アルゴリズム 40−4
(1994. 7. 22)

# グラフのk辺連結化問題に対する新しい近似解法

間島利也　渡辺敏正
広島大学工学部 第二類 回路システム工学講座
(724) 東広島市鏡山一丁目4-1
(電話) 0824-24-7662 (ファクシミリ) 0824-22-7195
(電子メール) watanabe@huis.hiroshima-u.ac.jp

**概要**: 本論文ではコスト付きのk辺連結化問題(W-kECA)に対する近似解法について報告する. 我我は既にW-kECAに対して4つの近似解法を提案し, 比較実験では最大コストマッチングを用いた近似解法FSMが最良であることを示した. ここでは, 次の2つの結果を与える. 第一に, W-kECAに対する新しい近似解法MWを提案し, 辺連結度を1だけ上げるW-kECAに対してMWの近似解は最適解の2倍以下で抑えられることを理論的に証明する. 第二に, 近似解を改良するための辺の交換手法EXCHANGEを既存の4つの近似解法とMWに組み込み, その有効性を実験的に明示する. さらにEXCHANGEの組み込み後でも, これらの5つの近似解法の中でFSMがやはり最良の近似解を与えることも実験的に示す.

# New Approximation Algorithms for the k-Edge-Connectivity Augmentation Problem

Toshiya Mashima 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) Fax: +81-824-22-7195
E-mail: watanabe@huis.hiroshima-u.ac.jp

**Abstract**: New approximation algorithms for the weighted k-edge-connectivity augmentation problem (W-kECA) are considered. We have already proposed four approximation algorithms for W-kECA, and have shown experimentally that the approximation algorithm FSM using maximum-cost matching algorithm gives the best approximation among them. In this paper, two results are given. First, a new approximation algorithm MW is proposed, and it is proved theoretically that MW produces approximation whose total cost can be bounded by twice the optimum for the case where the edge connectivity is increased by one. Secondly a simple postprocessing for improvement of a solution is incorporated into any one of the previous four algorithms as well as MW, and it is experimentally shown that total cost of each solution is greatly reduced. It is also presented that, even after such improvement, FSM remains giving the best approximation among these five algorithms.

## 1. Introduction

The *k-edge-connectivity augmentation problem* (kECA for short) is defined by "Given a graph G'=(V,E'), a cost function $c:V\times V\rightarrow Z^+$ and a positive integer k, find a minimum-cost set E" of edges, each connecting distinct vertices of V, such that the edge-connectivity of G"=(V,E'∪E") is k, where $V\times V=\{\{u,v\}|u,v\in V,u\neq v\}$, $Z^+$ is the set of nonnegative integers and the *edge-connectivity* of a graph G, ec(G), is the minimum number of edges whose deletion disconnect it. "

The problem is called the weighted version, denoted by W-kECA, if there may exist some distinct costs and the unweighted one, denoted by UW-kECA, otherwise. Costs c({u,v}) for $\{u,v\}\in V\times V$ is denoted as c(u,v) for simplicity, and we assume c(u,v)=1 for any $\{u,v\}\in V\times V$ in UW-kECA. Let kECA(*,**) denote 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 * means that G' may be a multiple graph; (ii) ** is set to SA if increase in edge-multiplicity in constructing G" is prohibited, and is set to MA otherwise. Let $\lambda$=ec(G') and $k=\lambda+\delta$ with $\delta\geq 0$.

Results related to W-kECA are very briefly mentioned. For the NP-completeness of W-kECA, see [1,4] for k=2, [20] for E'=∅ and $k=\lambda+\delta$ with $\delta\geq 1$, [21] for k=3, and [12] for $k=\lambda+\delta$ with $\delta=1$. For UW-kECA, see [3,6,15,20,22]. Concerning approximation algorithms for W-kECA, see [4] for k=2, [21] for k=3, and [11,12,13,18,19] for general k. [11] showed that, for W-kECA, there is an algorithm that produces worst approximation bounded by twice the optimum. Its time complexity is $O(k|V|(|E'|+\delta|V|^2)\log|V|)$. Their algorithm is based on an algorithm for finding k edge-disjoint arborescences of minimum total cost [5], and it does not seem that its implementation is easy. In [12,13,18,19], four approximation algorithms FSA, FSM, SMC and HBD for W-kECA were proposed, and they were evaluated theoretically and experimentally. Among them, FSM based on minimum-cost matchings experimentally gives the best approximation, even though it produces unbounded approximation for some inputs.

The paper has two subjects. First, a new approximation algorithm MW for W-kECA is proposed, and it is proved theoretically that MW produces approximation whose total cost can be bounded by twice the optimum for the case where the edge connectivity is increased by one. Secondly a simple postprocessing for replacing edges of a solution is incorporated into any one of the previous four algorithms as well as MW, and it is experimentally shown that total cost of each solution is greatly reduced. It is also presented that, even after such modification, FSM remains giving the best approximation among these five algorithms.

Approximation algorithms considered in this paper can be applied to both W-kECA(*,MA) and W-kECA(*,SA), and they are also useful to obtain approximate solutions for UW-kECA(*,SA) which is an open problem (see [16]). W(UW)-kECA(*,SA) can be handled if we modify a cost function c such that c(u,v)=∞ for any $\{u,v\}\in V\times V$ with $(u,v)\in E'$.

## 2. Preliminaries

### 2.1. Basic definitions

Technical terms not given here can be identified in [2,8,17]. An *undirected graph* G=(V(G),E(G)) or G=(V,E) (*directed graph* $\vec{G}=(V(\vec{G}),A(\vec{G})$ or $\vec{G}$=(V,A)), respectively) consists of a finite nonempty set V(G) ($V(\vec{G})$) of vertices and a finite set E(G) ($A(\vec{G})$) of undirected (directed) edges. An undirected edge e incident upon two vertices u, v in G is denoted by (u,v). A directed edge $\vec{e}$ emanating from u and entering v is denoted by <u,v>. In this paper, the term "a graph" means an undirected or a directed multigraph without self-loops unless otherwise stated.

A path P between u and v, or a (u,v)-path, in G is denoted as $P_G(u,v)$ or P(u,v). A pair of multiple edges are considered as a cycle of length two. For two vertices u, v of G, let M(u,v;G), or simply M(u,v), denote the maximum number of pairwise edge-disjoint paths between u and v. For a directed graph $\vec{G}=(V(\vec{G}),A(\vec{G}))$, we use similar notations.

For a nonempty vertex set $S\subset V(G)$, we denote $C(S,\bar{S})=\{(u,v)\in E(G)|u\in S \text{ and } v\in\bar{S}\}$, where $\bar{S}$=V(G)-S, it is called a *cut* in G. A cut whose cardinality is k is often called a *k-cut*. A cut $C(S,\bar{S})$ is called a *minimum cut* if its cardinality is minimum among all cuts of G. The *edge-connectivity* (denoted by ec(G)) of G is the number of edges in a minimum cut of G. G is said to be *k-edge-connected* if ec(G)≥k. A *k-edge-connected component* (k-ecc for short) of G is a subset $S\subseteq V(G)$ satisfying (a) and (b): (a) M(u,v;G)≥k for any pair $u,v\in S$; (b) S is a maximal set that satisfies (a). Note that distinct k-eccs are disjoint. It is known that ec(G)≥k if and only if V(G) is a k-ecc. A 1-ecc is simply called a *component*. A *degree* of a vertex v in G is the number of edges incident upon v in G, and it is denoted by $d_G(v)$ or simply d(v). v is often called a *degree-d(v) vertex*.

For a set $R\subseteq V(G)\cup E(G)$, let G[R] denote the subgraph having $R\cap V(G)$ as its vertex set and $\{(u,v)\in E(G)| u,v\in R\cap V(G) \text{ and } (u,v)\in R\cap E(G)\}$ as its edge set. G[R] is called the *subgraph* of G *induced* by R (or the *induced subgraph* of G by R). *Deletion* of $R\subseteq V(G)\cup E(G)$ from G is to construct G[V(G)∪E(G)-R], which is often denoted as G-R. For a set E' of edges such that E'∩E(G)=∅, let G+E' denote the graph (V(G),E(G)∪E'). Throughout the paper, we often write a singleton set {x} simply as x.

A *cactus* is an undirected connected graph in which any pair of cycles share at most one vertex: each shared vertex is a cutpoint. An edge in a cactus is called a *cycle edge* if it is contained in a cycle; otherwise it is called a *tree edge*. A *leaf* of a cactus is either a vertex v with d(v)=1 or v' included in a cycle with d(v')=2. An *arborescence* is a directed acyclic graph with one specified vertex, called the *root*, having no entering edges, and all other vertices having exactly one entering edge. For a vertex v in a rooted tree T, let $T_V$ denote the subgraph of T induced by all descendants of v (including v itself), where we virtually direct all edges from the root towards leaves, and vertices reachable from a vertex v are called *descendants* of v; *ancestors* of v are those vertices from which v is reachable. For any pair {u, v} of vertices in a rooted tree T, let $LCA_T(u,v)$ or LCA(u,v) denote the least (or nearest) common ancestor of u and v.

Given a set T and a mapping $c:T \rightarrow Z^+$, we use a notation $c(T') = \sum_{x \in T'} c(x)$ for $T' \subseteq T$.

### 2.2. Structural graphs

A *structural graph* F(G) [10] of a undirected multigraph G=(V,E) with edge-connectivity ec(G)=λ is a representation for all minimum cuts of G. F(G) is an edge-weighted cactus of $O(|V|)$ nodes and edges such that each tree edge has weight λ and each cycle edge has weight λ/2. Particularly if λ is odd then F(G) is a edge-weighted tree. Each vertex in G maps to exactly one vertex in F(G), and F(G) may have some other vertices, called *empty vertices*, to which no vertices of G are mapped. Let $\varepsilon(G) \subseteq V(F(G))$ denote the set of all empty vertices 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 $\rho:V(G) \rightarrow V(F(G))-\varepsilon(G)$ 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 $Y \subseteq V(F(G))$.

It is shown that F(G) can be constructed in $O(|V||E|)$ time [10] or $O(|E|+\lambda^2|V|\log(|V|))$ time [6]. Note that if λ is even then replacing each tree edge by a pair of multiple edges preserves the properties of structural graphs and makes their handling easy because the resulting graphs have no bridges. This graph and a tree in the case where λ is odd are called *modified cactuses*. In this paper, F(G) denotes a modified cactus unless otherwise stated. Let Z be a set of edges connecting vertices of F(G) and such that $Z \cap E(F(G))=\emptyset$. Suppose that $ec(F(G)+Z) \geq 2$ if λ is odd or $ec(F(G)+Z) \geq 3$ if λ is even. Then we call Z a *solution* to F(G).

## 3. Approximation Algorithms

We first describe a general scheme, as an algorithm REC, for finding an approximate solution E" to W-kECA. REC repeatedly finds approximate solution for W-(λ+1)ECA. REC first constructs a structural graph F(G'), also denoted by $G_s'=(V_s,E_s')$ (see Fig.1(1),(2)), and computes a cost function $c_s:V_s \times V_s \rightarrow Z^+ \cup \{\infty\}$ and a backpointer $b_s:V_s \times V_s \rightarrow V \times V$ defined as follows:

$$c_s(u,v)=\min\{\{\infty\} \cup \{c(x,y) | x \in \rho^{-1}(u) \text{ and } y \in \rho^{-1}(v)\}\};$$

$$b_s(u,v)=\begin{cases}(x,y) & \text{if } c_s(u,v)=c(x,y) \text{ with } x \in \rho^{-1}(u) \text{ and } y \in \rho^{-1}(v)\\ \emptyset & \text{if } c_s(u,v)=\infty\end{cases}$$

where $\rho^{-1}(w)$ is a (λ+1)-ecc of G' and is represented as $w \in V_s-\varepsilon(G')$ in F(G).

The following lemma shows that it suffices to consider W-(λ+1)ECA for a cactus $G_s'=(V_s,E_s')$ (instead of G').

**Lemma 1.** If $ec(G'+E") \geq \lambda+1$ for some E" then there is $E_s"$ with $c_s(E_s") \leq c(E")$ such that $E_s"$ is a solution to $G_s'$. For a set $E_s"$, let $b_s(E_s")=\{b_s(u,v) | (u,v) \in E_s"\}$ with multiplicity deleted. If $E_s"$ is a solution to $G_s'$ then $ec(G'+b_s(E_s")) \geq \lambda+1$ and $c_s(E_s") \geq c(b_s(E_s"))$. ♦

The algorithm REC for W-kECA is described as follows.

**Algorithm REC;**
/* Input: a graph G'=(V,E'), a cost function $c:V \times V \rightarrow Z^+$ and a positive integer k */
/* Output: An approximate solution E" to W-kECA */
**begin**
1. H'←G'; E"←∅;
2. Construct a structural graph $G_s'=(V_s,E_s')(=F(H'))$ of H'; θ←ec(H');
3. **if** θ≥k **then goto** Step 9;
4. Compute a cost function $c_s$ and a backpointer $b_s$;
5. Find an edge set $F_s$ of small total cost $c_s(F_s)$ such that $F_s$ is a solution to $G_s'$;
6. Construct another solution $F_s'$ from $F_s$ by a postprocessing;
7. $\Gamma \leftarrow \{b(u,v) \in V \times V | (u,v) \in F_s'\}$ (with multiplicity of edges deleted);
8. H'←H'+Γ; E"←E"∪Γ; **goto** Step 2;
9. **end**.

Constructing $G_s'$, $c_s$ and $b_s$ can be done in $O(\Delta+|V|^2)$ time, where Δ is the time complexity of finding a structural graph of G' at Step 2 in the algorithm REC and $\Delta=\min\{|V||E'|, |E'|+\lambda^2|V|\log(|V|)\}$ [6,10]. If Step 5 and Step 6 can be done in $O(\xi)$ time then REC for W-(λ+1)ECA runs in $O(\Delta+|V|^2+\xi)$ time.

Thus the remaining task is to devise an efficient algorithm producing a good approximate solution $F_s$ to $G_s'$ in Step 5 of REC for the case where ec(H')<k. In [12,13,18,19], four procedures FSA*, FSM*, SMC* and HBD* were proposed for Step 5 of REC. (In these papers, the algorithm EC, which is REC without Step 6, was considered, and EC with FSA*, for example, was denoted as FSA, and similarly for others.) In this paper, we propose another procedure MW* for Step 5 of REC, as well as a postprocessing for improving a solution $F_s$ at Step 6 of REC. They will be explained in 3.1 and 3.2, respectively. REC with MW* is denoted as RMW, and similarly for other procedures.

### 3.1. MW*

We describe the procedure MW* based on a minimum-cost arborescence algorithm [7]. MW* use the following procedure REMAKE that changes a modified cactus $G_s'$ (=F(G')) into a spanning tree $G_d'$ by adding some dummy vertices (see Fig.1(2),(3)). Notice that if ec(G') is odd then $G_s'$ is a tree.

**procedure REMAKE;**
/* Input: A structural graph $G_s'=(V_s,E_s')$ of H', $c_s$ and $b_s$ */
/* Output: A tree $G_d'=(V_d,E_d')$, $c_d$ and $b_d$ */
**begin**
1. $V_d \leftarrow V_s$; $E_d' \leftarrow E_s'$; $W \leftarrow \emptyset$;
 **for** every pair $\{u, v\} \in V_s \times V_s$ **do**
 **begin**
 $c_d(u, v) \leftarrow c_s(u, v)$;
 $b_d(u, v) \leftarrow b_s(u, v)$
 **end;**
2. **if** ec(H') is odd **then goto** 7;
3. Delete multiplicity of edges in $E_d'$;
4. Find all cycles in $G_d'$ by a depth-first-search;
5. **for** each cycle C **do**
 **begin**
 $V_d \leftarrow V_d \cup w_C$; /* $w_C$ is a dummy vertex for C. */
 $W \leftarrow W \cup \{w_C\}$;
 $E_d' \leftarrow E_d' \cup \{(u,w_C) \mid u$ is a vertex on C$\}$;
 $E_d' \leftarrow E_d' - \{e \mid e$ is an edge on C$\}$
 **end;**
6. **for** each dummy vertex $w_C \in W$ **do**
 **for** each vertex u in $V_d$ **do**
 **begin** $c_d(w_C,u) \leftarrow -\infty$; $b_d(w_C,u) \leftarrow \emptyset$ **end;**
7. **end;**

In the procedure MW*, we choose a leaf r of a cactus $G_s'=(V_s,E_s')$ and we call it the root of $G_s'$ (or of $G_d'$). After a tree $G_d'=(V_d,E_d')$ is constructed from $G_s'$ by REMAKE, we execute the breadth first search (BFS) starting from r and define son-parent and descendant-ancestor relations on $G_d'$: if v is visited from u and $(u,v) \in E_d'$ then v is a son of u and u is the parent of v, and if v' is reached by BFS starting from u' then v' is a descendant of u' and u' is an ancestor of v'. Suppose $G_s'$ has at least one cycle whose length is more than 2. For each cycle C of $G_s'$, let $s(C) \in V(C)$ denote the vertex which is nearest to the root r, and s(C) is called the starting vertex of C. (Note that s(C) is the first visited vertex in C when $G_s'$ is searched by the breadth-first search starting from r.) We number the vertices of C as $v_C^0(=s(C))$, $v_C^1, v_C^2, \ldots, v_C^{l(C)-1}$ clockwise, where $l(C)$ denotes the length of C. Let $L_C(j,k)$ denote the set of vertices $v_C^j$ through $v_C^k$ clockwise on C for $j \leq k$. For any vertex $u \notin V(C)$, let $A_C(j,k;u)=\{<v,u> \mid v \in L_C(j,k)\}$. Note that if $0<j\leq k<l(C)$ then $v_C^0 \notin L_C(j,k)$. Let u, v be vertices of $V_d$ such that u is neither an ancestor nor a descendant of v. Consider the situation that $t=LCA_{G_d'}(u,v) \in W=V_d-V_s$, that is, t is a dummy vertex and $t=w_C$ for a cycle C of $G_s'$. Let u(C) and v(C) denote the vertices of C such that the (u,t)-path and the (v,t)-path of $G_d'$ pass though, respectively. For notational simplicity, we assume that u(C) appears before v(C) if we start from s(C) and go around C clockwise. If $u(C)=v_C^j$ and $v(C)=v_C^k$ with $j\leq k$ then $L_C(j,k)$ is sometimes denoted as $L_C(u(C), v(C))$, and we denote

$A_C(u,v)=A_C(j,k;u) \cup A_C(j,k;v)$ (see Fig.2).

**procedure MW*;**
/* Input: $G_s'=(V_s,E_s')$, $c_s$ and $b_s$ */
/* Output: A set $F_s$ of edges, each connecting distinct vertices of $V_s$, such that $F_s$ is a solution to $G_s'$ */
**begin**
1. Choose any leaf r of $G_s'$ as the root;
2. Construct a tree $T=G_d'=(V_d,E_d')$, $c_d$, $b_d$ from $G_s'$, $c_s$, $b_s$ by REMAKE;
3. Construct a directed tree $\vec{T}=\vec{G}_d'=(V_d,A_d')$ from T by directing each edge of $E_d'$ toward r (see Fig.1(4));
 /* Construction of a directed graph $\vec{G}_d=(V_d,A_d)$, a cost function $\vec{c}_d:A_d \rightarrow Z^+ \cup \{\infty\} \cup \{0\}$ and backpionters $\vec{b}_d:A_d \rightarrow CAND$. */
4. $A_d \leftarrow A_d'$;
 **for** each $\vec{e} \in A_d$ **do begin** $\vec{c}_d(\vec{e}) \leftarrow 0$; $\vec{b}_d(\vec{e}) \leftarrow \emptyset$ **end;**
 Construct a complete graph $G_s=(V_s,E_s)$ with $E_s \cap E_s'=\emptyset$;
 $CAND \leftarrow \{(u,v) \in E_s \mid \{u,v\} \in V_s \times V_s, c_s(u,v) \neq \infty\}$;
5. **for** each edge $e \in (u,v) \in CAND$ **do**
 **begin** /* constructing the set CD(e) of directed edges */
 5.1. **if** one of {u,v}, say u, is an ancestor of the other, v, in T **then** $CD(e) \leftarrow \{<u,v>\}$
 5.2. **else if** u is neither an ancestor nor a descendant of v in T **then**
 **begin**
 $t \leftarrow LCA_T(u,v)$;
 5.2.1. **if** $t \notin W$ **then** $CD(e) \leftarrow \{<t,u>, <t,v>\}$
 /* W is the set of dummy vertices in T */
 5.2.2. **else if** $t=w_C \in W$ **then** $CD(e) \leftarrow A_C(u,v)$;
 **end;**
 5.3. $A_d \leftarrow A_d \cup CD(e)$;
 **for** each $\vec{f} \in CD(e)$ **do**
 **begin** $\vec{c}_d(\vec{f}) \leftarrow c_d(e)$; $\vec{b}_d(\vec{f}) \leftarrow e$ **end**
 **end;**
6. Find a minimum-cost arborescence $\vec{T}_m=(V_d,A_m)$, $A_m \subseteq A_d$, rooted at r, with respect to the cost function $\vec{c}_d$, in $\vec{G}_d$ (Fig.3);
7. $F_s \leftarrow \emptyset$;
 **for** each $\vec{e} \in A_m$ with $\vec{c}_d(\vec{e})>0$ **do** $F_s \leftarrow F_s \cup \{\vec{b}_d(\vec{e})\}$;
 Delete multiplicity of edges in $F_s$ (Fig.4);
 **end;**

Next, we will prove the correctness of the procedure MW*. From now on, we set $\theta=2$ if $\lambda$ is odd, and $\theta=3$ if $\lambda$ is even.

Construction of $\vec{T}(=\vec{G}_d')$ and $\vec{b}_d$ implies the following proposition.

**Proposition 1.** (1) If u is an ancestor of v on $T(=G_d')$ then u is a reachable from v in $\vec{T}$.

(2) No two dummy vertices in $\vec{T}$ are adjacent to each other.

(3) If there is a directed edge $<x,y> \in A_d$ in $\vec{G}_d$ and $(x',y')=\vec{b}_d(<x,y>)$ then $x'=y$ or $y'=y$. ♦

From Proposition 1(3), we can assume $\vec{b}_d(<x,y>)=(x',y)$ without loss of generality.

We define two vertex sets $S \subseteq V_s$ and $D(S) \subseteq V_d$ for any minimum cut K in $G_s'$. Note that K is a bridge or a 2-cut.

**Definition 1.** For any minimum cut K in $G_s'$, let $S \subseteq V_s$ be defined by the following (i) or (ii).

(i) If K is a bridge $e_1$ or a 2-cut $\{e_1,e_2\}$ consisting of multiple edges in $G_s'$ then $S \leftarrow \{u\}$, where $e_1=(u,u_p) \in E_d'$ and $u_p$ denotes the parent of u on T.

(ii) If K is a 2-cut $\{e_1,e_2\}$ included in the same cycle in C of $G_s'$ then $S \leftarrow L_C(v_1,u_2)$, where we assume that $e_i=(u_i,v_i)$ $(i=1,2)$, and $u_1,v_1,u_2,v_2$ appear clockwise on C.

Let $D(S) \subseteq V_d$ be the union of $V(T_a)$ of all vertices $a \in S$ in T. ♦

**Proposition 2.** If $\vec{G}_d$ has a directed edge $<x,y> \in A_d$ with $x \notin D(S)$ and $y \in D(S)$, and $e'=(x',y)=\vec{b}_d(<x,y>)$ then $x' \notin D(S)$. ♦

**Lemma 2.** If $ec(G_s'+Z) \geq \theta$ for a set $Z \subseteq E_s$ then $\vec{T}+CD(Z)$ is strongly connected, where $CD(Z)=\bigcup_{e \in Z} CD(e)$.

**(Proof)** Suppose that $\vec{T}+CD(Z)$ is not strongly connected, while $ec(G_s'+Z) \geq \theta$ holds. Clearly, r is reachable from any vertex $v \in V_d-\{r\}$ through edges in $\vec{T}$. Suppose u is the nearest vertex from r in $\vec{T}$ such that u is not reachable from r in $\vec{T}+CD(Z)$. Note that $u_p$ denotes the vertex with $<u,u_p> \in A(\vec{T})$ and that $T_v$ denotes the subtree induced by the set of all descendants of v in T. We select the vertex set $U \subseteq V_d$ containing u, as in the following (a1) or (a2).

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

(a2) $u_p=w_C \in W$. We define a vertex set $U \subseteq V(C) \subseteq V_d$ by $U=L_C(p(u),p'(u))$ with $0<p(u) \leq p'(u)<l(C)$, where U is a maximal vertex set such that $u \in U$, $v_C^0 \notin U$ and any vertex in U is not reachable from r. From the maximality of U, $v_R=v_C^{p(u)-1}$ and $v_L=v_i^{p'(u)+1}$ are reachable from r in $\vec{T}+CD(Z)$, where superindices are mod $l(C)$.

Clearly $u \in D(U)$, $u_p \notin D(U)$, and any vertex in D(U) is not reachable from r in $\vec{T}+CD(Z)$ (by Proposition 1(1)). Note that D(U) denotes the union of $V(T_a)$ of all $a \in U$. Any cut K separating D(U)-W from $V_s-(D(U) \cup W)$ is a minimum cut in $G_s'$. (If $u \notin W$ and $u_p \notin W$ then K is a bridge or a 2-cut consisting of multiple edges; if $u \in W$ or $u_p \in W$ then K is a 2-cut consisting of two edges in C.) Because K is not a cut in $G_s'+Z$, there is an edge $e=(s,v) \in Z$ such that $s \notin D(U)-W$ and $v \in D(U)-W$. Suppose that $v \in V(T_x)$ for a vertex $x \in U$. Let $t=LCA_T(v,s)$, where t may be equal to s. Then $t \notin D(U)$ and t is reachable from r in $\vec{T}+CD(Z)$. We select a vertex $z \in V_s$ as in the following (b1) or (b2). Let u' denote the son of t on the path from u to t in $\vec{T}$.

(b1) The case with $u_p \notin W$. If $t \notin W$ then $z=t$; otherwise $z=u'$.

(b2) The case with $u_p \in W$. If $t \notin W$ then $z=t$; if $t \in W$ and $t \neq u_p$ then $z=u'$; if $t=u_p \in W$ then we set z to $v_R$ or $v_L$ by the following rule. (See Fig.5.) Let $t=w_C$ and $v_C^j$ be on the <s,t>-path in $\vec{T}$. Then set $z=v_R$ if $0<j<p(u)$, or $z=v_L$ if $p'(u)<j<l(C)$.

For such a vertex z, we have $z \notin W$ and $z \notin D(U)$, and z is reachable from r. From the construction of CD(e), there is an edge $<z,v> \in CD(e) \subseteq CD(Z)$. This means that $v \in D(U)$ is reachable from r by using <z,v>, a contradiction. **Q.E.D.**

**Lemma 3.** $ec(G_s'+F_s) \geq \theta$.

**(Proof)** Suppose a minimum cut K exists in $G_s'+F_s$. Then K is a minimum cut of $G_s'$. Define two vertex sets $S \subseteq V_s$ and $D(S) \subseteq V_d$ for K by Definition 1. Clearly $ec(G_s'+CAND) \geq \theta$ holds. From Lemma 2, $\vec{G}_d(=\vec{T}+CD(CAND))$ is strongly connected. Therefore we can find a minimum-cost arborescence $\vec{T}_m=(V_d,A_m)$ in Step 6 of procedure MW*. $\vec{T}_m$ has an edge $<w,v> \in A_m$ such that $w \notin D(S)$ and $v \in D(S)$. Suppose $e'=(s,v)=\vec{b}_d(<w,v>)$. By Proposition 2, $s \notin D(S)$ and $v \in D(S)$. Because $e' \in F_s$, this contradicts that K is a cut in $G_s'+F_s$. **Q.E.D.**

**Lemma 4.** Let E" be an approximate solution for W-($\lambda$+1)ECA by MW and $E^*$ be an optimum solution for W-($\lambda$+1)ECA. Then $c(F_s) \leq 2c(E^*)$.

**(Proof)** Let $E_s^*$ denote the set of edges obtained from $E^*$ by transforming G' and c to $G_s'$, $c_s$ and $b_s$ as described at the beginning of this section. By Lemma 1, $E_s^*$ is an optimum solution such that $ec(G_s'+E_s^*) \geq \theta$. We will show that $c_d(F_s) \leq 2c_d(E_s^*)$. From Lemma 2, $\vec{T}+CD(E_s^*)$ is strongly connected and, therefore, it contains an arborescence $\vec{T}^*$ rooted at r. Since $\vec{T}_m$ is a minimum-cost arborescence, $A(\vec{T}^*) \subseteq A(\vec{T}+CD(E_s^*)) \subseteq A_d$ and $A_m \subseteq A_d$, we have $\vec{c}_d(A_m) \leq \vec{c}_d(A(\vec{T}^*))$. Because each edge of the set CD(e) generated by $e=(u,v) \in E_s^*$ has u or v as an endvertex, $\vec{T}^*$ contains at most two edges of CD(e) (one edge entering u or another one entering v). All edges of CD(e) have the same cost $c_d(e)$. Hence $\vec{c}_d(A(\vec{T}^*)) \leq 2c_d(E_s^*)$. Since $c_d(F_s) \leq \vec{c}_d(A_m)$, we have $c_d(F_s) \leq 2c_d(E_s^*)$. Because $c(E")=c_d(F_s)$ also holds, we get $c(E")=c_d(F_s) \leq 2c_d(E_s^*)=2c(E^*)$. **Q.E.D.**

We can show examples for which MW produces worst approximation such that $2c(E^*) \geq c(E")>(2-\varepsilon)c(E^*)$ for any

$\varepsilon>0$.

Since Step 5.2.2 spending $O(|V|)$ time is not executed if $\lambda$ is odd, we obtain the following theorem.

**Theorem 1.** MW correctly generates E" with $ec(G'+E")\geq\lambda+1$ in $O(\Delta+|V|^6)$ time. ♦

### 3.2. A postprocessing for improving solutions

We describe a postprocessing EXCHANGE for improvement of approximations. EXCHANGE tries to find a set of at most two edges whose replacement with an edge reduces the total cost of a solution. A solution $F_s$ for $G_s'$ is called to be *minimal* if, for any $e\in F_s$, $F_s$-{e} is not a solution to $G_s'$. The procedure REDUCE finds a minimal solution to $G_s'$.

```
procedure REDUCE(Fs);
/* Input: a solution Fs to Gs' */
/* Output: an improved solution Fs to Gs' */
   begin
1. E(1)←Fs;
2. while(E(1)≠∅) do
      begin
3.    Choose an edge e∈E(1) of largest cost cd(e);
4.    E(1)←E(1)-{e};
5.    E(2)←Fs-{e};
6.    if E(2) is a solution to Gs' then Fs←E(2)
      end
   end;
```

By using REDUCE as a preprocessing, EXCHANGE is stated as follows.

```
prodedure EXCHANGE(Fs);
/* Input: a solution Fs to Gs' */
/* Output: an improved solution Fs' to Gs' */
   begin
1. REDUCE(Fs);
2. Fs'←Fs; E(3)←Fs';
3. while(E(3)≠∅) do
      begin
4.    Choose an edge e=(u,v)∈E(3) of largest cost cd(e);
5.    E(3)←E(3)-e;
6.    if there is an edge set X having at most two edges
          such that (Fs-{e})∪X is a solution to Gs' and
          cd(e)>cd(X) then
            begin
7.          find an edge set Xmin of minimum total cost
               among all such X;
8.          Fs'←(Fs'-{e})∪Xmin;
9.          E(3)←E(3)∪Xmin
            end;
      end;
end;
```

Since the number of elements in $E^{(3)}$ is shown to be $O(|V|^2)$ and Step 7 takes $O(|V|^2)$ time for each $e\in E^{(3)}$, the time complexity of EXCHANGE is $O(|V|^4)$.

[12,13,14] show several classes of input problems for which FSM generates unbounded approximate solutions. Fig.6 shows such an example. It has been observed that RFSM finds optimum solutions to all such problems. In Fig.6, we have OPT=5. FSM generates a solution {(a,g),(b,f),(c,e),(d,h)} with total cost APP=2M+2, and APP/OPT=2M/5 +2/5. On the other hand, EXCHANGE replaces (c,e) and (a,g) with {(a,c), (e,f)} and {(g,h)}, respectively. Hence, RFSM finds an optimum solution consisting of only edges of cost 1 in Fig.6.

We have already found examples for which RFSM produces approximation slightly less than twice the optimum. However we do not know if this is always the case with RFSM. Effect of incorporating EXCHANGE will be shown by experimental results: this will be given in the next section.

## 4. Experimental Evaluation

### 4.1. Input data.

First we explain how input data, G' and c, are constructed for W-kECA.

1. Two types of data are provided: type C and type T, $|V|\in${10,15,20,40,60, 80,100,120,140,160,180,200} and $\lambda\in${1,2,3,4,5} (C means cactus-like and T means tree-like: the details are omitted for shortage of space: see [13] for the details).
2. Costs on edges are chosen randomly from {1, 2, ..., 99}.

### 4.2. Experimental results.

We have tried 3000 test data of W-($\lambda$+1)ECA so far. A workstation SUN SPARC station is used. We evaluate our algorithms with respect to the ratio APP/OPT if $|V|\in${10,15,20} (see Table 1) where the optimum solutions are found by exhaustive search, or with respect to APP/FSM otherwise (see Table 2). APP/OPT is the ratio of approximation of any one of our algorithms to the optimum, and APP/FSM is the ratio of approximation of any our algorithm to that of FSM.

Experimental results show the following (1)-(3).

(1) Incorporating EXCHANGE greatly improves approximate solutions. (For example, 2548 data for RMW show improvement. ) As can been seen from Tables 1 through 3, capability of every algorithm is improved, and, the worse approximate solutions are, the greater the total cost reductions are.

(2) RFSM shows the best performance: that is, even after incorporating EXCHANGE, FSM remains giving the best approximation. We have 750 data to each of which an optimum solution is obtained by exhaustive search. For 715 data (95.3%) of them, RFSM generates approximate solutions with errors APP/OPT-1<0.05.

(3) RMW finds the worst approximate solutions in longest computation time among the five algorithms. Although it is theoretically guaranteed that its approximation is bounded by twice the optimum, it shows the worst capability in our experimentation even if EXCHANGE is incorporated.

## 5. Concluding Remarks

In this paper, a new approximation algorithm MW and a postprocessing EXCHANGE to improve approximate solutions are proposed.

The following (1) through (2) are left for future research:

(1) Analysing worst case behavior of RFSM;

(2) Comparing experimental results for W-kECA by the approximation algorithm of [11] with RFSM;

## References


[1] K.P.Eswaran and R.E.Tarjan, Augmentation problems, SIAM J.Comput., 5, pp.653-665 (1976).

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

[3] A.Frank, Augmenting graphs to meet edge connectivity requirements, SIAM J.Discrete Mathematics, 5, No.1, pp.25-53 (1992).

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

[5] H.N.Gabow, A matroid approach to finding edge connectivity and packing arborescences, 23rd Annual Symposium on Theory of Computing, pp.112-122 (1991).

[6] H.N.Gabow, Applications of a poset representation to edge connectivity and graph rigidity, Proc.32nd IEEE Symp.Found.Comp.Sci., pp.812-821 (1991).

[7] H.N.Gabow, Z.Galil, T.Spencer and R.E.Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, Combinatorica, 6(2), pp.109-122 (1986).

[8] M.R.Garey and D.S.Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman, San Francisco (1978).

[9] D.Harel and R.E.Tarjan, Fast algorithms for finding nearest common ancestors, SIAM J.Comput., 13, pp.338-355 (1984).

[10] A.V.Karzanov and E.A.Timofeev, Efficient algorithm for finding all minimal edge cuts of a nonoriented graph, Cybernetics, pp.156-162, Translated from Kibernetika, No.2, pp.8-12 (March-April, 1986).

[11] S.Khuller and R.Thurimella, Approximation algorithms for graph augmentation, Proc. 19th International Colloquium on Automata, Languages and Programming, (Springer Lecture Notes in Computer Science 623, Springer-Verlag, Berlin, Germany), pp.330-341 (July 1992).

[12] T.Mashima, S.Taoka and T.Watanabe, Approximation algorithms for the k-edge-connectivity augmentation problem, Tech. Rep. of IEICE of Japan, COMP92-24, pp.11-20 (July 1992).

[13] T.Mashima, S.Taoka and T.Watanabe, The k-edge-connectivity augmentation problem with multiple-edge addition, Tech. Rep. of IEICE of Japan, COMP92-49, pp.11-20 (Nov. 1992).

[14] Toshiya Mashima, A Study on Approximation Algorithms for Minimum-Cost Augmentation to k-Edge-Connect a Graph, Master Thesis, Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima, Japan (March 1994).

[15] D.Naor, D.Gusfield and C.Martel, A fast algorithm for optimally increasing the edge-connectivity, Proc. 31st Annual IEEE Symposium on Foundations of Computer Science, pp.698-707 (1990).

[16] 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, pp.49-56 (Jan. 1994).

[17] R.E.Tarjan, Data Structures and Network Algorithms, CBMS-NSF 44, Regional Conference Series in Applied mathematics, SIAM, Philadelphia, PA (1983).

[18] T.Watanabe, T.Mashima and S.Taoka, The k-edge-connectivity augmentation problem of weighted graphs, Proc. 3rd International Symposium on Algorithms and Computation (ISAAC'92) (Springer Lecture Notes in Computer Science 650, Springer-Verlag, Berlin, Germany), pp.31-40 (Dec. 1992).

[19] T.Watanabe, T.Mashima and S.Taoka, Approximation algorithms for minimum-cost augmentation to k-edge-connect a multigraph, Proc. 1993 IEEE ISCAS, pp.2556-2559 (May 1993).

[20] T.Watanabe and A.Nakamura, Edge-connectivity augmentation problems, Journal of Computer and System Sciences, 35(1), pp.96-144 (1987).

[21] T.Watanabe, T.Narita and A.Nakamura, 3-Edge-connectivity augmentation problems, Proc. 1989 IEEE ISCAS, pp.335-338 (1989).

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


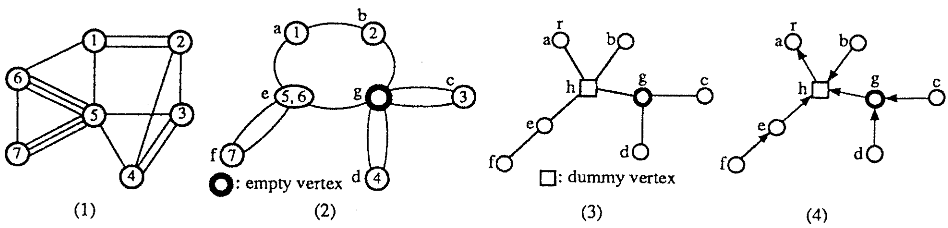


Fig.1. An example of our graph transformation: (1) $G'=(V,E')$; (2) $G_s'=(V_s,E_s')(=(F(G'))$; (3) $T=(V_d,E_d')$; (4) $\vec{T}=(V_d,A_d')$.

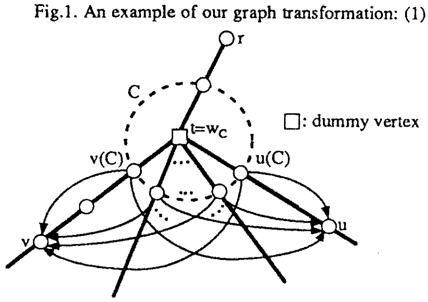


Fig.2. The set $A_c(u,v)$ of directed edges in case $LCA_{G_d'}(u,v) \in W$. $G_d'$ may be replaced by $\vec{G}_d'$: in that case, edges of $G_d'$ are directed towards r.

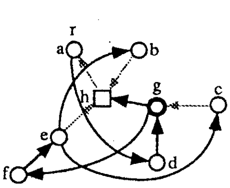

Fig.3. An example of a minimum-cost arborescence $\vec{T}_m=(V_d,A_m)$.

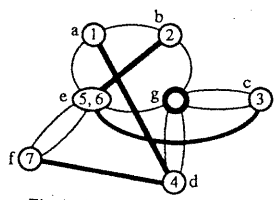

Fig.4. An example of an edge set $F_s$ obtained from the arborescence in Fig.3.

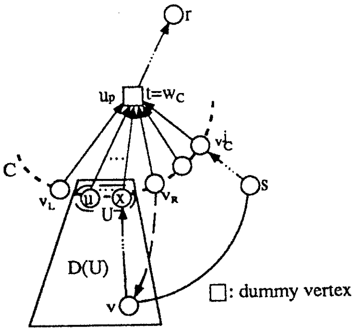


Fig.5. Proof of Lemma 2 in case $u \notin W$, $u_p \in W$, $t \in W$ and $t=u_p$.

Table 2. Comparison of APP/FSM. The average (left) and the maximum (right) of the ratio APP/FSM over 3000 data of type C and type T for each $\lambda \in \{1,2,3,4,5\}$ concerning W-($\lambda$+1)-ECA of large size, where no optimum can be found.

| | ave. | max |
|---|---|---|
| FSA | 1.185 | 1.808 |
| RFSA | 1.037 | 1.524 |
| FSM | 1.000 | 1.000 |
| RFSM | 0.998 | 1.000 |
| SMC | 1.059 | 1.808 |
| RSMC | 1.042 | 1.524 |
| HBD | 1.025 | 1.409 |
| RHBD | 1.011 | 1.250 |
| MW | 1.202 | 1.808 |
| RMW | 1.052 | 1.524 |

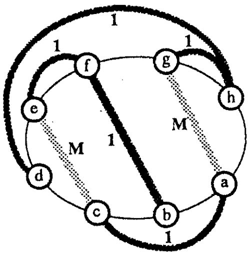


Fig.6. An example for which FSM generates an approximate solution whose total cost cannot be bounded by constant times the optimum.

Table 1. Comparison of errors (APP/OPT-1). Total number of data (left) and its ratio (right) such that each algorithm produces solutions with errors falling into the corresponding intervals. $\delta$=1 and the total number of data is 750 (type C and type T), to each of which an optimum solution is found by exhaustive search. The average and the maximum of errors APP/OPT-1 is also shown in the rightmost two columns.

| err. | err.=0 | | 0<err.≤0.05 | | 0.05<err.≤0.10 | | 0.10<err.≤0.15 | | 0.15<err. | | ave. err. | max err. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FSA | 292 | 38.9% | 64 | 8.5% | 75 | 10.0% | 84 | 11.2% | 235 | 31.3% | 0.115 | 0.816 |
| RFSA | 564 | 75.2% | 67 | 8.9% | 55 | 7.3% | 30 | 4.0% | 34 | 4.5% | 0.022 | 0.524 |
| FSM | 604 | 80.5% | 66 | 8.8% | 49 | 6.5% | 15 | 2.0% | 16 | 2.1% | 0.015 | 0.565 |
| RFSM | 682 | 90.9% | 33 | 4.4% | 25 | 3.3% | 5 | 0.7% | 5 | 0.7% | 0.006 | 0.265 |
| SMC | 457 | 60.9% | 80 | 10.7% | 74 | 9.9% | 62 | 8.3% | 77 | 10.3% | 0.047 | 0.808 |
| RSMC | 502 | 66.9% | 86 | 11.5% | 73 | 9.7% | 49 | 6.5% | 40 | 5.3% | 0.030 | 0.524 |
| HBD | 386 | 51.5% | 114 | 15.2% | 103 | 13.7% | 83 | 11.1% | 64 | 8.5% | 0.046 | 0.565 |
| RHBD | 592 | 78.9% | 70 | 9.3% | 48 | 6.4% | 31 | 4.1% | 9 | 1.2% | 0.015 | 0.250 |
| MW | 289 | 38.5% | 65 | 8.7% | 83 | 11.1% | 87 | 11.6% | 226 | 30.1% | 0.108 | 0.808 |
| RMW | 533 | 71.1% | 75 | 10.0% | 64 | 8.5% | 30 | 4.0% | 48 | 6.4% | 0.029 | 0.524 |

Table 3. Comparison of improvement ratios r=cost(R*)/cost(*). Total number of data (left) and its ratio (right) such that each algorithm * produces solutions with improvement ratios r falling into the corresponding intervals. Other situations are the same as Table 1.

| r | r<0.85 | | 0.85≤r<0.90 | | 0.90≤r<0.95 | | 0.95≤r<1 | | r=1 | | ave. r | min r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFSA | 961 | 32.0% | 757 | 25.2% | 573 | 19.1% | 272 | 9.1% | 437 | 14.6% | 0.882 | 0.551 |
| RFSM | 5 | 0.2% | 8 | 0.3% | 34 | 1.1% | 156 | 5.2% | 2797 | 93.2% | 0.998 | 0.639 |
| RSMC | 24 | 0.8% | 22 | 0.7% | 108 | 3.6% | 1083 | 36.1% | 1763 | 58.8% | 0.985 | 0.553 |
| RHBD | 27 | 0.9% | 51 | 1.7% | 138 | 4.6% | 926 | 30.9% | 1858 | 61.9% | 0.986 | 0.639 |
| RMW | 997 | 33.2% | 796 | 26.5% | 526 | 17.5% | 229 | 7.6% | 452 | 15.1% | 0.881 | 0.553 |