\in S$. For a k-component S, S is a *leaf k-component* if $d_G(S)=\lambda(G)$. Note that distinct k-components are pairwise disjoint. For a k-cut $(X,V-X;G)$, $(X,V-X;G)$ is a minimum cut if $k=\lambda(G)$. A vertex v is called a *cutvertex* of G if the number of 1-components of $G-v$ is greater than that of G, where $G-v$ denotes $G[V-\{v\}]$. In this paper, let $\sigma=\lambda(G)$.

A *tree* is a connected graph which does not contain any cycle as its subgraph. A *cactus* is an undirected connected graph in which any pair of cycles shares at most one vertex: any vertex v with $1\le d_G(v)\le 2$ is called a *leaf* and any shared vertex is a cutvertex.

[Structural Graphs] A structural graph $F(G)=(V(F(G)),E(F(G)))$ (see [10], [6] for example) of G with $\lambda(G)=\sigma$ is a representation of all minimum cuts of G (see Fig. 2). $F(G)$ is an edge-weighted cactus of $O(|V|)$ vertices and edges such that each tree edge (a bridge in $F(G)$) has weight $\lambda(G)$ and each cycle edge (an edge included in a cycle) has weight $\lambda(G)/2$.

Particularly if $\lambda(G)$ is odd then $F(G)$ is an edge-weighted tree.

Any minimum cut (σ-cut) of G corresponds to either a tree edge or a pair of two cycle edges in the same cycle of $F(G)$, and vice versa. Each $(\sigma+1)$-component of G is represented as a vertex of $F(G)$. In particular each leaf $(\sigma+1)$-component of G corresponds to a leaf of $F(G)$, and vice versa. Several vertices, called *empty vertices*, are added in order to form a cactus.

Let ρ: $V(G) \to V(F(G))$ denote such a mapping. We use the following notations: $\rho(X) = \{\rho(v) \mid v \in X\}$ for $X \subseteq V$ and $\rho^{-1}(Y) = \{v \in V \mid \rho(v) \in Y\}$ for $Y \subseteq V(F(G))$. If $Y = \{v\}$ then $\rho^{-1}(Y)$ is written as $\rho^{-1}(v)$, instead of $\rho^{-1}(\{v\})$.

We can regard $F(G)$ as a *modified cactus* which is defined as follows: if $F(G)$ has any bridge of weight $\lambda(G)$ then we replace such a bridge by a pair of multiple edges, assigning each edge weight $\lambda(G)/2$ even if $\lambda(G)$ is odd, and regard such a pair of multiple edges as a simple cycle of length two. We assume that $F(G)$ is a modified cactus in this paper unless otherwise stated. $F(G)$ can be obtained in $O(|V||E| + |V|^2 \log |V|)$ time as a modified cactus [10].

Note that we can handle this modified cactus $F(G)$ as a structural graph of G and $\lambda(F(G)) = 2$. Even if $\lambda(G)$ is odd, finding an edge set E_f such that $\lambda(G + E_f) = \lambda(G) + 1$ can be solved by finding an edge set E' such that $\lambda(F(G) + E') = 3$ for a modified cactus $F(G)$. This is because there is a bijection $\xi : E' \to E_f$ such that $\xi((u, v)) = (n_u, n_v)$, with $\rho(n_u) = u$ and $\rho(n_v) = v$.

A vertex $y \in V(F(G))$ with $\rho^{-1}(y) = \emptyset$ is an empty vertex. Let $\varepsilon(G) \subseteq V(F(G))$ denote the set of all empty vertices of $F(G)$. See [10] for efficiently constructing $F(G)$ from G.

Let us call each vertex of V_B (of V_W, respectively) a black vertex (a white one) of G. Given a structural graph $F(G)$ of a graph $G = (V, E)$ with a bipartition $\pi = \{V_B, V_W\}$, we classify vertices $v \in V(F(G)) - \varepsilon(G)$ into three types as follows:

(i) $\rho^{-1}(v) \subseteq V_B$ ($\rho^{-1}(v) \subseteq V_W$, respectively) ($v$ is called a *black vertex* (*a white one*) of $F(G)$);
(ii) $\rho^{-1}(v) \cap V_B \neq \emptyset$ and $\rho^{-1}(v) \cap V_W \neq \emptyset$ (v is called a *hybrid one* of $F(G)$).

The set of *black leaves* (*white ones*, or *hybrid ones*, respectively) of $F(G)$ is denoted as $BF(G)$ ($WF(G)$, or $HF(G)$). Let $LF(G) = BF(G) \cup WF(G) \cup HF(G)$. In this paper, without loss of generality, we assume that $V_B \neq \emptyset$, $V_W \neq \emptyset$ and $|BF(G)| \geq |WF(G)|$.

We say that $F(G)$ is *B-dominant* [5] if $|BF(G)| > |WF(G)| + |HF(G)|$ holds.

In figures for $F(G)$ of this paper, we represent a hybrid vertex as a square, and a black one or a white one as a closed circle or an open one (see Fig. 2).

III. A Lower Bound on a Feasible Solution to $(\sigma+1)$ECABP

Since $(\sigma+1)$ECABP is a subproblem of kECAMP, we obtain the following proposition by setting $k = \sigma + 1$ for a lower bound shown in [1] on kECAMP.

Assume that $F(G)$ has a sequence of t pairs ($t \geq 1$) of multiple edges (consisting of a simple cycle of length two) from a vertex u_0 to a vertex u_t. Then let us shrink all vertices $u_0, \ldots, u_{t-1}$ to u_t and remove any resulting self-loop, and then rename u_t as u_0. If u_0 is a black or a white or a hybrid vertex then so is the renamed vertex u_0. We call this operation 2-*cycle-pruning* (from u_0 to u_t). Fig. 3 shows schematic explanation of 2-cycle-pruning when u_t is included in a cycle of length at least three. If $x = y$ in Fig. 3 then 2-cycle-pruning can be repeated.

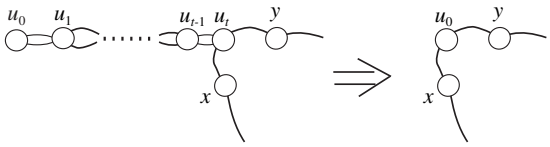

Fig. 3. Schematic explanation of 2-cycle-pruning from a leaf u_0 to a vertex u_t ($t \geq 1$) of $F(G)$ in a cycle of length at least three of $F(G)$.

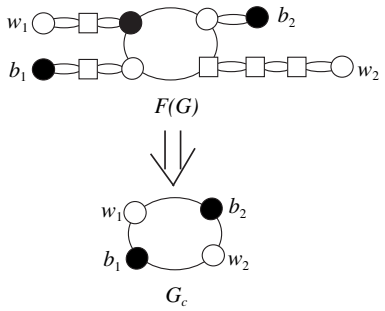

Fig. 4. An example of constructing G_c from a given $F(G)$ (by repeating 2-cycle-pruning).

Proposition 3.1 [12] Let G_c be any graph constructed from $F(G)$ by repeating 2-cycle-pruning as many times as possible. A lower bound $\mathcal{L}$ on the number of edges required in augmenting edge-connectivity of G by one bipartition is given as follows:

(i) If $|LF(G)| = 4$ and G_c is a simple cycle of length four such that two black leaves and two other ones (white one or hybrid one) appear alternately then $\mathcal{L} = 3$;
(ii) Otherwise,
$\mathcal{L} = \max\{|BF(G)|, |WF(G)|, \lceil |LF(G)|/2 \rceil\}$.

In the next section, we show that **Algorithm** S-Sol_($\sigma+1$)ECABP finds an edge set whose size is equal to the lower bound of Proposition 3.1, giving us an optimal solution E_f.

IV. A Simplified Algorithm for $(\sigma+1)$ECABP

In this section, we give a simplified algorithm S-Sol_($\sigma+1$)ECABP for $(\sigma+1)$ECABP.

A. Ideas for the Algorithm

Now we show ideas for designing the algorithm and the main procedure.

[A Linear Ordering on a Structural Graph] Now we explain a *linear ordering* on vertices of a modified cactus $F(G)$. This is introduced in [11] for finding an optimal solution to 3ECA for $F(G)$ efficiently, and is obtained as follows.

First all simple cycles in the cactus are assigned distinct colors. Note that this "color" is different from a color "black" or "white" used to present bipartition constraints. This coloring can be done in $O(|V|+|E|)$ time based on a depth-first search.

Next another depth-first search starts at an arbitrary vertex according to the following manner: if any vertex u is visited for the first time via an edge included in some simple cycle (for example, its color is red) then, before traversing another edge which is in the red cycle and incident to u, the other edges incident to u are traversed.

This search assigns a linear ordering (denoted as $\hat{\beta}(v)$) to each vertex v of $V(F(G))$ from 1 to $|V(F(G))|$, and traversing vertices v in the order of $\hat{\beta}(v)$ from 1 to $|V(F(G))|$ makes an Eulerian closed trail $ET(F(G))$ of $F(G)$. A vertex $v \in V(F(G))$ appears more than once in $ET(F(G))$ if and only if v is a cutvertex of $F(G)$.

Suppose that we fix a linear ordering $\hat{\beta}(v)$ for $v \in V(F(G))$. By traversing $ET(F(G))$ in the order of $\hat{\beta}(v)$ for $v \in V(F(G))$ from 1 to $|V(F(G))|$ also assigns another linear ordering $\beta(v)$ for leaves $v \in LF(G)$, that is, all leaves of $F(G)$ are numbered according to $\hat{\beta}$ by skipping cutvertices. Let us denote $LF(G) = \{v_1, \ldots, v_{|LF(G)|}\}$ with indices denoting this ordering β. Put $t = \lceil |LF(G)|/2 \rceil$. Let E_F be an edge set defined as follows:

$$E_F = \begin{cases} \{(v_i, v_{i+t}) \mid i = 1, \ldots, t\} & \text{if } |LF(G)| \text{ is even} \\ \{(v_i, v_{i+t}) \mid i = 1, \ldots, t-1\} \cup \{(v_t, v_1)\} & \text{if } |LF(G)| \text{ is odd} \end{cases}$$

Clearly $|E_F| = \lceil |LF(G)|/2 \rceil$, and it follows from the result of [11] that E_F is an optimal solution to 3ECA for a modified cactus $F(G)$.

[An Optimal Solution and a Strctural Graph] We can focus on the case where $F(G)$ is a modified cactus with $\lambda(F(G)) = 2$. From properties of a structural graph, it is enough to solve 3ECABP for $F(G)$, instead of $(\sigma+1)$ECABP for G (see [11], [7], for example). We call an optimal solution E' to 3ECABP for $F(G)$ simply as "an optimal solution E' to $F(G)$". Note that $|LF(G + \xi(E'))| = 0$.

It is easy to see that any optimal solution E' to $F(G)$ requires the following (i) and (ii).

(i) Edges $(u, v) \in E'$ connect as many leaves as possible (in order to efficiently augment the edge-connectivity of G by one);

(ii) n_u or n_v, respectively, should be a black vertex in $\rho_G^{-1}(u)$ or a white one in $\rho_G^{-1}(v)$ (in order to keep bipartition constraints).

Algorithm S-Sol_$(\sigma+1)$ECABP and **Procedure** FindEdgesBP are outlined as follows.

[Algorithm S-Sol_$(\sigma+1)$**ECABP]** We outline how to find an optimal solution E' to $F(G)$.

First, in order to narrow the gap between the number of black leaves and that of white ones, each hybrid leaf is appropriately regarded as a black leaf or a white one, since any hybrid leaf can be treated as a black or white one.

Next we find an edge set E'_p by **Procedure** FindEdgesBP, where the edge set E'_p consists of edges connecting different members of a given bipartition. As input to the procedure we choose a leaf set $L_p = W_p \cup B_p$, where W_p consists of all white leaves and B_p is a set of black ones arbitrarily selected so that the number of black ones may be equal to that of white ones. Note that $|L_p|$ is even. If $F(G)$ is B-dominant then, after adding E'_p to $F(G)$, some black leaves (or possibly hybrid ones) are left. Thus we add edges, each connecting a black leaf and either a hybrid one or a white vertex which is not a leaf.

It follows that an optimal solution E' to $F(G)$ is obtained and then we convert it into an optimal solution E_f to G.

[Procedure FindEdgesBP**]** When $|LF(G)| \neq 4$, it finds an edge set E'_p to be added to $F(G)$ such that L_p is included of a 3-component S of $F(G) + E'_p$ and E'_p is legal.

Let us linearly order vertices of L_p as $\{v_1, \ldots, v_{|L_p|}\}$ by traversing $ET(F(G))$ according to a fixed linear ordering β. Let $\beta_p(v)$ denote this new ordering on vertices $v \in L_p$, where if $\beta(u) < \beta(v)$ then $\beta_p(u) < \beta_p(v)$ for $u, v \in L_p$.

First, if $|L_p| = 2$ then we can easily obtain such an edge set E'_p with $|E'_p| = 1$. Next we consider the case with $|L_p| \geq 6$.

There are two cases:

(i) There exists a pair $b \in B_p$ and $w \in W_p$ satisfying either $\beta_p(b) + |L_p|/2 = \beta_p(w)$ or $\beta_p(b) = \beta_p(w) + |L_p|/2$ ($w = v_1$ and $b = v_{1+|L_p|/2}$ in Fig. 5);

(ii) Otherwise (see Fig. 6).

Let $L_{p_1} = \{v_2, \ldots, v_{|L_p|/2}\}$ and $L_{p_2} = \{v_{2+|L_p|/2}, \ldots, v_{|L_p|}\}$ in Fig. 5, while $L_{p_1} = \{v_3, \ldots, v_{|L_p|/2}\}$ and $L_{p_2} = \{v_{3+|L_p|/2}, \ldots, v_{|L_p|}\}$ in Fig. 6.

[The case (i)] As shown in Fig. 5, the number of black leaves in L_{p_1} is equal to that of white leaves in L_{p_2} and the number of white leaves in L_{p_1} is equal to that of black leaves in L_{p_2}, because $B_p \cup W_p = L_p$, $|B_p| = |W_p|$ and $|L_{p_1}| = |L_{p_2}|$.

Then we can easily construct a matching $\mathcal{M}$ consisting of edges (u, w) satisfying either ($u \in W_p \cap L_{p_1}$ and $w \in B_p \cap L_{p_2}$) or ($u \in B_p \cap L_{p_1}$ and $w \in W_p \cap L_{p_2}$). Let $E'_p = \mathcal{M} \cup \{(v_1, v_{1+|L_p|/2})\}$.

[The case (ii)] First we find two leaves $w \in W_p$ and $b \in B_p$ with consecutive order of β_p (in Fig. 6, $w = v_1$ and $b = v_2$). In this case, $u \in L_p$ with $|\beta_p(w) - \beta_p(u)| = |L_p|/2$ is a white leaf and $v \in L_p$ with $|\beta_p(b) - \beta_p(v)| = |L_p|/2$ is a black leaf (in Fig. 6, $u = v_{1+|L_p|/2} \in W_p$ and $v = v_{2+|L_p|/2} \in B_p$). Similarly to (i), $|L_{p_1} \cap B_p| = |L_{p_2} \cap W_p|$ and $|L_{p_1} \cap W_p| = |L_{p_2} \cap B_p|$. Hence we can construct a matching $\mathcal{M}$ similarly to (i). Let $E'_p = \mathcal{M} \cup \{(v_1, v_{2+|L_p|/2}), (v_2, v_{1+|L_p|/2})\}$.

Let us note that FindEdgesBP is a combination of two procedures ETC and AETC in [5]. ETC and AETC, however, can be used only to 2-edge-connect a given connected graph, while we generalize them so that may be used in augmenting edge-connectivity by one for a σ-edge-connected graph ($\sigma \geq 1$). ETC (AETC, respectively) corresponds to Step 6 (Step 11) of FindEdgesBP.

B. Description of the Algorithm

We give formal description of **Algorithm** S-Sol_$(\sigma+1)$ECABP and **Procedure** FindEdgesBP.

Algorithm S-Sol_$(\sigma+1)$ECABP
Input: A connected graph $G = (V, E)$,
with a bipartition $\pi = \{V_B, V_W\}$ with $|V_B| \geq |V_W|$.

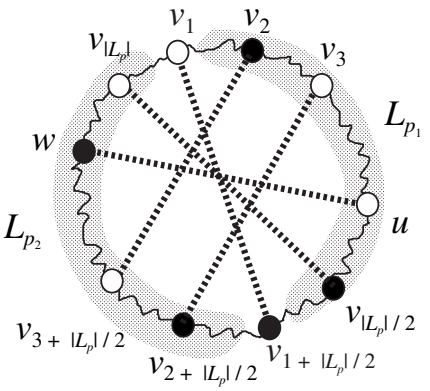

Fig. 5. Schematic explanation of E'_p at Step 6 of FindEdgesBP: each dashed line denotes an edge of E'_p and wavy lines show an Eulerian closed trail $ET(F(G))$.

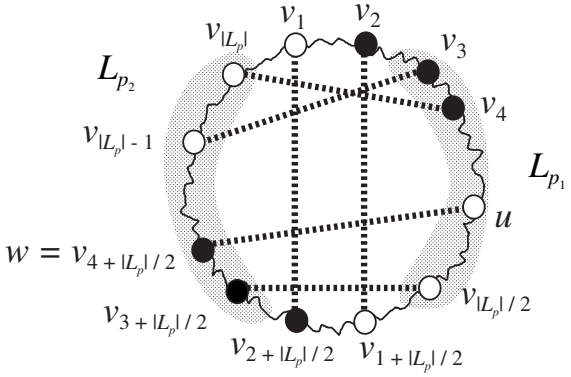

Fig. 6. Schematic explanation of E'_p at Step 11 of FindEdgesBP: each dashed line denotes an edge of E'_p and wavy lines show an Eulerian closed trail $ET(F(G))$.

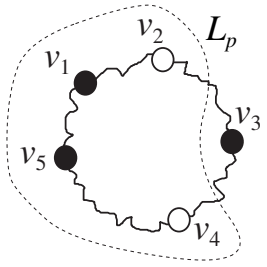

Fig. 7. Schematic explanation of Step 11 (i) of S-Sol_$(\sigma+1)$ECABP: choosing $\{v_5, v_1\}$ as B_p ensures existence of a pair v_a and $v_{a+|L_2|/2}$ in Step 3 of FindEdgesBP, where wavy lines show an Eulerian closed trail $ET(F(G))$.

Output: An edge set E_f with minimium size such that $(V, E \cup E_f)$ is $(\sigma+1)$-edge-connected and E_f is legal.

1: Construct a structural graph $F(G) = (V(F(G)), E(F(G)))$;
2: Find a linear ordering β;
3: $E'_2 \leftarrow \emptyset$, $B \leftarrow BF(G)$, $W \leftarrow WF(G)$, $H \leftarrow HF(G)$, and $L \leftarrow LF(G)$;
4: **if** $H \neq \emptyset$ **then**
5: Choose $\min\{\lfloor(|L| - 2|W|)/2\rfloor, |H|\}$ hybrid leaves, insert them into W (regarded as white ones) and insert the other hybrid ones into B;
6: $H \leftarrow \emptyset$; /* After this step, we have $|B| \geq |W|$ and $|L| = |B| + |W|$, since $|W| \leq \lfloor|L|/2\rfloor$. */
7: **end if**
8: **if** $|LF(G)| = 4$ **then**
9: Find an edge set E_f by Lemma 4.2, output E_f and terminate;
10: **else** /* $|LF(G)| \neq 4$ */
11: Choose a leaf set $B_p \subseteq B$ with $|B_p| = |W|$ as in (i) or (ii):
(i) If $|W| = 2$ two black leaves b, $b' \in B$ ($|B| \geq 3$) satisfying either $(\beta(b) + 1 = \beta(b'))$ or $(\beta(b) = 1$ and $|L| = \beta(b'))$ (see Fig. 7) and $B_p \leftarrow \{b, b'\}$;
(ii) Otherwise /* $|W| \geq 3$ */ choose $B_p \subseteq B$ arbitrarily;
12: $W_p \leftarrow W$, $L_p \leftarrow B_p \cup W_p$;
13: (i) Compute a linear ordering β_p on L_p by traversing $ET(F(G))$ according to β such that, for any $v, w \in L_p$, if $\beta(v) < \beta(w)$ then $\beta_p(v) < \beta_p(w)$;
(ii) Let $L_p = \{v_1, \ldots, v_{|L_p|}\}$ with indices denoting the order of β_p;
14: Find an edge set E'_1 by applying FindEdgesBP to L_p;
15: $B \leftarrow B - B_p$;
16: **end if**
17: **if** $|B| > 0$ **then**
18: $E'_2 \leftarrow \{(b, w) \mid b \in B\}$, where w is set to a leaf $v_1 \in WF(G) \cup HF(G)$ defined at Step 2, 5 or 10 of FindEdgesBP;
19: **end if**
20: $E' \leftarrow E'_1 \cup E'_2$;
21: Output $\xi(E') = \{(n_b, n_w) \mid (b, w) \in E'\}$, where $(b \in BF(G)$ and $w \in WF(G) \cup HF(G))$ and $(n_b \in \rho^{-1}(b)$ and $n_w \in \rho^{-1}(w) \cap V_W)$. □

Procedure FindEdgesBP
Input: Leaf sets L_p, B_p and W_p, and a linear ordering β_p on $L_p = \{v_1, \ldots, v_{|L_P|}\}$ with indices representing the order of β_p.
Output: An edge set E'_p.

1: **if** $|L_p| = 2$ **then**
2: $E'_p \leftarrow \{(v_1, v_2)\}$, where $v_1 \in W_p$ and $v_2 \in B_p$ without loss of generality;
3: **else if** there exists a subscript a satisfying either $(v_a \in B_p$ and $v_{a+|L_p|/2} \in W_p)$ or $(v_a \in W_p$ and $v_{a+|L_p|/2} \in B_p)$ **then**
4: /* Execute Steps 5 and 6 */
5: Regard v_c as v_1 and assume that $v_a \in W_p$ without loss of generality;

Let $L_{p_1} = \{v_2, \ldots, v_{|L_p|/2}\}$ and $L_{p_2} = \{v_{2+|L_p|/2}, \ldots, v_{|L_p|}\}$;

6: $E'_p \leftarrow \mathcal{M} \cup \{(v_1, v_{1+|L_p|/2})\}$ (see Fig. 5), where $\mathcal{M}$ is a matching with $|\mathcal{M}| = |L_p|/2 - 1$ consisting of edges (u, v) satisfying ($u \in L_{p_1} \cap W_p$ and $v \in L_{p_2} \cap B_p$) or ($u \in L_{p_1} \cap B_p$ and $v \in L_{p_2} \cap W_p$); /* Since $|B_p| = |W_p|$ and $|L_{p_1}| = |L_{p_2}|$, such a matching $\mathcal{M}$ exists. */

7: **else** /* Such a pair as in Step 3 does not exist. */

8: /* Execute Steps 9–11 */

9: Find any subscript c satisfying either ($v_c \in W_p$ and $v_{c+1} \in B_p$) or ($v_{|L_p|} \in W_p$ and $v_1 \in B_p$);

10: Regard v_c as v_1 and assume that $v_c \in W_p$ without loss of generality;
Let $L_{p_1} = \{v_3, \ldots, v_{|L_p|/2}\}$ and $L_{p_2} = \{v_{3+|L_p|/2}, \ldots, v_{|L_p|}\}$;
/* Then $v_{1+|L_p|/2} \in W_p$ and $v_{2+|L_p|/2} \in B_p$ */

11: $E'_p \leftarrow \mathcal{M}' \cup \{(v_1, v_{2+|L_p|/2}), (v_2, v_{1+|L_p|/2})\}$ (see Fig. 6), where $\mathcal{M}'$ is a matching with $|\mathcal{M}'| = |L_p|/2 - 2$ consisting of edges (u, v) satisfying ($u \in L_{p_1} \cap W_p$ and $v \in L_{p_2} \cap B_p$) or ($u \in L_{p_1} \cap B_p$ and $v \in L_{p_2} \cap W_p$);
/* Since $|B_p| = |W_p|$ and the $|L_{p_1}| = |L_{p_2}|$, such a matching $\mathcal{M}'$ exists. */

12: **end if**

13: Output E'_p. /* E'_p is legal and $|E'_p| = |L_p|/2$ */ □

C. Correctness of the Algorithm

We prove correctness of the algorithm by using several lemmas. First, we have the next lemma for a structural graph $F(G)$.

Lemma 4.1 [12] In $F(G)$ with $|LF(G)| \geq 4$, if there are distinct four leaves $v, w, x, y \in V(F(G))$ with $\beta(v) < \beta(x) < \beta(w) < \beta(y)$ then there are distinct four vertices $n_v, n_w, n_x, n_y \in V(G)$ such that $|LF(G+\{n_v, n_w\})| = |LF(G)|-2$ and $|LF(G+\{n_x, n_y\})| = |LF(G)| - 2$.

The lemma for the case with $|LF(G)| = 4$ follows from Proposition 3.1 and Lemma 4.1.

Lemma 4.2 [12] Assume that $|LF(G)| = 4$.

(i) If $|BF(G)| \geq 3$ then there exists an optimal solution E_f with $|E_f| = |BF(G)|$.

(ii) If $|BF(G)| \leq 2$ then let us consider a graph G_c defined in Proposition 3.1.

(ii-1) If G_c is a simple cycle of length four such that two black leaves and two other ones that are either white or hybrid ones appear alternately (see Fig. 8) then there exists an optimal solution E_f with $|E_f| = 3$;

(ii-2) otherwise, there exists an optimal solution E_f with $|E_f| = 2$.

Next we are going to prove the next lemma on a subset of an optimal solution to 3ECABP for a modified cactus $F(G)$ with $\lambda(F(G)) = 2$ and $|LF(G)| \neq 4$.

Lemma 4.3 For a structural graph $F(G)$ with $|LF(G)| \neq 4$, **Procedure** FindEdgesBP finds an edge set E'_p to be added to $F(G)$ such that L_p is included in a 3-component S of $F(G)+E'_p$ and E'_p is legal.

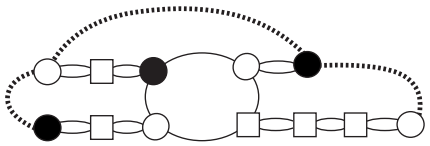

Fig. 8. Schematic explanation of Lemma 4.2 (ii-1), where the set of dashed lines is an optimal solution E' to a structural graph $F(G)$.

Proof: Clearly, for any $(v, w) \in E'_p$, $\{v, w\}$ is included in a 3-component of $F(G) + E'_p$ because there are three edge-disjoint paths between v and w in the graph. Furthermore, it is well-known that $\{u_1, u_2, u_3\}$ is a subset of a 3-component if there are three edge-disjoint paths between u_1 and u_2 and between u_2 and u_3.

Now we consider the three cases:

(i) $|L_p| = 2$;

(ii) $|B_p| = |W_p| \geq 2$ and a subscript a satisfying either ($v_a \in B_p$ and $v_{a+|L_p|/2} \in W_p$) or ($v_a \in W_p$ and $v_{a+|L_p|/2} \in B_p$) exists;

(iii) $|B_p| = |W_p| \geq 3$ and any subscript a satisfying either ($v_a \in B_p$ and $v_{a+|L_p|/2} \in W_p$) or ($v_a \in W_p$ and $v_{a+|L_p|/2} \in B_p$) does not exist.

[The case (i)] Clealy the lemma follows.

[The case (ii)] We have $\lambda(v_1, v_{1+|L_p|/2}; F(G)+E'_p) \geq 3$. Let u be any vertex in $L_p - \{v_1, v_{1+|L_p|/2}\}$. Let $(u, v) \in E'_p$. If $u \in L_{p_1}$ then $w \in L_{p_2}$, while if $u \in L_{p_2}$ then $w \in L_{p_1}$. Since the discussion is symmetric for the two cases where $u \in L_{p_1}$ or $u \in L_{p_2}$, we explain only the case with $u \in L_{p_1}$. As shown in Fig. 9, $F(G) + E'_p$ has three edge-disjoint (v_1, u)-paths P_1, P_2 and P_3. Hence L_p is included in a 3-component of $F(G) + E'_p$.

[The case (iii)] We have $\lambda(v_i, v_{i+|L_p|/2}; F(G) + E'_p) \geq 3$ for $i = 1, 2$. Let u be any vertex in $L_p - \{v_1, v_2, v_{1+|L_p|/2}, v_{2+|L_p|/2}\}$. Let $(u, w) \in E'_p$. If $u \in L_{p_1}$ then $w \in L_{p_2}$, while if $u \in L_{p_2}$ then $w \in L_{p_1}$. Since the discussion is symmetric, we explain only the case with $u \in L_{p_1}$. As shown in Fig. 10 (1), $F(G) + E'_p$ has three edge-disjoint (v_1, v_2)-paths P_1, P_2 and P_3. Furthermore, as in Fig. 10 (2), there are three edge-disjoint (v_1, u)-paths P'_1, P'_2 and P'_3 of $F(G)+E'_p$. Hence L_p is included in a 3-component of $F(G) + E'_p$.

Clearly E'_p is legal. □

Now the correctness of the algorithm is proved by the next lemma.

Lemma 4.4 For a given graph $G = (V, E)$ with $\lambda(G) = \sigma$ and a bipartition $\{V_B, V_W\}$ of V, **Algorithm** S-Sol_$(\sigma+1)$ECABP finds an optimal solution E_f.

Proof: We prove the next lemma by showing that $V(F(G))$ is only one 3-component in $F(G) + E'$ and by counting $|E'|$, where E' is an optimal solution to 3ECABP for $F(G)$.

For any edge set found in Step 9, 14, 18 or 20, there are three cases as follows:

(i) $|LF(G)| = 4$;

(ii) $|LF(G)| \neq 4$ and $F(G)$ is not B-dominant;

(iii) $|LF(G)| \neq 4$ and $F(G)$ is B-dominant.

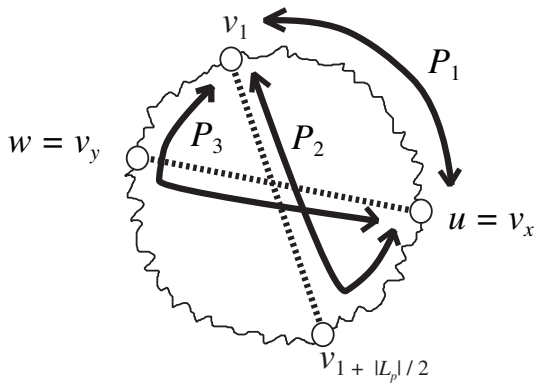

Fig. 9. Schematic explanation of three edge-disjoint (v_1, u)-paths of $F(G)+E'_p$ in the case (ii), where an $ET(F(G))$ is denoted by wavy lines.

[The case (i)] Lemma 4.2 shows that Step 9 finds an edge set E' with $|E'| = \mathcal{L}$ (see Proposition 3.1 (i)) such that $LF(G)$ is included in a 3-component of $F(G) + E'$.

[The case (ii)] We have $B = \emptyset$ just before Step 17 of S-Sol_$(\sigma+1)$ECABP. Lemma 4.3 shows that Steps 10–16 find an edge set $E' = E'_1$ with $|E'| = |E'_1| = |LF(G)|/2 = \mathcal{L}$ (see Proposition 3.1 (iii)) such that $LF(G)$ is included in a 3-component of $F(G) + E'$.

[The case (iii)] Steps 10 –16 of S-Sol_$(\sigma + 1)$ECABP find an edge set E'_1 with $|E'_1| = |W|$ and we have $B \neq \emptyset$ just before Step 17. Since, for each $(b, w) \in E'_2$, $\{b, w\}$ is a subset of a 3-component of $F(G) + E'_1$, Step 20 gives us an edge set $E' = E'_1 \cup E'_2$ with $|E'| = |BF(G)| = \mathcal{L}$ (see Proposition 3.1(iii)) such that $LF(G)$ is included in a 3-component of $F(G) + E'$.

Clearly E'_p is legal in all cases.

Suppose that $F(G) + E'_p$ has a 2-cut $(X, V(F(G)) - X; F(G))$ in the case (i), (ii) or (iii). Then it is a 2-cut of $F(G)$ and, therefore, both X and $V(F(G)) - X$ include at least one leaf of $F(G)$. However this is not possible, since all leaves of $F(G)$ are included in a 3-component of $F(G) + E'_p$. Hence $V(F(G))$ is a 3-component of $F(G) + E'_p$ in all the cases (i), (ii) and (iii). □

D. Time Complexity

A structural graph $F(G)$ can be constructed in $O(|V||E| + |V|^2 \log |V|)$ time [10]. Since all $(\sigma + 1)$-components are extracted in linear time [14], [13], [8], [15] when $1 \leq \sigma \leq 2$, a structural graph can be constructed in linear time in this case. Step 5 can be done in $O(|V|)$ time. FindEdegsBP and Step 18 can be done in $O(|V|)$ time. Finally, an optimal soluion E' to $F(G)$ can be converted into an optimal one E_f to G in $O(|E|)$ time.

From above discussion, Proposition 3.1 and Lemma 4.4, Theorem 1.1 follows.

V. Conclusion

In this paper, we have given a simplified $O(|V||E| + |V|^2 \log |V|)$ time algorithm to find an optimal solution to $(\sigma + 1)$ECABP when $\sigma = \lambda(G)$. Moreover, it is shown that the problem can be solved in linear time when $1 \leq \sigma \leq 2$.

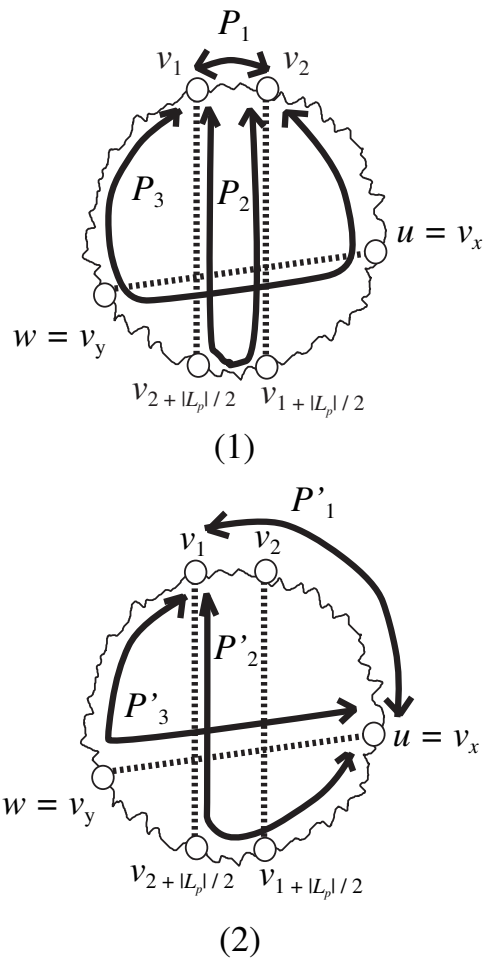

Fig. 10. Schematic explanation of (1) three edge-disjoint (v_1, v_2)-paths and (2) three edge-disjoint (v_1, u)-paths of $F(G) + E'_p$ in the case (iii), where an $ET(F(G))$ is denoted by wavy lines.

Note that, by means of S-Sol_$(\sigma+1)$ECABP, we can easily solve a problem such that a subset $\Gamma \subseteq V$ is additionally given in kECABP and we require that $\lambda(\Gamma; G) \geq (\sigma + 1)$ when $\sigma = \lambda(V; G) = \lambda(\Gamma; G)$.

Giving an efficient algorithm for $(\sigma + \delta)$ECAMP with $\sigma = \lambda(G)$ and $\delta > 1$ under several conditions is left as future research.

Acknowledgements

The research is partly supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan (No. 22500029).

References

[1] J. Bang-Jensen, H. N. Gabow, T. Jordán, and Z. Szigeti. Edge-connectivity augmentation with partition constraints. *SIAM J. Discrete Mathematics*, 12(2):160–207, 1999.

[2] Y. Chen, H. Wei, P. Huang, W. Shih, and T. Hsu. The bridge-connectivity augmentation problem with a partition constraint. *Theor. Comput. Sci.*, 411(31-33):2878–2889, 2010.

[3] K. P. Eswaran and R. E. Tarjan. Augmentation problems. *SIAM J. Comput.*, 5:653–655, 1976.
[4] A. Frank. Augmenting graphs to meet edge connectivity requirements. *SIAM J. Discrete Mathematics*, 5(1):25–53, 1992.
[5] P. Huang, H. Wei, W. Lu, W. Shih, and T. Hsu. Smallest bipartite bridge-connectivity augmentation. *Algorithmica*, 54(3):353–378, 2009.
[6] 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), 8–12.
[7] T. Mashima, S. Taoka, and T. Watanabe. A 2-approximation algorithm to $(k+1)$-edge-connect a specified set of vertices in a k-edge-connected graph. *IEICE Trans. Fundamentals*, E88-A(5):1290–1300, May 2005.
[8] H. Nagamochi and T. Ibaraki. A linear time algorithm for computing 3-edge-connected components in a multigraph. *Japan J. Industrial and Applied Math.*, 9(7):163–180, 1992.
[9] H. Nagamochi, S. Nakamura, and T. Ibaraki. A simplified $\tilde{O}(nm)$ time edge-splitting algorithm in undirected graphs. *Algorithmica*, 26:50–57, 2000.
[10] H. Nagamochi, S. Nakamura, and T. Ishii. Constructing a cactus for minimum cuts of a graph in $O(mn + n^2 \log n)$ time and $O(m)$ space. *IEICE Trans. Fundamentals*, E86-D(2):179–185, 2003.
[11] D. Naor, D. Gusfield, and C. Martel. A fast algorithm for optimally increasing the edge connectivity. *SIAM J. Comput.*, 26(4):1139–1165, August 1997.
[12] T. Oki, S. Taoka, T. Mashima, and T. Watanabe. A fast algorithm for augmenting edge-connectivity by one with bipartition constraints. *IEICE Trans. Information and Systems*, E95-D(3), 2012 (to appear).
[13] S. Taoka, T. Watanabe, and K. Onaga. A linear-time algorithm for computing all 3-edge-connected components of an multigraph. *IEICE Trans. Fundamentals*, E75–A(3):410–424, 1992.
[14] R. E. Tarjan. A note on finding the bridges of a graph. *Information Processing Letters*, 2:160–161, 1974.
[15] Y. H. Tsin. Yet another optimal algorithm for 3-edge-connectivity. *J. Discrete Algorithms*, 7(1):130–146, 2009.
[16] T. Watanabe and A. Nakamura. Edge-connectivity augmentation problems. *Journal of Computer and System Sciences*, 35(1):96–144, 1987.