Vol.2010-AL-131 No.10
2010/9/22

A Fast Algorithm for $(\sigma+1)$-Edge-Connectivity Augmentation of a σ-Edge-Connected Graph with Multipartition Constraints

Tadachika Oki,[†1] Satoshi Taoka[†1] and Toshimasa Watanabe [†1]

The k-edge-connectivity augmentation problem with multipartition constraints (kECAM for short) is defined by "Given an undirected graph $G = (V, E)$ and a multipartition $\pi = \{V_1, \ldots, V_r\}$ of V with $V_i \cap V_j = \emptyset$ for $\forall i, j \in \{1, \ldots, r\}$ $(i \neq j)$, find an edge set E' of minimum cardinality, consisting of edges that connect distinct members of π, such that $G' = (V, E \cup E')$ is k-edge-connected." In this paper, we propose a fast algorithm for finding a solution to $(\sigma+1)$ECAM when G is σ-edge-connected $(\sigma > 0)$, and show that the problem can be solved in linear time if $\sigma \in \{1, 2\}$. The main idea is to reduce $(\sigma+1)$ECAM to the bipartition case, that is, $(\sigma+1)$ECAM with r = 2. Moreover, we propose a parallel algorithm for finding a solution to $(\sigma+1)$ECAM, when a structural graph $F(G)$ which represents all minimum cuts of G is given and σ is odd.

1. Introduction

The k-edge-connectivity augmentation problem (kECA for short) is defined by "Given an undirected graph $G = (V, E)$ find an edge set E' of minimum cardinality, such that $G' = (V, E \cup E')$ is k-edge-connected." We often denote G' as $G + E'$, and E' is called a *solution* to the problem. There are several applications for construction of a fault-tolerant network, and so on. It is called the *k-edge-connectivity augmentation problem with multipartition constraints* (kECAM, for short) when a multipartition $\pi = \{V_1, \ldots, V_r\}$ $(r \geq 2)$ of V with $V_i \cap V_j = \emptyset$, $\forall i, j \in \{1, \ldots, r\}$ $(i \neq j)$, is additionally given and we require that E' consists of edges connecting between V_i and V_j $(i, j \in \{1, \ldots, r\}, i \neq j)$ (see Fig. 1). A multipartite graph is a graph (V, E) such that V is partitioned into r sets $V^1 \ldots V^r$, and any edge $(u, v) \in E$ satisfies $(u \in V^i$ and $v \in V^j)$ or $(u \in V^j$ and $v \in V^i)$ $(i, j \in \{1, \ldots, r\}, i \neq j)$. A multipartite graph is denoted by $G = (V^1 \cup \ldots V^r, E)$. If G is multipartite and we set $V_i = V^i$ $(\forall i \in \{1, \ldots, r\})$ in kECAM then G' is multipartite. This problem, denoted as M-kECAM, is a typical subproblem of kECAM, and there are several applications for security of statistic data of a cross tabulated table[8)], and so on.

Many algorithms for kECA have been proposed. 4) proposed a linear time algorithm for 2ECA, and 19) and 11) proposed polynomial time algorithms for kECA.

Now, we introduce the k-vertex-connectivity augmentation problem (kVCA for short) to describe existing results. The problem is defined by "Given an undirected graph $G = (V, E)$ find an edge set E' of minimum cardinality, such that $G' = (V, E \cup E')$ is k-vertex-connected." We often denote G' as $G + E'$, and E' is called a *solution* to the problem. It is called the *k-vertex-connectivity augmentation problem with multipartition constraints* (kVCAM, for short) when kVCA has same partition constraints in kECAM.

Moreover, graph connectivity augmentation problems with partition constraints have been studied. In kECAM, 8) proposed a linear time algorithm for B-2ECAB (M-kECAM when G is bipartite and π is bipartition) , and a parallel algorithm on EREW PRAM. 1) proposed an $O(|V|(|E| + |V| \log |V|) \log |V|)$ time algorithm for kECAM. 14) proposed an $O(|V||E| + |V|^2 \log |V|)$ time algorithm for B-$(\sigma+1)$ECAB. 2) proposed an linear time algorithm for 2ECAM, and a parallel algorithm on EREW PRAM.

Furthermore, in kVCAM, 6) proposed a linear time algorithm for B-2VCAB (M-kVCAM when G is bipartite and π is bipartition). 7) proposed a linear time algorithm for 2VCAM.

The main result of the paper is to propose a fast algorithm for obtaining an optimum solution to $(\sigma+1)$ECAM when G is σ-edge-connected and a structural graph $F(G)$ which represents all minimum cuts of G is given. The time complexity of the proposed algorithm is $O(|V||E| + |V|^2 \log |V|)$ because $F(G)$ can be constructed in $O(|V||E| + |V|^2 \log |V|)$ time[12)]. Moreover, when $\sigma \in \{1, 2\}$, this is a linear time because $F(G)$ can be constructed in $O(|V| + |E|)$ time. Note that the proposed algorithm is faster than the algorithm proposed in 1) for $(\sigma+1)$ECAM G is σ-edge-connected. Furthermore, we propose a parallel algorithm for finding a solution to $(\sigma+1)$ECAM, when a structural graph $F(G)$ is given and σ is odd in $O(\log |V|)$ parallel time on an EREW PRAM using a linear number of processors.

The paper is organized as follows. Section 2 provides some definitions and notations. Section 3 shows a lower bound on solutions to this problem. Section 4 presents an algorithm for finding a solution to this problem. Its correctness and time complexity in Sects. 4.2 and 4.3. In Sect. 5, we propose a parallel algorithm for $(\sigma+1)$ECAM when σ is odd. The concluding remarks are given in Sect. 6.

†1 Graduate School of Engineering, Hiroshima University

2. Definitions

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 considered, 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 by (u, v). 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 written (X, X') if G is clear from context. We denote $d_G(X) = |(X, V - X; G)|$ which is called *degree* of X (in G). For a vertex v, the total number of edges incident to v is called degree of v and denote $d_G(v)$ (in G). A *cut* in a pair of sets $\{X, V - X\}$ of G and, for simplicity, $(X, V - X; G)$ is also called a cut. It is called k-cut when $|(X, V - X)| = k$. For a set E' of edges, let $G + E'$ denote the graph by adding all edges of E'.

A *trail* is a sequence of edges (v_0, v_1), (v_1, v_2), . . ., (v_{r-1}, v_r) in which there may appear the same endvertices. It is called a *closed trail* when $v_0 = v_r$. A closed trail is called an *Eulerian closed trail* of G if all edges of G are included. A *path* is a trail such that all vertices $v_0, v_1, \ldots, v_r$ are distinct. A *cycle* consists of a path with $r \geq 2$ and an edge (v_r, v_0).

We say that G is *connected* if there is a path for any pair of vertices. For two vertices $u, v \in V$, let $\lambda(u, v; G)$ denote the maximum number of edge-disjoint paths between u and v in G. Edge-connectivity $\lambda(G)$ of G is defined by $\lambda(G) = \min\{\lambda(u, v; G) \mid u, v \in V\}$, and we say that G is *k-edge-connected* if $\lambda(G) \geq k$, for a nonnegative integer k. If G is k-edge-connected then a graph constructed by deleting any set of $k-1$ edges is connected. Any set $Z \subseteq V$ that is a maximal vertex set such that $\lambda(u, v; G) \geq k$ holds for any pair of vertices $u, v \in Z$. Z is called a *k-edge-connected component* (k-component, for short) of G. In addition, we call Z a *block* when $k = 2$. Z is also called a *leaf k-component* if and only if $d_G(Z) = \lambda(G)$. Note that distinct k-components are disjoint.

A *cactus* is an undirected connected graph in which any pair of cycles shares at most one vertex. A structural graph $F(G) = (V(F(G)), E(F(G)))$[9] of G with $\lambda(G) = \sigma$ (see Fig. 2) is a representation for all minimum cuts of G. $F(G)$ is an edge-weighted cactus of $O(|V|)$ vertices and edges such that each tree edge (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 $\lambda(G)$ is odd then $F(G)$ is a weighted tree. Each vertex in G maps to exactly one vertex in $F(G)$. 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 of $F(G)$, and vice versa. Let ρ: $V(G) \to V(F(G))$

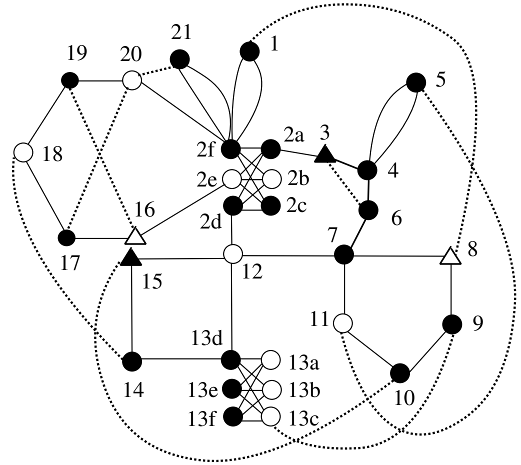

Fig. 1 A graph G with $\lambda(G) = 2$, where a closed circle (an open circle, a close triangle and a open triangle, respectively) represents a vertex which belongs to V_1 (V_2, V_3 and V_4). The set of dashed lines represents a solution $E_f = \{(1, 8), (3, 6), (5, 11), (9, 13), (10, 15), (14, 18), (16, 19), (17, 20), (21, 20)\}$

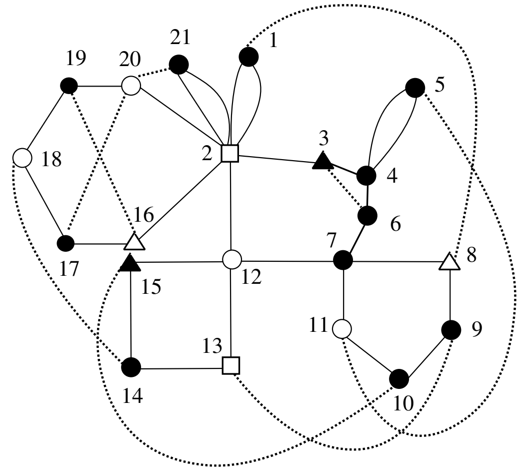

Fig. 2 The set of dashed lines represents a solution $E' = \{(1, 8), (3, 6), (5, 11), (9, 13c), (10, 15), (14, 18), (16, 19), (17, 20), (21, 20)\}$ on a structural graph $F(G)$ for G in Fig. 1

denote this mapping. We use the following notations: $\rho(X) = \{\rho(v) \mid v \in X\}$ for $X \subseteq V$, $\rho(Y)^{-1} = \{v \in V \mid \rho(v) \in Y\}$ for $Y \subseteq V(F(G))$. A vertex $y \in V(F(G))$ with $\rho(y)^{-1} = \emptyset$ is called *empty vertex*. Let $\varepsilon(G) \subseteq V(F(G))$ denote the set of all empty vertices of $F(G)$.

For any cut $(X, V(F(G)) - X; F(G))$, if summation of weights of all edges in the cut is equal to σ then $(\rho^{-1}(X), V - \rho^{-1}(X); G)$ is a σ-cut (a minimum cut) of G. Conversely, for any σ-cut $(X, V - X; G)$, $F(G)$ has at least one cut $(Y, V(F(G)) - Y; F(G))$ in which summation of weight of all edges in the cut is equal to σ, where Y is a vertex set such that $\rho(X) = Y - \varepsilon(G)$. Each $(\sigma + 1)$-component S of G is represented as a vertex $\rho(S) \in V(F(G)) - \varepsilon(G)$, and for any vertex $v \in V(F(G)) - \varepsilon(G)$, $\rho^{-1}(v)$ is $(\sigma + 1)$-component of G. For any $v \in V(F(G)) - \varepsilon(G)$, if summation of weights of all edges that are incident to v in $F(G)$ equals to σ, then v is called a *leaf* of $F(G)$ and $\rho^{-1}(v)$ is a leaf $(\sigma + 1)$-component. Conversely, for any leaf $(\sigma + 1)$-component L of G, $\rho(L)$ is a leaf of $F(G)$. Let $LF(G)$ denote the set of all leaves of $F(G)$. It is shown that $F(G)$ can be constructed in $O(|V||E| + |V|^2 \log |V|)$[12].

If $F(G)$ has any bridge of weight $\lambda(G)$ then we replace such a bridge by a pair of multiple edges, each having weight $\lambda(G)/2$. We consider such a pair of multiple edges to be a cycle of length two. We call this graph a *modified cactus*, and we assume $F(G)$ is a modified one in this paper unless otherwise stated. Note that a modified cactus $F(G)$ is also a structural graph of G and $\lambda(F(G)) = 2$.

Given a structural graph $F(G)$ of a graph $G = (V, E)$ with multipartition constraints $\pi = \{V_1, V_2, \ldots V_r\}$, we classify vertices v into $(r+1)$ types of vertices in $F(G)$ as follows: (i) $\rho^{-1}(v) \subseteq V_i$, $(i \in \{1, \ldots, r\})$, (v is called a C_i *vertex of* $F(G)$), (ii) $\rho^{-1}(v) \cap V_i \neq \emptyset, \rho^{-1}(v) \cap V_j \neq \emptyset, i \neq j,\ i, j \in \{1, \ldots, r\}$ (v is called a *hybrid one of* $F(G)$). The set of C_i *leaves*, *hybrid leaves*, respectively, is denoted by $L_iF(G)$ or $HF(G)$. In this paper, without loss of generality, we assume that, for any $i, j \in \{1, \ldots, r\}$, if $i < j$ then $|L_iF(G)| \geq |L_jF(G)|$ (that is $L_iF(G)$ is stored is non decreasing order of $|L_iF(G)|$). If $|L_1F(G)| > \sum_{j=2}^{r} |L_iF(G)| + |HF(G)|$ holds then $F(G)$ is called C_1-*dominated*[8].

In figures of this paper, a hybrid vertex is represented by a square, and any C_1 one, a C_2 one, C_3 one and C_4 one are represented by a closed circle, an open one, an open triangle and a closed one, respectively.

3. A Lower Bound of a Solution to $(\sigma + 1)$ECAM

In the rest of the paper, we set $\lambda(G) = \sigma$.

In this section, a lower bound of on any solution to $(\sigma + 1)$ECAM is given since $(\sigma + 1)$ECAM is a subproblem of kECAM, we obtain the following proposition by setting $k = \sigma + 1$ for a lower bound shown in 1) to kECAM.

Proposition 3.1 Let G_c be a graph obtained from $F(G)$ by shrinking all multiple edges in $F(G)$ (with all resulting self-loops removed). The number $\mathcal{L}$ of edges required to $(\sigma + 1)$-edge-connect a σ-edge-connected graph G is given as follows. (1) If a graph G_c is a simple cycle of length four such that (i) two C_1 leaves and two C_2 ones appear alternately or (ii) A C_1 leaf, a C_2 one , a C_1 one and a C_3 one appear in order without loss of generality (see Fig. 5) then $\mathcal{L} = 3$. (2) If a graph G_c is a simple cycle of length six such that two C_1 leaves, two C_2 ones and two C_3 ones appear alternately (see Fig. 6) then $\mathcal{L} = 4$. (3) Otherwise, $\mathcal{L} = \max_{i=1}^{r}\{|L_iF(G)|, \lceil |LF(G)|/2 \rceil\}$.

The algorithm to be proposed in the next section finds a set of edges whose number is equal to the lower bound of Proposition 3.1, showing that the algorithm finds an optimal solution.

4. A Proposed Algorithm for $(\sigma + 1)$ECAM

In this section, we propose an algorithm for $(\sigma + 1)$ECAM when $\lambda(G) = \sigma$.

4.1 An Outline of the Proposed Algorithm

Clearly it is enough to consider $F(G)$ instead of G for $(\sigma + 1)$ECAM. In order to efficiently augment the connectivity of G by one, we require each edge (u, v) in a solution for $F(G)$ to connect at least one leaf. Although connecting two leaves is desirable, it is not always the case. Furthermore, in order to keep multipartition constraints, u and v should include in different partitions.

Our proposed algorithm solves $(\sigma+1)$ECAM by reducing it to $(\sigma+1)$ECAB as follows. First, if $HF(G) \neq \emptyset$ then we make the gap between the number of C_1 leaves and that of C_2 ones as narrow as possible by regarding each hybrid leaf as a C_1 leaf or a C_2 one. This is because any hybrid leaf can be treated as a C_i one. Note that, after this operation, the facts $|L_1F(G)| = \max_{j=1}^{r} |L_jF(G)|$ and $|L_2F(G)| = \max_{j=2}^{r} |L_jF(G)|$ are kept.

If $F(G)$ is C_1-dominated then we solve $(\sigma + 1)$ECAB for a bipartition $\{B, W\}$, where we set $B \leftarrow L_1F(G)$ and $W \leftarrow LF(G) - B$. Note that any hybrid leaf is regarded as a C_2 one in this case. If $F(G)$ is not C_1-dominated then it is reduced to $(\sigma + 1)$ECAB in the following two phases.

(The first phase) In order to reduce $(\sigma + 1)$ECAM to $(\sigma + 1)$ECAB, we find an edge set E'_f such that $LF(G + E'_f)$ has exact $LF(G) - 2|E'_f|$ leaves and can be partitioned into B_2 and W_2 such that $|W_2| \leq |B_2| \leq |W_2| + 1$ and $i \neq j$ for any $i, j \in \{1, \ldots, r\}$ with

$L_iF(G) \cap B_2 \neq \emptyset$ and $L_iF(G) \cap W_2 \neq \emptyset$ (see Fig. 3). In order to find such E'_f, we determine the minimum integer j_h with $\sum_{i=1}^{j_h} |L_iF(G)| \geq \lceil |LF(G)|/2 \rceil$, and if $a > 0$ then we find a vertex set $B_1 \subset L_{j_h}F(G)$ with $|B_1| = a + 1$ and a vertex set $W_1 \subset L_1F(G)$ with $|W_1| = a + 1$ arbitrarily, where $a = \sum_{i=1}^{j_h} |L_iF(G)| - \lceil |LF(G)|/2 \rceil$. Note that $1 \neq j_h$, $|L_{j_h}F(G)| \geq a + 1$ and $|L_1F(G)| \geq a + 1$ hold because of the fact that $F(G)$ is not C_1-dominated, the way to determine j_h and $|L_1F(G)| = \max_{i\in\{1,\ldots,r\}} |L_iF(G)|$, respectively. Then we can find the edge set E'_f by adapting an algorithm $Sol_(\sigma+1)$ECAB for B_1 and W_1, where an algorithm solving $(\sigma + 1)$ECAB is denoted by $Sol_(\sigma + 1)$ECAB.

(The second phase) We obtain an edge set E'_2 which is a solution found by $Sol_(\sigma + 1)$ECAB under the situation that a structural graph is $F(G)$ and a black (white, respectively) leaf set in $F(G)$ is regarded to B_2 (W_2).

Finally, we find a solution to G from an edge set found by the above reduction.

In the proposed algorithm, we use a *special type of preorder* (denoted as $\beta(v)$) of a modified cactus $F(G)$, as used in 13), that is useful in efficiency finding a solution to $F(G)$. It can be found in linear time by searching which is based on a depth-first search and which is assigned to each vertex v from 1 to $|V(F(G))|$. Note that traversing vertices in the order of $\beta(v)$ from 1 to $|V(F(G))|$ makes an Eulerian closed trail.

The algorithm is described in in Algorithm $Sol_(\sigma + 1)$ECAM and the subroutine is detailed in Subroutine FIND_EDGES.

Algorithm $Sol_(\sigma + 1)$ECAM
Input: A connected graph $G = (V, E)$, with multipartition constraints $\pi = \{V_1, \ldots, V_r\}$.
Output: An edge set E_f with minimum cardinality such that E_f consists of edges connecting between V_i and V_j ($i \neq j$) and such that $(V, E \cup E_f)$ is $(\sigma + 1)$-edge-connected.

Compute a structural graph $F(G) = (V(F(G)), E(F(G)))$.
$H \leftarrow HF(G)$, and $L_i \leftarrow L_iF(G)$ ($\forall i \in \{1, \ldots, r\}$) (see the definition of $L_iF(G)$ in Section 2), where, an ordered family of sets of leaves $L_1, L_2, \ldots, L_r$ (descending order) by BUCKETSORT (PARALLEL BUCKETSORT[5] in parallelization (see Section 5)). (All we have to do is to compute the first largest cardinality of a sets of leaves and second largest cardinality of one.)
if $|L_1| = |L_2| = |L_3| = 2$ and $\sum_{i=4}^{r} |L_i| = 0$ **then**
  Obtain an edge set E' by Lemma 4.2.
else
  if $H \neq \emptyset$ **then**
    if $|H| \leq |L_1| - |L_2|$ **then**
      Add all hybrid leaves to L_2.
    else $\{|H| > |L_1| - |L_2|\}$
      Add $\lceil(|H| - |L_1| + |L_2|)/2\rceil$ hybrid leaves to L_1, and add the other hybrid leaves to L_2. /* After this, $|L_1| = |L_2| + 1$ or $|L_1| = |L_2|$ holds. */
    end if
  end if
  if $|L_1| > \lceil |LF(G)/2| \rceil$ **then**
    $B \leftarrow L_1$ and $W \leftarrow \bigcup_{j=2}^{r} L_j$, and find a set E' of edges which is a solution found by $Sol_(\sigma + 1)$ECAB under the situation that a structural graph is $F(G)$ and a black (white, respectively) leaf set in $F(G)$ is regarded as B (W).
  else $\{|L_1| \leq \lceil |LF(G)/2| \rceil\}$
    Obtain an edge set E' by FIND_EDGES;
  end if
end if
Output $E_f = \{(n_u, n_v) \mid (u, v) \in E'\}$, where u (v, respectively) is a type C_k (C_l) leaves or hybrid leaves of $F(G)$ ($k \neq l$), and n_u and n_v are vertices with different types of vertices in $\rho^{-1}(u)$ and $\rho^{-1}(v)$, respectively.

Subroutine FIND_EDGES
Input: Leaf sets $L_1, \ldots, L_r$
Output: An edge set E'

Set $p_i \leftarrow |L_i|$ for any $i \in \{1, \ldots, r\}$, $j_h \leftarrow 1$, $s \leftarrow p_1$.
for $i \leftarrow 1$; $i < r$; $i++$ **do**
  if $s \geq \lceil |LF(G)|/2 \rceil$ **then**
    $j_h \leftarrow i$, break;
  else $\{s < \lceil |LF(G)|/2 \rceil\}$
    $s \leftarrow s + p_{i+1}$;
  end if
end for
$a \leftarrow s - \lceil |LF(G)|/2 \rceil$; /* $s = \sum_{i=1}^{j_h} p_i$ */
if $a > 0$ **then**
  Find a vertex set $B_1 \subset L_{i_h}$ with $|B_1| = a + 1$ and a vertex set $W_1 \subset L_1$ with $|W_1| = a + 1$ arbitrarily; /* See Fig. 3. */
  Obtain an edge set E'_1 with $|E'_1| = a + 1$ which is a solution found by $Sol\_(\sigma +$

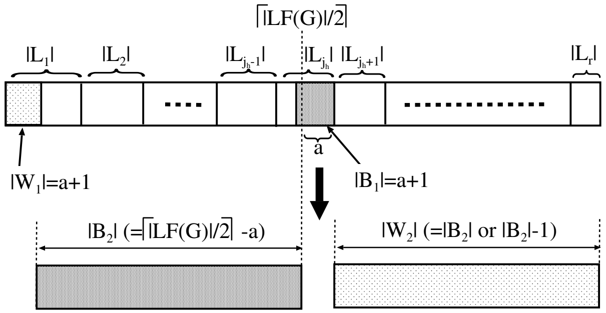

Fig. 3 Schematic explanation of reduction to $(\sigma+1)$ECAB

1)ECAB[14] under the situation that a structural graph is $F(G)$ and a black (white, respectively) leaf set in $F(G)$ is regarded as B_1 (W_1).

Delete an arbitrary edge (x, y) from E'_1 (we suppose that $x \in B_1$ and $y \in W_1$ without loss of generality), and set $B_1 \leftarrow B_1 - \{x\}$ and $W_1 \leftarrow W_1 - \{y\}$;

else

$E'_1 \leftarrow \emptyset$;

end if

Set $B_2 \leftarrow \bigcup_{i=1}^{j_h} L_i - (B_1 \cup W_1)$ and $W_2 \leftarrow LF(G) - (B_1 \cup W_1 \cup B_2)$; /* $|B_2| = |W_2|$ or $|B_2| = |W_2| + 1$ */

Find an edge set E'_2 which is a solution found by $Sol_(\sigma+1)$ECAB[14] under the situation that a structural graph is $F(G)$ and a black (white, respectively) leaf set in $F(G)$ is regarded as B_2 (W_2).

Output $E'_1 \cup E'_2$;

4.2 Correctness of the Algorithm

We prove correctness of the algorithm using several lemmas and a theorem.

First, we show the next lemma for a structural graph $F(G)$ of a graph G which may be not with multipartition

Lemma 4.1 (14)) Suppose that $|LF(G)| \geq 4$ for a structural graph $F(G)$. Now, if there are distinct four leaves v, w, x, y with $\beta(v) < \beta(x) < \beta(w) < \beta(y)$ then it can be chosen four vertex $n_v, n_w, n_x, n_y \in V(G)$ such that the number of leaves of $F(G + \{(n_v, n_w)\})$ and $F(G + \{(n_x, n_y)\})$ are two less than that of leaves of $F(G)$, where for $a \in \{v, w, x, y\}$ n_a is any vertex in $\rho^{-1}(a)$.

In Lemma 4.1 a pair of v and w (or a pair of x and y) is called an augmenting pair with

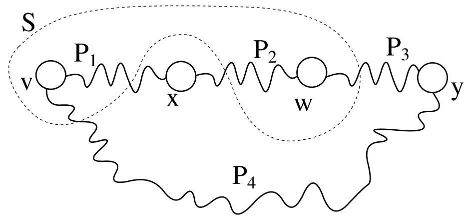

Fig. 4 Schematic explanation of Lemma 4.1

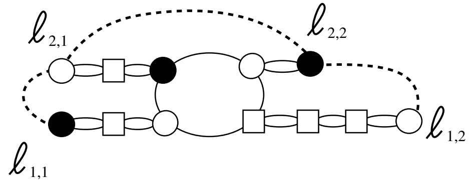

Fig. 5 Schematic explanation of Proposition 3.1 (1), where dash lines represent a solution, $\ell_{i,j}$: i is the number of color C_i, j is the number of vertices in same color vertices.

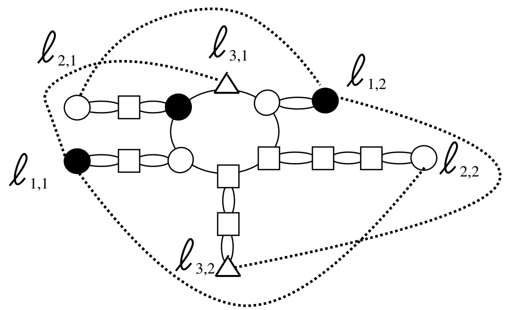

Fig. 6 Schematic explanation of Proposition 3.1 (2), where dash lines represent a solution, $\ell_{i,j}$: i is the number of color C_i, j is the number of vertices in same color vertices.

respect to v, w, x and y.

In the next lemma, we show a special case of finding an edge set which is considered Proposition 3.1 (2) and (3).

Lemma 4.2 (i) (1)) If $|L_1(G)| = 2$ then we consider a graph G_c defined in Proposition 3.1 (2). G_c is a simple cycle whose length is six and in which two C_1 vertices, two C_2 ones and two C_3 ones appear alternately (see Fig. 6) then there is a solution E_f with $|E_f| = 4$ to $F(G)$; (ii) (14)) Otherwise, C_3 vertices are treated as one C_1 vertex and one C_2 vertex, and we obtain a solution E_f with $|E_f| = 3$ by resulting to $(\sigma + 1)$ECAB.

We consider Steps 12 and 18 of FIND_EDGES in order to reduce $(\sigma + 1)$ECAM to $(\sigma + 1)$ECAB.

Lemma 4.3 (14)) For any connected graph G with $\lambda(G) = \sigma$ and any bipartition $\{V_1, V_2\}$ of V with $V_1 \cap V_2 = \emptyset$. Suppose that $\forall i \in \{1, 2, 3\}$, $|L_1F(G) \cup L_2F(G)| = 2i$ and $|L_1F(G)| = |L_2F(G)|$, then we obtainan edge set E''_1 such that $|LF(G + E''_1)| = |LF(G)| - 2|E''_1|$ $(1 \leq |E''_1| \leq |L_1F(G)|)$

Lemma 4.4 (14)) For any connected graph G with $\lambda(G) = \sigma$ and any bipartition $\{V_1, V_2\}$ of V with $V_1 \cap V_2 = \emptyset$. Suppose that $|L_1F(G) \cup L_2F(G)| \geq 5$ and $L_2F(G) \neq \emptyset$, then we can choice a C_1 leaf ℓ_1 and a C_2 leaf ℓ_2 of $F(G)$ such that the number of leaves of $F(G + \{(n_{\ell_1}, n_{\ell_2})\}$ is two less than that of $F(G)$, where n_{ℓ_1} (n_{ℓ_2}, respectively) is a C_1 vertex (a C_2 vertex) in $\rho^{-1}(\ell_1)$ $(\rho^{-1}(\ell_2))$.

From Lemmas 4.3 and 4.4, we obtain the next corollary.

Corollary 4.1 For any connected graph G with $\lambda(G) = \sigma$ and any bipartition $\{V_1, V_2\}$ of V with $V_1 \cap V_2 = \emptyset$. Suppose that $|L_1F(G)| = |L_2F(G)|$, then we obtain an edge set E'_1 such that $|LF(G + E'_1)| = |LF(G)| - 2|E'_1|$ $(1 \leq |E'_1| \leq |L_1F(G)|)$ by adapting Lemma 4.4 iteratively, or Lemma 4.3.

From Lemmas 4.1–Corollary 4.1, we obtain the next theorem.

Theorem 4.1 For any connected graph G with $\lambda(G) = \sigma$, *Sol_*$(\sigma + 1)$*ECAM* finds an edge set E_f with $|E_f| = \max_{i=1}^{r}\{|L_iF(G)|, \lceil |LF(G)|/2 \rceil\}$ and $\lambda(G + E_f) = \sigma + 1$.

(Proof) We consider an edge set found in Steps 12, 18 of FIND_EDGES and Step 14 of *Sol_*$(\sigma + 1)$ECAM.

We discuss the following cases whether $F(G)$ is C_1-dominated or not. Case (i): $F(G)$ is C_1-dominated. Step 15 of *Sol_*$(\sigma + 1)$ECAM finds $|L_1F(G)|$ edges, thus $|E_f| = |L_1F(G)|$. Case (ii): $F(G)$ is not C_1-dominated.

We classify Case (ii) into two cases as follows: Case (ii-i) $|L_1| = |L_2| = |L_3| = 2$ and $\sum_{i=4}^{r} |L_i| = 0$ then we obtain $|E_f| = 4$ ($|E_f| = 3$) edges by Lemma 4.2 (1)((2), respectively).

Case (ii-ii) otherwise Since $|L_{j_h}F(G)| \geq a + 1$ and $|L_1F(G)| \geq a + 1$ hold, we can find two sets W_1 and B_1. Step 14 of *Sol_*$(\sigma + 1)$ECAM is not executed.

Let E'_1 (with $|E'_1| = a + 1$) be an edge set found in Step 12 of FIND_EDGES(The first phase). Since an edge e is deleted in Step 13 of FIND_EDGES, $|E'_1| = a$ holds and $F(G) + E'_1$ has $LF(G) - 2|E'_1|$ leaves.

Moreover, it is not generated a new leaf in $F(G) + E'_1$ by adding an edge set because a pair of endvertices u and v of a deleted edge $e = (u, v)$ can be considered as an augmenting pair in Lemma 4.1.

Let E'_2 (with $|E'_2| = |B_2|$) be an edge set found in Step 18 of FIND_EDGES. Moreover, $|E'_2| = \lceil |LF(G)|/2 \rceil - |E'_1|$, $|E'| = |E'_1 \cup E'_2| = \lceil |LF(G)|/2 \rceil$ hold and it is not generated a new leaf in $F(G) + E'_1 + E'_2$ (deleting all σ-cuts in $F(G)$).

Thus, $|E_f| = \max\{|L_1F(G)|, \ldots, |L_rF(G)|, \lceil |LF(G)|/2 \rceil\}$. Since E'_2 is a solution to $(\sigma + 1)$ECAB for a graph $F(G) + E'_1$, E_f is a edge set with $\lambda(G + E_f) = \sigma + 1$. □

4.3 Time Complexity

In this section, we discuss time complexity of the proposed algorithm.

The above operation is done in $O(|V|)$ time and can find all augmenting pairs by *Sol_*$(\sigma + 1)$ECAB[14], Lemma 4.2. A structural graph is constructed in $O(|V||E| + |V|^2 \log |V|)$ time[12]. Moreover, in the case of $\lambda(G) \in \{1, 2\}$, a structural graph is constructed in linear time because all $(\sigma + 1)$-components are computed in linear time[10),15),16),18]. From the above discussion, Proposition 3.1 and Theorem 4.1, We obtain the next theorem.

Theorem 4.2 Algorithm *Sol_*$(\sigma + 1)$*ECAM* computes a solution for $(\sigma + 1)$ECAM when $\sigma = \lambda(G)$ in $O(|V||E| + |V|^2 \log |V|)$ time. Moreover, it does in $O(|V| + |E|)$ time when $\sigma \in \{1, 2\}$.

5. Parallelization

In this section, we propose a parallel algorithm for $(\sigma + 1)$ECAM with $r \geq 2$, when a structural graph $F(G)$ is given and σ is odd by reducing to 2ECAB.

2) proposed also an linear time algorithm for 2ECAM, and a parallel algorithm on an EREW PRAM. However, our approach is different from 2). Moreover, we augment edge-connectivity by one in tje same approach even if $\sigma > 1$ and σ is odd. Thus, our approach is more general.

FIND_EDGES is done in $O(\log |V|)$ parallel time with $O(|V|)$ processors by two modifications.

First, Steps 1–8 of FIND_EDGES replacing into the Steps 1–22 of the following procedure. Note that each of merging partitions and decomposing partitions is done in $O(\log |V|)$ parallel time with r processors.

Set $p_i \leftarrow |L_i|$ for any $i \in \{1, \ldots, r\}$. /* p_i is stored in a shared memory on PRAM. */
/* See Fig. 7. */
$exp\_i = 2$; /* exp_i is used for calculating 2^i. */
for $i \leftarrow 1$; $i \leq \log r$; $i++$ **do**
  for $j \leftarrow 1$; $j \leq r/exp\_i$; $j++$ **do**
    $n \leftarrow (2j-1) \cdot (exp\_i/2)$, $m \leftarrow j \cdot exp\_i$, $p_m \leftarrow p_n + p_m$.
    /* This step is executed on each processor m in parallel */
  end for
  if this processor's number is exp_i $(= 2^i)$ and $p_{exp\_i} \geq \lceil |LF(G)|/2 \rceil$ **then**
    $i_h \leftarrow i$, $s_h \leftarrow p_{exp\_i}$, break;
  end if
  $exp\_i \leftarrow 2 \cdot exp\_i$; /* $exp\_i = 2^{i+1}$ */
end for
/* Executing on processor 1 */
$j_h \leftarrow 1$; /* $exp\_i = 2^{i_h}$ */;
for $i \leftarrow i_h$; $i \geq 1$; $i--$ **do**
  $n \leftarrow (2j_h - 1) \cdot (exp\_i/2)$, $m \leftarrow j_h \cdot exp\_i$, $exp\_i \leftarrow exp\_i/2$; /* $n = (2j_h - 1) \cdot 2^{i-1}$, $m = j_h \cdot 2^i$ */
  $p_m \leftarrow p_m - p_n$, $s'_h \leftarrow s_h - p_m$;
  if $s'_h < \lceil |LF(G)|/2 \rceil$ **then**
    $j_h \leftarrow 2j_h$, $s_h \leftarrow s_h$;
  else
    $j_h \leftarrow 2j_h - 1$, $s_h \leftarrow s'_h$;
  end if
end for

Next, we replace $(\sigma + 1)$ECAB of FIND_EDGES into 2ECAB, and, in 2ECAB, using a cactus as a structural graph (not a modified cactus). Because a cactus is a tree when σ is odd, we can use the existing parallel algorithm for 2ECAB to eliminate all σ-cuts by the algorithm. We show a theorem and a corollary to describe reduction to 2ECAB.

Theorem 5.1 (8)) We can obtain an optimum solution to B-2ECAB in sequential linear time and $O(\log |V|)$ parallel time on an EREW PRAM using a linear number of processors.

The algorithm proposed in 8) can be kept not only bipartiteness of a bipartite-graph but also bipartition constraints of a graph for adding edges. Thus we obtain the following corollary from Theorem 5.1.

Corollary 5.1 We can obtain an optimum solution to 2ECAB in sequential linear time and $O(\log |V|)$ parallel time on an EREW PRAM using a linear number of processors.

From the above discussion, Proposition 3.1 and Theorem 4.1 we obtain the next theorem.

Theorem 5.2 Algorithm *Sol_$(\sigma + 1)$ECAM_Parallel* computes a solution to $(\sigma + 1)$ECAM for any σ-edge-connected graph, when a structural graph $F(G)$ is given, in $O(\log |V|)$ parallel time on an EREW PRAM using a linear number of processors.

Moreover, we consider a parallel algorithm for 2ECAM with $r \geq 2$ for any graphs by reducing to 2ECAB. In 2ECAM, Step 3 of *Sol_*$(\sigma + 1)$ECAM does not execute. We add an edge set to a shirinked 2-compoenent graph instead of a structural graph. The set of C_i *isolated vertices* or *hybrid isolated vertices*, respectively, is denoted by $L_i^* F(G)$ or $H^* F(G)$. A lower bound of on any solution to 2ECAM for any graphs is given the following proposition.

Proposition 5.1 (2)) The number of edges required to 2-edge-connect a graph G is given $\max\{\max_{i=1}^{r}\{|L_i F(G)| + 2|L_i^* F(G)|\}, \lceil (2\sum_{i=1}^{r} |L_i^* F(G)| + \sum_{i=1}^{r} |L_i F(G)|)/2 \rceil\}$

A structural graph is constructed in $O(\log |V|)$ parallel time on an EREW PRAM with linear number processors[3),17)].

From the above discussion and Theorem 5.2, we obtain the next corollary.

Corollary 5.2 Algorithm *Sol_2ECAM_Parallel* computes a solution to 2ECAM for any graphs in $O(\log |V|)$ parallel time on an EREW PRAM using a linear number of processors.

6. Concluding Remarks

In this paper, we propose a fast algorithm for finding a solution to $(\sigma+1)$ECAM when $\sigma = \lambda(G)$ in $O(|V||E| + |V|^2 \log |V|)$ and show that the problem can be solved in linear time if $\sigma \in \{1, 2\}$. Moreover, we propose a parallel algorithm for finding a solution to $(\sigma + 1)$ECAM, when a structural graph $F(G)$ is given and σ is odd in $O(\log |V|)$ parallel time on an EREW PRAM using a linear number of processors, and also show that 2ECAM for any graphs can be solved in linear time.

As future research, proposing an efficient algorithm for $(\sigma + \delta)$ECAM when $\sigma = \lambda(G)$ and $\delta > 1$ is left.

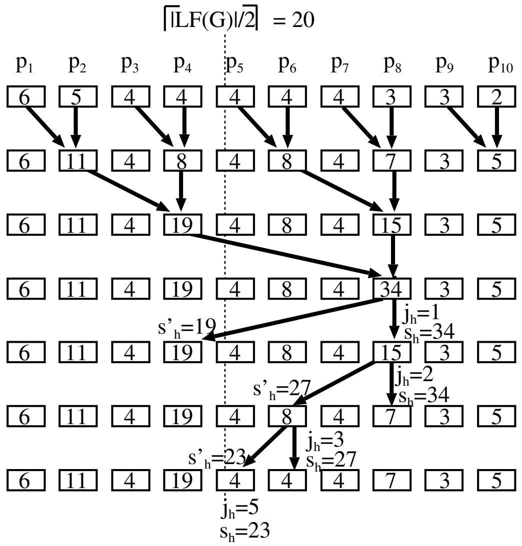

Fig. 7 Schematic explanation of Steps 1–8 in FIND_EDGES

Acknowledgements

The research is partly supported by the Grant-in-Aid for Scientific Research (C) (No. 20500015 and 22500029) of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

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.W. Chong, Y.Han, and T.W. Lam. Concurrent threads and optimal parallel minimum spanning trees algorithm. *J. ACM*, 48(2):297–323, 2001.
4) K.P. Eswaran and R.E. Tarjan. Augmentation problems. *SIAM J. Comput.*, 5:653–655, 1976.
5) D.S. Hirschberg. Fast parallel sorting algorithms. *Commun. ACM*, 21(8):657–661, 1978.
6) T.Hsu and M.Kao. Optimal augmentation for bipartite componentwise biconnectivity in linear time. *SIAM J. Comput*, 32(6):1493–1515, 2005.
7) P.Huang, H.Wei, Y.Chen, M.Kao, W.Shih, and T.Hsu. Two-vertex connectivity augmentations for graphs with a partition constraint (extended abstract). In Y.Dong, D.Du, and O.H. Ibarra, editors, *ISAAC*, volume 5878 of *Lecture Notes in Computer Science*, pages 1195–1204. Springer, 2009.
8) P.Huang, H.Wei, W.Lu, W.Shih, and T.Hsu. Smallest bipartite bridge-connectivity augmentation. *Algorithmica*, 54(3):353–378, 2009.
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), 8–12.
10) 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.
11) H.Nagamochi, S.Nakamura, and T.Ibaraki. A simplified $\tilde{O}(nm)$ time edge-splitting algorithm in undirected graphs. *Algorithmica*, 26:50–57, 2000.
12) 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.
13) 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.
14) T.Oki, S.Taoka, T.Mashima, and T.Watanabe. A fast algorithm for $(\sigma+1)$-edge-connectivity augmentation of a σ-edge connected bipartite graph. In *Proceeding of the 23th Karuizawa Workshop on Circuits and Systems*, pages 404–409, 2010.
15) 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.
16) R.E. Tarjan. A note on finding the bridges of a graph. *Information Processing Letters*, 2:160–161, 1974.
17) R.E. Tarjan and U.Vishkin. An efficient parallel biconnectivity algorithm. *SIAM J. Comput*, 14(4):862–874, 1985.
18) Y.H. Tsin. Yet another optimal algorithm for 3-edge-connectivity. *J. Discrete Algorithms*, 7(1):130–146, 2009.
19) T.Watanabe and A.Nakamura. Edge-connectivity augmentation problems. *Journal of Computer and System Sciences*, 35(1):96–144, 1987.