アルゴリズム　30－1
（1992. 11. 20）

指定点集合に対する3-辺連結化問題

田岡智志　間島利也　渡辺敏正

広島大学工学部第二類(電気系)
724 東広島市鏡山一丁目4-1
(電話) 0824-22-7111 内3442(渡辺)
(ファクス) 0824-22-7195
(電子メール) watanabe@huis.hiroshima-u.ac.jp

あらまし

本稿の主題は以下のように定義される，グラフの指定点集合に対する3-辺連結化問題(3-ECA-SV)である："グラフG'=(V,E')，各2点間のコストを表すコスト関数c:V×V→Z^+(非負整数)及び指定点集合Γ⊆Vが与えられたとき，Vの2点を結ぶ辺の集合でG'に付加して得られるグラフG"=(V,E'∪E")がΓの任意の2点間に辺素なパスを3本以上持つことになるもののうちコスト総和が最小となるものE"を求めよ"．3-ECA-SV(M)をG'に多重グラフを許し，且つE"の付加によって新しく多重辺が生じることも許す3-ECA-SVとする．3-ECA-SV(M)に対し4つの近似アルゴリズムGS-A, GS-M, GS-S, GS-Hを提案し，これらの近似解の理論的，実験的評価を行う．

和文キーワード　辺付加問題，辺連結度，近似アルゴリズム，グラフ，時間複雑度

The 3-Edge-Connectivity Augmentation Problem for a Specified Set of Vertices

Satoshi Taoka, 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:+81-824-22-7111 Ext.3442(Watanabe), Fax:+81-824-22-7195
E-mail:watanabe@huis.hiroshima-u.ac.jp

Abstract

The *3-edge-connectivity augmentation problem for a specified set of vertice* (3-ECA-SV for short) is defined by "Given a graph G'=(V,E'), a cost function c:V×V→Z^+ (nonnegative integers) with V×V={{u,v}|u,v∈ V,u≠v} and a subset Γ⊆V, find a minimum-cost set E" of edges, each connecting distinct vertices of V, such that G"=(V,E'∪E") has at least three edge-disjoint paths between any pair of vertices in Γ." Let 3-ECA-SV(M) denote 3-ECA-SV such that G' may have multiple edges and creation of multiple edges in G" is allowed. Four approximation algorithms GS-A, GS-M, GS-S and GS-H for 3-ECA-SV(M) are proposed, and both theoretical and experimental evaluation are given.

英文 key words　Graph augmentation problems, edge-connectivity, 3-edge-connectedness, approximation algorithms

1. INTRODUCTION

The *3-edge-connectivity augmentation problem for a specified set of vertice* (3-ECA-SV for short) is defined by "Given a graph $G'=(V,E')$, a cost function $c:V\times V\rightarrow Z^+$ (nonnegative integers) with $V\times V=\{\{u,v\}|u,v\in V,u\neq v\}$ and a subset $\Gamma\subseteq V$, find a minimum-cost set E'' of edges, each connecting distinct vertices of V, such that $G''=(V,E'\cup E'')$ has at least three edge-disjoint paths between any pair of vertices in Γ." Such an edge set E'' is called a *minimum solution* to the problem, and we may assume $|\Gamma|\geq 2$. G'' is also written as $G'+E''$. Costs $c(\{u,v\})$ for $\{u,v\}\in V\times V$ is denoted as $c(u,v)$ for simplicity. The problem is called the *weighted version*, denoted by W-3-ECA-SV, if there may exist some distinct edge costs and the *unweighted one*, denoted by UW-3-ECA-SV, otherwise.

Let 3-ECA-SV(S) denote 3-ECA-SV with the following restriction (i)-(iii) on G', c and E'', respectively: (i) G' is a simple graph; (ii) $c:E\rightarrow Z^+$, with $E=\{(u,v)|\{u,v\}\in V\times V\}$ (the edge set of a complete graph having V as the vertex set), such that $c(e)=0$ if $e\in E'$ and $c(e)>0$ if $e\in E-E'$; (iii) $E''\subseteq E-E'$. (We assume $c(u,v)=1$ for any $\{u,v\}\in V\times V$ in UW-3-ECA-SV in the paper.) Let 3-ECA-SV(M) denote 3-ECA-SV such that G' may have multiple edges and creation of multiple edges in G'' is allowed. 3-ECA-SV is extension of 2-ECA-SV, which is defined similarly, and results for W-2-ECA(S) are shown in [13,14]. If $\Gamma=V$ then the problem is called the 3-edge-connectivity augmentation problem (denoted as 3-ECA). 3-ECA and k-ECA (generalization of 3-ECA) have been mainly discussed in literature: see [17,18,19,20,21] for 3-ECA and [1,2,3,4,7,9,12,15,16] for k-KECA with $k\geq 2$; $k\neq 3$. Concerning UW-k-ECA, only UW-k-ECA(M) has been discussed so far, while, as for W-k-ECA, only W-k-ECA(S) has been considered. Since W-3-ECA(S) is known to be NP-complete [17] and since this proof technique can be used in proving the NP-completeness of W-3-ECA(M), both W-3ECA-SV(S) and W-3-ECA-SV(M) are NP-complete.

The subject of the paper is 3-ECA-SV(M). The paper first shows that there is an $O(|V|+|E'|)$ algorithm for solving UW-3-ECA-SV(M). It is shown that we can equivalently transform UW-3-ECA-SV into UW-3-ECA in $O(|V|+|E'|)$ time in Section 3. Since it is known that UW-3-ECA(M) has an an $O(|V|+|E'|)$ algorithm (by combining results [6, 8, 10] and [19, 20]; see also [21]), UW-3-ECA-SV(M) can be solved in linear time. On the other hand, the above transformation can be used in reducing W-3-ECA-SV to W-3-ECA: in this case optimality is not always preserved and we manipulate cost functions so that existence of solutions to W-3-ECA may imply that of those to W-3-ECA-SV. Four approximation algorithms GS-A, GS-M, GS-S and GS-H are proposed for W-3-ECA-SV(M) in Section 4. Each of them is given as a general scheme GS incorporating any one of procedures FSA**, FSM**, SMC** and HBD** for finding approximate solutions to W-3-ECA(M), they are denoted as GS-A, GS-M, GS-S and GS-H, respectively. For simplicity G' is restricted to 2-edge-connected one in this paper unless otherwise stated. Both theoretical and experimental evaluation are given in Section 5. As theoretical evaluation it is shown that both GS-A and GS-M produce worst approximation no greater than twice the optimum if they are applied to UW-3-ECA-SV(M). As experimental evaluation of W-3-ECA-SV(M), it is shown that $cost(GS\text{-}M)<cost(GS\text{-}H)<cost(GS\text{-}S)<cost(GS\text{-}A)$ and $time(GS\text{-}S)<time(GS\text{-}A)<time(GS\text{-}H)<time(GS\text{-}M)$, where cost(GS-M) and time(GS-M) are the total cost of an approximate solution by GS-M and its computation time, and similarly for others.

It should be mentioned that UW-3-ECA-SV(S) and W-3-ECA-SV(S) can be solved similarly to the paper: the former is optimally solved since UW-3-ECA(S) and UW-3-ECA(M) have the same minimum solution if $|V|\geq 4$ (see [21]), and, for the former, it is experimentally observed that the algorithms to be proposed by the paper produce good approximate solutions.

In the rest of the paper UW-3-ECA-SV(M) or W-3-ECA-SV(M) is denoted as USV or WSV, respectively, for notational simplicity.

2. PRELIMINARIES

An *undirected graph* $G=(V(G),E(G))$ consists of a finite and nonempty set of vertices V(G) and a finite set of undirected edges E(G); an edge e incident upon two vertices u,v is denoted by (u,v); u and v are the *endvertices* of an edge e; e is called a *loop* if u=v. V(G) and E(G) are often denoted as V and E, respectively. If there are two edges both of which have the same pair of endvertices then G is called a *multigraph*. Such edges are called *mutiple edges*; otherwise G is called a *simple graph*. A *directed graph* $\overrightarrow{G}=(V(\overrightarrow{G}),A(\overrightarrow{G}))$, or simply denoted as $\overrightarrow{G}=(V,A)$, consists of a set of vertices $V(\overrightarrow{G})$ and a set of *directed edges* $A(\overrightarrow{G})$. A directed edge from u to v is denoted as $<u,v>$. In this paper, only graphs without loops are considered, and the term "a graph" means an undirected multigraph unless otherwise stated. The *degree* of a vertex v in G, denoted by $d_G(v)$ or d(v), is the total number of edges (v,v'), $v'\neq v$, incident upon v, and v is often called a degree-d(v) vertex (of G).

For a set $R\subseteq V(G)$, let G[R] denote the subgraph having R as its vertex set and $\{(u,v)\in E(G)|\ u,v\in R\}$ 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 $R\subseteq V(G)$ from G is to construct G[V(G)-R], which is often denoted by G-R. If $R=\{v\}$ then we often denote G-v. Deletion of $Q\subseteq E(G)$ from G defines a spanning subgraph of G, denoted by G-Q, having E(G)-Q as its edge set. If $Q=\{e\}$ then we denote G-e. For a set E' of edges such that $E'\cap E(G)=\emptyset$, let G+E' denote the graph $(V(G),E(G)\cup E')$. If $E'=\{e\}$ then we denote G+e. Shrinking $R\subseteq V(G)$ to a vertex r_0 is to construct a graph having $(V(G)-R)\cup\{r_0\}$ as its vertex set and $E(G-S)\cup\{(r_0,v)|(u,v)\in E(G), u\in R, v\in V(G)-R\}$ as its edge set, where any loop created is deleted.

A *path* between u and v, or a *(u,v)-path*, is an alternating sequence of vertices and edges $u=v_0,e_1,v_1,\ldots,v_{n-1},e_n,v_n=v$ ($n\geq 0$) such that if $n\geq 1$ then $v_0,\ldots,v_n$ are all distinct and $e_i=(v_{i-1},v_i)$ for each i, $1\leq i\leq n$. The *length* of this path is n. Vertices $v_1,\ldots,v_{n-1}$ are called *inner vertices* of this path if $n\geq 2$. A *cycle* is a (v_0,v_n)-path together with an edge (v_0,v_n). The length of this cycle is n+1. A pair of multiple edges are considered as a cycle of length two.

Two paths P, P' are said to be *edge-disjoint* (*internally disjoint*, respectively) if $E(P)\cap E(P')=\emptyset$ (P and P' have no inner vertex in common). The *edge-connectivity* (*vertex-connectivity*, respectively) of a graph G, denoted by $\lambda(G)$ ($\kappa(G)$), is the minimum number of edges (vertices) whose deletion from G disconnect it. A graph G is *k-edge-connected* (*k-vertex-connected*) if and only if $\lambda(G)\geq k$ ($\kappa(G)\geq k$). A *k-edge-connected* (*vertex-connected*, respectively) *component* of G is a maximal subset of vertices such that, for any two vertices in the set, G has at least k edge-disjoint (internally disjoint) paths between them. A k-edge-connected component is

often denoted as a k-ecc (a k-vcc) in this paper. It is known that $\lambda(G)\geq k$ ($\kappa(G)\geq k$, respectively) if and only if $V(G)$ is a k-ecc (a k-vcc). Note that distinct k-eccs are disjoint sets. Each 1-ecc is often called a *component*. A set $K\subseteq E(G)$ is called a (u,v)-*separator* if and only if u and v belong to distinct components of G-K. Let $\lambda(u,v;G)$ ($\kappa(u,v;G)$, respectively) denote the maximum number of edge-disjoint (internally disjoint) (u,v)-paths of G. A (u,v)-separator $K\subseteq E$ is called a (u,v)-*cut* if and only if $|K|=\lambda(u,v;G)$. A (u,v)-cut K is called a (u,v)-*cutpair* or simply a cutpair if and only if $|K|=2$ and is called a *bridge* if and only if $|K|=1$. A vertex $v\in V(G)$ is called a *cutpoint* of G if and only if the number of components of G-v is greater than that of G. For any cutpoint $u\in V(G)$ and each component H of G-u, $G[V(H)\cup\{u\}]$ is called a u-*block* of G. For nonempty disjoint sets S, $S'\subset V(G)$, let $\lambda(S,S';G)=|\{(u,v)\in E(G)|u\in S \text{ and } v\in S'\}|$. If $S'=V(G)-S$ then it is written as $K_G(S)$ and we denote $d(S,G)=|K_G(S)|$.

A *leaf* of a tree is a vertex with only one edge incident on it. 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. A minimum-cost arborescence is an arborescence of minimum total cost. A *branching* of a directed graph G is a graph whose weakly connected components are arborescence. A *cactus* is an undirected connected graph in which any pair of cycles share at most one vertex: each shared vertex is a cutpoint. A *leaf* of a cactus G is a vertex v with either $d_G(v)=1$ or $d_G(v)=2$ and v is on a cycle.

A pair of edges without sharing vertices in G are said to be *independent*. An edge set in which any pair are independent in G is called a *matching* of G, and a matching of maximum cardinality is called a *maximum matching*. A *maximum-cost matching* of a weighted undirected graph is a matching whose total cost is maximum. A maximum cost matching of a given graph G is obtained in $O(|V||E|)$ time [11].

3. Solving 3-ECA-SV as 3-ECA

We first explain reduction of 3-ECA-SV into 3-ECA such that a minimum solution to one of the two problems implies one to the other. Then we describe a general scheme for solving 3-ECA-SV as an algorithm GS, where we assume that $\lambda(G')=2$ for ease of understanding: the cases with $0\leq\lambda(G')\leq 1$ are omitted due to shortage of space. Note that all h-components with $h\leq 3$ of G' can be found in $O(|V|+|E'|)$ time: see [11] for $h\leq 2$ and [6,8,10] for h=3.

Let $G_1'=(V_1,E_1')$ denote the graph obtained by shrinking each 3-ecc of G' into individual vertex, and let $V_a\subseteq V_1$ be the set of vertices into each of which a 3-ecc containing at least one vertex of Γ is shrunk, where we assume that each 3-ecc S is shrunk into a vertex v_S in S. If G' of Fig. 1 is given then G_1' is shown in Fig. 2, where vertices in V_a are denoted as black spots. We denote v_S as $\alpha_1(S)$ and conversely S as $\beta_1(v_S)$. Since we assume $\lambda(G')=2$, G_1' is a cactus consisting of some cycles. Note that $d_{G_1}(v)$ is even for every vertex $v\in V_1$ and that v is a cutpoint if and only if $d_{G_1}(v)\geq 4$. A cycle of G_1' is called a *pendant* if it contains at most one cutpoint. A pendant is called a *core pendant* if it contains at least one vertex of V_a that is not a cutpoint. If G_1' has a cutpoint and there is any pendant that is not a core one then delete all vertices except the cutpoint of this pendant, and repeat this procedure as much as possible. Let $G_2'=(V_2,E_2')$ denote the resulting cactus (see Fig. 3). Clearly any pendant of G_2' is a core one and $V_a\subseteq V_2$. The set V_1-V_2 has a partition $V_1-V_2=W_1\cup\ldots\cup W_k$ ($k\geq 1$; $W_i\cap W_j=\emptyset$ if $i\neq j$) such that, for each W_i, there is a cutpoint v_i for which $X_i=W_i\cup\{v_i\}$ induces a v_i-block of G_1'. Each X_i and v_i are called an *outer component* of G_1' and the attachment of X_i, respectively. Clearly G_2' is obtained from G_1' by shrinking each X_i into v_i, $i=1,\ldots,k$. We denote $v_i=\alpha_2(X_i)$ and $X_i=\beta_2(v_i)$. Also we denote $\alpha_2(v)=v$ and $\beta_2(v)=v$ for any $v\in V_2-\{v_1,\ldots,v_k\}$ ($\subseteq V_1$). If G_2' has a (v_0,v_n)-path of length $n\geq 2$ with inner vertices $v_i\notin V_a$ and $d_G(v_i)=2$, $i=1,\ldots,n-1$, then delete all inner vertices $v_1,\ldots,v_{n-1}$ and add an edge (v_0,v_n). Repeat this procedure as much as possible, and let $\chi(G_2')$ denote the resulting graph, which is called the *condensation* of G_2'. Denote $\chi(G_2')$ as $G_3'=(V_3,E_3')$ (Fig. 4). Note that $V_a\subseteq V_3$ and any vertex v with $d_{G_3'}(v)=2$ belongs to V_a. Let $L_3=\{v\in V_3|d_{G_3'}(v)=2\}$ ($\subseteq V_a$).

Let $v_0,v_1,\ldots,v_n$ ($n\geq 1$) be a sequence of vertices of G_2' such that $v_i\neq v_j$ for $i,j\in\{0,1,\ldots,n\}$, $i\neq j$. If we consider each pair $\{v_i,v_{i+1}\}$ as an edge (v_i,v_{i+1}), $i=0,\ldots,n-1$, then this sequence represents a (v_0,v_n)-path. Hence we often call it a (v_0,v_n)-sequence of V_2. Suppose that $\lambda(G')=2$ and $|V_1|\geq 3$. Let w,w' be a pair of vertices of V_1 such that G_2' has a cutpoint separating them. Let $a_1,\ldots a_k$ ($k\geq 1$) be the sequence of those cutpoints separating w from w' of G_2' such that there is the sequence of cycles $C_1,\ldots,C_{k+1}$ satisfying $V(C_j)\cap V(C_{j+1})=\{a_j\}$, $\{w,a_1\}\subseteq V(C_1)$, $\{a_k,w'\}\subseteq V(C_{k+1})$ for $j=1\ldots k$. The two sequences are called the (w,w')-*cutpoint sequence* and the (w,w')-*cycle sequence* of G_2', respectively. Note that $\{w,w'\}\cup\{a_1,\ldots,a_k\}$ is a 3-ecc of $G_2'+(w,w')$. Put $K_{ww'}=\{a_1,\ldots a_k\}$. Let $\{u,v'\}$ and $\{u',v\}$ be two distinct pairs of vertices of G_2'. (Consider u=1, u'=6, v'=10 and v=4 in G_2' of Fig. 5 as an example: $\{a_1=2, a_2=3\}$ in the (u,v)-cutpoint sequence of G_2'.) These pairs are called *crossing* if the following (i) and (ii) hold:

(i) if there is a cycle C containing $\{u,v'\}$ of G_2' then C includes $\{u',v\}$ and they appear as u,u',v',v in this order clockwise;

(ii) if G_2' has a cutpoint separating u from v' then u' and v belong to distinct a_1-blocks of G_2'', where G_2'' is obtained from G_2' by shrinking $K_{uv'}\cup\{u,v'\}$ into the first cutpoint $a_1\in K_{uv'}$ of the (u,v')-cutpoint sequence.

It is easy to see that $G_2'+\{(u,v'),(u',v)\}$ has a 3-ecc S containing $\{u,v'\}\cup\{u',v\}$ if and only if $\{u,v'\}$ and $\{u',v\}$ are crossing pairs.

Example 3.1. Consider a graph G_2' of Fig. 5. $V_2=\{1,\ldots,10\}$, $V_a=\{1,4\}$ and c_2' is shown in the figure between pairs of vertices, where costs not shown are assumed to be very large. $E_2''=\{(1,10),(4,6)\}$ is a minimum solution with total cost 20. In G_3' of Fig. 6, if we have $c_3(1,4)=20$ then we will obtain $\{(1,4)\}$ as a minimum solution to W-3ECA for G_3', while if $c_3(1,4)=c_2(1,4)=300$ or $c_3(1,4)=210$ (which is the length of a shortest (1,4)-path in G_2') then a minimum solution to W-3ECA for G_3' is not a minimum one to WSV for G_2'. ♦

Now we describe an algorithm GS.

Algorithm GS

/* *Input*: a graph $G'=(V,E')$, a set $\Gamma\subseteq V$ of specified vertices and a cost function $c:V\times V\to Z^+$(nonnegative integers) such that, in case of USV, $c(u,v)=1$ for any $\{u,v\}\in V\times V$, where $V\times V=\{\{u,v\}|u,v\in V, u\neq v\}$. */

/* *Output*: a minimum-cost set E" of edges, each connecting distinct vertices of V, such that $\lambda(u,v;G'+E")\geq 3$ for any pair of vertices $u,v\in\Gamma$. */

1. (1) Construct a graph $G_1'=(V_1,E_1')$ from G' by shrinking each 3-ecc of G' into individual vertex.

(2) Let $V_a\subseteq V_1$ be the set of vertices into each of which a 3-ecc containing at least one vertex of Γ is shrunk.

(3) In WSV, define a cost function $c_1:V_1\times V_1\to Z^+$ and a backpointer $b_1:V_1\times V_1\to V\times V$ by

$c_1(u,v)=\min\{c(u',v')|u'\in\beta_1(u), v'\in\beta_1(v)\}$,

$b_1(u,v)=(u",v")$ such that $c_1(u,v)=c(u",v")$.

2. Find all outer components $X_1,\ldots,X_k$ ($k\geq 0$) of G_1'.

3. (1) If $k\geq 1$ then construct $G_2'=(V_2,E_2')$ by shrinking each outer component X_i into its attachment v_i, $i=1,\ldots,k$.

(2) In WSV, define a cost function $c_2:V_2\times V_2\to Z^+$ and a backpointer $b_2:V_2\times V_2\to V_1\times V_1$ by

$c_2(u,v)=\min\{c_1(u',v')|u'\in\beta_2(u), v'\in\beta_2(v)\}$,

$b_2(u,v)=(u",v")$ such that $c_2(u,v)=c_1(u",v")$.

4. (1) Construct the condensation $\chi(G_2')$ and denote $\chi(G_2')$ as $G_3'=(V_3,E_3')$.

(2) In WSV, define a cost function $c_3:V_3\times V_3\to Z^+$ and a backpointer

$b_3:V_3\times V_3\to\{(v_0,v_n)$-sequences of a complete graph $G_2=(V_2,E_2)|v_0,v_n\in V_2\}$

by

$c_3(u,v)$=(the length of a shortest (u,v)-path P_{uv} in G_2', and

$b_3(u,v)=P_{uv}$.

(For efficiency, if P_{uv} has an inner vertex $w\in V_a$ then

$c_3(u,w)\leftarrow$the length of P_{uw}, $b_3(u,w)\leftarrow P_{uw}$,

$c_3(v,w)\leftarrow$the length of P_{vw}, $b_3(v,w)\leftarrow P_{vw}$,

where P_{uw} and P_{vw} are the (u,w)-subpath and (v,w)-subpath of P_{uv}.)

(3) In WSV, for each pair $u,v\in V_a$ of G_3', define a cost function $c_4:V_3\times V_3\to Z^+$ and a backpointer

$b_4:V_3\times V_3\to\{(u,v)|u,v\in V_2, u\neq v\}$

$\cup\{\{\{u,v'\},\{u',v\}\}|\{u,v'\}\neq\{u',v\}, u,u',v,v'\in V_2\}$

by

$c_4(u,v)=\min\{\{c_3(u,v)\}\cup\{c_2(u,v')+c_2(u',v)|$

$\{u,v'\}$ and $\{u',v\}$ are a crossing pair of $G_2'\}\}$,

$$b_4(u,v)=\begin{cases}\{(u,v)\} & \text{if } c_3(u,v)\leq c_2(u,v')+c_2(u',v),\\ \{(u,v'),(u',v)\} & \text{otherwise.}\end{cases}$$

5. (1) In USV, solve UW-3ECA for G_3' (that is, find a set $E_3"$ of minimum cardinality such that $\lambda(G_3'+E_3")=3$) by means of an $O(|V_3'|+|E_3'|)$ algorithm, denoted as ATEC, proposed in [19,20,21].

(2) In WSV, find an approximate solution to W-3ECA for G_3' (that is, find a set $E_3"$ of small total cost such that $\lambda(G_3'+E_3")=3$) by means of any one of FSA**, FSM**, SMC** and HBD** to be proposed in Section 4.

6. Define E" by

$E"=\{b_1(b_2(b_3(b_4(u,v))))|(u,v)\in E_3"\}$ (with multiplicity deleted). ♦

Remark 3.1. If SMC is used for solving WSV then Step 4(2) is omitted. ♦

The relation among G', G_2' and G_3' are shown by the following lemmas.

Lemma 3.1. There is an edge set E" of minimum total cost $c(E")$ such that $\lambda(u,v;G'+E")\geq 3$ for $\forall u,v\in\Gamma$ if and only if there is an edge set $E_2"$ of minimum total cost $c_2(E_2")$ such that $\lambda(u',v';G_2'+E_2")\geq 3$ for $\forall u',v'\in V_a$, where $c(E")=c_2(E_2")$. ♦

Lemma 3.2. In USV, there is an edge set $E_2"$ of minimum cardinality such that $\lambda(u',v';G_2'+E_2")\geq 3$ for $\forall u',v'\in V_a$ if and only if there is an edge set $E_3"$ of minimum cardinality such that $\lambda(G_3'+E_3")\geq 3$, where $|E_2"|=|E_3"|$. In WSV, there is an edge set $E_2"$ such that $c_2(E_2")=c_3(E_3")$ and $\lambda(u',v';G_2'+E_2")\geq 3$ for $\forall u',v'\in V_a$ if there is an edge set $E_3"$ of total cost $c_3(E_3")$ such that $\lambda(G_3'+E_3")\geq 3$. ♦

We obtain the next theorem.

Theorem 3.1. USV can be solved optimally by GS-U in $O(|V|+|E'|)$ time if $\lambda(G')=2$. ♦

(It should be mentioned that, in USV, a similar propositon holds for cases $0\leq\lambda(G')\leq 1$.) For WSV, constructing G_3', c_3 and b_3 from G_2', c_2 does not always preserve equivalence between two solutions of minimum total costs: it depends upon how to define c_3 from c_2. In the next section, we propose approximation algorithms for solving WSV: finding a solution of small total cost to W-3ECA for G_3' so that it may make a good approximate one to WSV for G_2'.

4. APPROXIMATION ALGORITHMS FOR WSV

Four procedures FSA**, FSM**, SMC** and HBD** that produce approximate solutions to W-3-ECA(M) for G_3' in Step 5(2) of GS will be proposed. They run in $O(|V|^2)$, $O(|V|^3 log|V|)$, $O(|V|^3)$ and $O(|V|^3 log|V|)$ time, respectively, since we can devise an $O(|V|^3)$ algorithm for Step 4(3) of GS (the detail is omitted here). Similar procedures FSA*, FSM*, SMC* and HBD* have been proposed as approximation algorithms for W-3-ECA(S) in [18], and they are modified so that they may make approximation ones for W-3-ECA(M), resulting in FSA**, FSM**, SMC** and HBD**. Let $G_s'=(V_s,E_s')$ denote G_3', and note that L_3 is the set of all leaves (degree-2 vertices) of G_s'.

4.1 Procedure FSA** Based on Minimum-Cost Arborescence

The first procedure to be proposed is FSA* based on a

minimum-cost arborescence algorithm [5]. The idea, using minimum-cost arborescence in finding solutions to W-3-ECA(M), is based on the results for W-2-ECA(S) in [3]. The algorithm is outlined as follows. First, G_S' is changed into a simple graph by deleting multiplicity, and then a tree G'_b will be constructed as follows: for each cycle C that is remaining in this simple graph, a new vertex v_C is added. For every cycle C, each vertex on C and v_C are connected by an edge. Then all edges of C are deleted. Let $G_b'=(V_b,E_b')$ denote the resulting tree. Next, we choose a specified degree-1 vertex r of G_b' as the root, and direct every edge of G'_b toward r. Let $\overline{G}_b'$ denote the resulting directed graph. Some modification of costs will be done. Let $\overline{G}_b$ and d' be the complete directed graph on V_b and a final cost function, respectively. We find a minimum-cost arborescence $T=(V_b, A_b'')$ of $\overline{G}_b$ with respect to d'. Finally an approximate solution is obtained by means of backpointers.

We first present procedure REMAKE that changes a cactus G_S' into a spanning tree G_b'.

procedure REMAKE;
/* *Input*: $G_S'=(V_S,E_S')$. *Output*:$G_b'=(V_b,E_b')$. */
1. Delete multiplicity of edges in G_S', making it simple. Then find all cycles by a depth-first-search.
2. For each cycle C, add a dummy vertex w_C, connect w_C to every vertex on C and delete all edges (u,v) of C. Let $G_b'=(V_b,E_b')$ be the resulting graph. Define a cost function $c'':V_b\times V_b\rightarrow Z^+\cup\{\infty\}$ as follows: for all pairs $u,v\in V_S$, set $c''(u,v)=c'(u,v)$; for all dummy vertices w_C and all vertices u of V_S, set $c''(w_C,u)=\infty$. ♦

We use a distance function $d:V_b\times V_b\rightarrow Z^+\cup\{\infty\}$. This function is introduced in [3] in order to avoid poor choice of edges in finding minimum-cost arborescence, and is defined by

$$d(u,v)=\min\{c'(x,y)|\ u \text{ and } v \text{ are on a path from } x \text{ to } y \text{ in } G_b'\}.$$

The following procedure has been proposed in [3] and is rewritten here so that it can be applied to G_b'.

procedure DIST;
/* *Input*: a graph $G_b'=(V_b,E_b')$, a cost function $c'':V_b\times V_b\rightarrow Z^+\cup\{\infty\}$, a backpointer b. */
/* *Output*: A distance function $d:V_b\times V_b\rightarrow Z^+\cup\{\infty\}$ and a backpointer $b':V_b\times V_b\rightarrow V_b\times V_b$ such that $c'(b'(u,v))=d(u,v)$. */
1. For each pair $u,v\in V_b$, compute the number a(u,v) of edges on the path between u and v in G_b', and find the vertex s(u,v) adjacent to v on this path. Let

$d(u,v)\leftarrow c''(u,v)$ and $b'(u,v)\leftarrow(u,v)$.

2. Bucketsort the pairs (u,v) in $V_b\times V_b$ into nonincreasing order of a(u,v). For each edge $(u,v)\in V_b\times V_b$ do the following in its sorted order:

$d(u,s(u,v))\leftarrow d(u,v)$ and $b'(u,s(u,v))\leftarrow b'(u,v)$ if $d(u,v)<d(u,s(u,v))$;

$d(s(v,u),v)\leftarrow d(u,v)$ and $b'(s(v,u),v)\leftarrow b'(u,v)$ if $d(u,v)<d(s(u,v),v)$. ♦

It can be proved, similarly to [3], that DIST correctly computes the distance function d and the backpointer b' in $O(|V|^2)$ time. For two different edges $(u,v),(u',v')\in V_b\times V_b$, it may happen that $b'(u,v)=b'(u',v')$. Hence, in general, $b'(Z)=\{b'(u,v)|(u,v)\in Z\}$ may be a multiset for a set $Z\subseteq V_b\times V_b$. However, we assume that b'(Z) denotes the one with multiplicity deleted unless otherwise stated, for notational simplicity. Similar notation will be used for other backpointers to be defined later. Procedure FSA** is stated as follows.

procedure FSA**;
/* *Input*: a graph $G_S'=(V_S,E_S')$ with $ec(G')=\lambda$, a cost function $c':V_S\times V_S\rightarrow Z^+\cup\{\infty\}$, and a backpointer $b:V_S\times V_S\rightarrow V\times V$. */
/* *Output*: A set E_S'' of edges, each connecting distinct vertices of V_S, such that E_S'' is a solution to G_S'. */
1. Construct a complete graph G_b from G_S, and a tree G_b' from G_S' by using **REMAKE**.
2. Compute $d:V_b\times V_b\rightarrow Z^+$ and $b':V_b\times V_b\rightarrow V_b\times V_b$ by using **DIST**.
3. $A_b\leftarrow\varnothing$. Insert <u,v> and <v,u> into A_b for each pair $\{u,v\}\in V_b\times V_b$, constructing a complete directed graph $\overline{G}_b=(V_b,A_b)$.
4. Choose any degree-1 vertex of G_b' as the root r. Let A_b' be the set of directed edges generated by directing each edge in E_b' toward r. Denote the resulting graph by $\overline{G}_b'=(V_b,A_b')$.
5. $d'<u,v>\leftarrow d(u,v)$ and $d'<v,u>\leftarrow d(u,v)$ for all $\{u,v\}\in V_b\times V_b$.
6. For $<u,v>\in A_b$, if $u\notin V_S$ or $v\notin V_S$ then $d'<u,v>\leftarrow\infty$; if $<u,v>\in A_b'$ and $\{u,v\}\subseteq V_S$ then $d'<u,v>\leftarrow 0$ and $d<v,u>\leftarrow\infty$; if v=r then $d'<u,r>\leftarrow\infty$.
7. Find a minimum-cost arborescence $T=(V_b,A_b'')$ with the root r of $\overline{G}_b$ with respect to d'.
8. For each edge <u,v> in A_b'' with $0< d'<u,v> <\infty$, insert the corresponding edge b'(u,v) into E_S'' (with multiplicity deleted).

Remark 4.1. In Step 5 of FSA** we may define d' and b'' as in Step 5 of FSM** (to be given in 4.2) by finding shortest paths. We implemented this version and applied it to 120 data, but no improvement has been observed so far. Hence in this paper Step 5 of FSA** is left as it is. ♦

The following lemma can be proved.

Lemma 4.1. FSA** generates a set of directed edges A_b'' such that $(V_b,A_b'\cup A_b'')$ is strongly connected. ♦

We obtain the next theorem.

Theorem 4.1. GS-A generates a set E'' such that $\lambda(G'+E'')\geq 3$ in $O(|V|^3)$ time ♦

4.2 Procedure FSM** Based on Maximum-Cost Matching

The second procedure to be proposed is FSM** based on a maximum-cost matching algorithm. The idea is very simple. As a set of $\lceil|L_3|/2\rceil$ edges, each connecting a pair of leaves in a cactus G_S', the one E_0'' with minimum total cost is selected by using a maximum-cost matching algorithm in $O(|V||E|)$ time, which is $O(|V|^3)$. If $\lambda(G_S'+E_0'')<3$ then similar process will be repeated: repetition is at most $O(log|V|)$ time. The description of FSM** is given as follows. We denote $J_i'=(Y_i,B_i')$ and $H_i'=(W_i,F_i')$.

procedure FSM**;
/* *Input*: a graph $G_S'=(V_S,E_S')$, a cost function

$c': V_S \times V_S \to Z^+ \cup \{\infty\}$, and a backpointer $b: V_S \times V_S \to E$. */
/* *Output*: A set of edges E_S" that is a solution to G_S'. */

1. $J_0 \leftarrow G_S$, $J_0' \leftarrow G_S'$, $c_0' \leftarrow c'$, $i \leftarrow 0$.
2. Construct a tree G_b' from J_i' by using REMAKE.
3. $H_i' \leftarrow G_b'$.
4. Find a distance function $d_i: W_i \times W_i \to Z^+ \cup \{\infty\}$ with a back pointer $b_i': W_i \times W_i \to Y_i \times Y_i$ by using DIST. For each dummy vertex w_C added within the cycle C of G_S', $d_i(w_C, v) \leftarrow \infty$ for any vertex v on C.
5. Compute $d_i': W_i \times W_i \to Z^+ \cup \{\infty\}$ and $b_i'': W_i \times W_i \to W_i \times W_i$ by finding, for each pair u,v of degree-1 vertices of H_i', a shortest path P(u,v) in a complete graph on W_i as follows: for the edge set E_P of P(u,v), $d_i'(u,v) \leftarrow d_i(E_P)$ and $b_i''(u,v) \leftarrow E_P$ (actually $b_i''(u,v)$ is a pointer to the list maintaining E_P).
6. Construct a complete subgraph S_i on the set of all degree-1 vertices of H_i'.
7. For each cost $d_i'(u,v)$ with $(u,v) \in E(S_i)$, $d_i''(u,v) \leftarrow MAX+1-d_i'(u,v)$, where MAX is the maximum edge-cost of S_i. Find a maximum-cost matching $M_i \subseteq E(S_i)$ of S_i with respect to d_i''.
8. For each edge $(u,v) \in M_i$, insert the corresponding set of edges $b_i'(b_i''(u,v))$ into E_i" (and then any multiplicity is deleted).
9. If E_i" is not a solution to J_i' then $\Delta \leftarrow E_i$" and execute (i)-(iii):
(i) $J_{i+1}' \leftarrow$ the cactus constructed by shrinking each of 3-eccs of $J_i'+E_i$" into a vertex:
(ii) define a new cost function $c_{i+1}: V_{i+1} \times V_{i+1} \to Z^+ \cup \{\infty\}$ by

$$c_{i+1}'(u,v) = \min\{c_i'(x,y) \mid (x,y) \in V_i \times V_i,\ x \in S_u,\ y \in S_v\},$$

where S_w denotes the set of vertices of $J_i'+\Delta$ that are shrunk into $w \in Y_{i+1}$ and the edge (x,y) is referenced by a backpointer $b_{i+1}(u,v) = (x,y)$, where $b_{i+1}: V_{i+1} \times V_{i+1} \to V_i \times V_i$.
(iii) $i \leftarrow i+1$ and goto step 2.
10. If E_i" is a solution to J_i' then do the following (i) and (ii):
(i) $E_S'' \leftarrow E_0$";
(ii) if $i \geq 1$ then for each j (j=1, ..., i) repeat $E_S'' \leftarrow E_S'' \cup b_1(...(b_j(E_j''))...)$. ♦

We obtain the next theorem.

Theorem 4.2. GS-M generates a set E" such that $\lambda(G'+E'') \geq 3$ in $O(|V|^3 log|V|)$ time if Step 4(3) is omitted or in $O(|V|^4)$ time otherwise. ♦

4.3 Procedure SMC** Based on Minimum-Cost Edges

The third procedure to be proposed is SMC**, which is a greedy algorithm that finds approximate solutions without using a maximum-cost matching algorithm. The description of SMC** is almost the same as that of FSM**. The only difference is that SMC** finds a solution to G_S' by choosing, at each leaf v, a minimum-cost edge e_v among those incident upon v.

Description of SMC.** Delete Step 5, replace Step 7 by the following statement in FSM** and rewrite $b_i'(b_i''(u,v))$ as $b_i'(u,v)$ in Step 8, and SMC** follows:

Step 7. For each degree-1 vertex v of H_i', let e_v denote an edge connecting a pair $(u,v) \in W_i \times W_i$ with

$d_i'(u,v) = \min\{d_i'(u',v) \mid (u',v) \in W_i \times W_i\}$, and
$M_i \leftarrow \{e_v \mid v$ is a degree-1 vertex of $H_i'\}$
(with multiplicity deleted). ♦

We obtain the next theorem.

Theorem 4.3 GS-S generates a set E" such that $\lambda(G'+E'') \geq 3$ in $O(|V|^3)$ time. ♦

4.4 Procedure HBD** by Combination of FSM** and SMC**

HBD** is a combination of FSM** and SMC**, and is almost the same as that of FSM**. The only difference is that a maximum-cost matching algorithm, to be repeatedly used in finding a solution to G_S', is applied to the set $E(K_i) = \{e_v \mid v$ is a degree-1 vertex of $G_S'\}$ (with multiplicity deleted) instead of the set of all edges connecting degree-1 vertices of G_S'.

Description of HBD.** If we replace Step 7 of FSM** by the following statement then the description of HBD** is obtained:
Step 7. For each degree-1 vertex v of H_i', let f_v denote an edge $(u,v) \in E(S_i)$ with $d_i'(u,v) = \min\{d_i'(u',v) \mid (u',v) \in E(S_i)\}$, and $E(K_i) \leftarrow \{f_v \mid v$ is a degree-1 vertex of $H_i'\}$ (with multiplicity deleted). Let K_i denote the subgraph $(V(S_i), E(K_i))$ of S_i. For each cost $d_i'(u,v)$ with $(u,v) \in E(K_i)$, $d_i''(u,v) \leftarrow MAX+1-d_i'(u,v)$, where MAX is the maximum edge-cost of K_i. Find a maximum-cost matching $M_i \subseteq E(K_i)$ of K_i with respect to d_i''. ♦

We obtain the next theorem.

Theorem 4.4. GS-H generates a set E" such that $\lambda(G'+E'') \geq 3$ in $O(|V|^3 log|V|)$ time. ♦

5 EVALUATION OF WORST APPROXIMATION

Worst approximation by proposed algorithms is evaluated theoretically for unweighted cases and experimentally for weighted cases. Let OPT or APP denote total cost of optimum or approximate solutions to 3-ECA-SV, respectively. The ratio APP/OPT is going to be evaluated concerning USV and WSV with $\lambda(G')=2$.

5.1 Theoretical Evaluation for USV

It sufficies to consider UW-3-ECA(M), for which an optimum solution can be obtained in $O(|V|+|E'|)$ time, and it is shown in [19,20,21] that $OPT = \lceil q/2 \rceil$, where $q=|L_3|$ and we assume $c(e)=1$ for any $e \in E-E'$. Clearly $OPT=q/2$.
(1) GS-A. A minimum arborescence to be found by FSA** has total cost at most q-1, and we have $APP \leq q-1$. Therefore $APP/OPT \leq (q-1)/(q/2) = 2-2/q < 2$.
(2) GS-M and GS-H. Let G_0 denote a complete graph on $n_0(=q)$ degree-2 vertices of G_S'. Let M_0 be a maximum matching of G_0, and $e_0 = |M_0| = \lfloor n_0/2 \rfloor \leq q/2$. $G_1 = G_0 + M_0$ has at most $n_1 = \lceil n_0/2 \rceil \leq q/2 + 1/2$ degree-2 vertices. In general, for $i=1,...m = \lceil log q \rceil - 1$, $G_i = G_{i-1} + M_{i-1}$ has at most $n_i = \lceil n_{i-1}/2 \rceil$ degree-2 vertices, where *log*q is abbreviation of $log_2 q$ and

$$n_i = \lceil n_{i-1}/2 \rceil \leq q/(2^i) + 1/(2^i) + ... + 1/2.$$

If M_i is a maximum matching of G_i then its cardinality $e_i=|M_i|=\lfloor n_i/2 \rfloor$, where

$\lfloor n_i/2 \rfloor \le q/(2^{i+1})+1/(2^{i+1})+...+1/(2^2)$.

Hence $G_{i+1}=G_i+M_i$ has at most $n_{i+1}=\lceil n_i/2 \rceil$ degree-2 vertices. At the final stage,

$n_m=\lceil n_{m-1}/2 \rceil \le q/(2^m)+1/(2^m)+...+1/2=3-2/n<3$

and $n_m=2$. Hence $e_m=1$ and $n_{m+1}=0$. Thus the total number of edges added is

$e_0+e_1+...+e_{m-1}+e_m$

$\le (q/2)+\{q/(2^2)+1/(2^2)\}+...+\{q/(2^m)+1/(2^m)+...+1/(2^2)\}+1$

$=q-5/2+(1/2)\log q+2/q$.

We can prove that

$APP/OPT \le (q-5/2+(1/2)\log q+2/q)/(q/2)=2-5/q+(\log q)/q+4/(q^2)$

and

$\lim_{q\to\infty}(APP/OPT) \le 2$.

It seems that this is slightly overestimated, since only APP/OPT<2 has been observed so far in our experimentation by FSM** and HBD**.

5.2 Experimental Evaluation for WSV

Experimental evaluation of the four algorithms for WSV is given. It suffices to consider FSA**, FSM**, SMC** and HBD** for W-3-ECA(M).

(1) Input data. We explain how input data, G' and c, are constructed as in Steps 1-3.

1. The number |V| of vertices is given as follows:

$|V| \in \{10,15,20,40,60,80,100,120,140,160,180,200\}$.

2. Two types of data are provided: type C and type T constructed as follows.

2.1. (type C)

(1) Partition $V=\{1,...,|V|\}$ into four sets $L_1,...,L_4$ with $|L_i|=|V|/4$ if $|V| \ge 40$, where if $|V| \le 20$ then V is partitioned into at most two sets.

(2) For each i, construct a cycle C_i of $|L_i|$ vertices.

(3) Add k_i edges to C_i randomly, where k_i satisfies $1 \le k_i \le |L_i|/2$ and is given randomly.

(4) Let G_i be the resulting graph. Construct G_{12} (G_{34}, respectively) by coalescing a pair of vertices, one from G_1 (G_3) and the other from G_2 (G_4). Let G be the graph given by coalescing a pair of vertices, one from G_{12} and the other from G_{34}. All pairs of vertices are chosen randomly.

2.2. (type T)

(1) Partition $V=\{1,...,|V|\}$ into k sets $L_1,...,L_k$, where k is given randomly.

(2) For each i, $1 \le i \le k$, choose two vertices $w_0, w_1 \in L_i$, and add two edges (w_0,v), (w_1,v) for each $v \in L_i-\{w_0,w_1\}$. Let G_i denote the resulting graph, and let $R=\{G_1,...,G_k\}$.

(3) Choose any $G_i \in R$, $R \leftarrow R-\{G_i\}$ and $G \leftarrow G_i$. Then repeat the following (i)-(iii) until R becomes empty: (i) choosing $G_j \in R$, (ii) $R \leftarrow R-\{G_j\}$, and (iii) $G \leftarrow$ the graph given by coalescing a pair of vertices, one from G and the other from G_j, where a pair of vertices are chosen randomly.

3. $(|\Gamma|/|V|)\times 100 \in \{10,20,...,90\}(\%)$ is randomly chosen, and vertices of Γ are also randomly selected.

4. Costs on edges are selected randomly from the set $\{1,...,99\}$ of integers.

(2) Experimental results. We have tried 4240 data so far concerning WSV with $\lambda(G')=2$. A workstation SUN SPARC STATION is used. Tables 1 and 3 show a part of our experimental results on GS-A, GS-M, GS-S and GS-H. For 1000 data with |V|=10,15,20 among them, optimum solutions are sought by exhaustive search. Tables 2, 4 and 5 show some statistical data on approximate solutions. Table 2 shows error=(AP/OP-1)×100(%) for 500 data of type C in (1) or for those of type T in (2); average of the ratio cost(*)/OPT over 500 data for each of types C and T in (3), where * is any one of GS-A, GS-M, GS-S and GS-H. Table 4 shows average of the ratio cost(*)/cost(GS-M) over 2120 data for each of types C and T. Table 5 shows distribution of these ratios concerning 2120 data for each type: type C in (1) and type T in (2). Main points shown by our experimental results are the following (1)-(3).

(1) For almost all data,

cost(GS-M)<cost(GS-H)<cost(GS-S)<cost(GS-A)

is observed. Especially for data of type T, optimum or nearly optimum solutions are obtained by GS-M.

(2) We have 1000 data to each of which an optimum solution is found by exhaustive search. For 414 data (82.80%) of type C and 459 data (91.80%) of type T, GS-M generates approximate solutions with errors (APP/OPT-1)×100≤10%. In total GS-M generates such approximate solutions for 873 data (87.30%) among 1000 data.

(3) In general,

time(GS-S) <time(GS-A)<time(GS-H)<time(GS-M)

is observed.

5.3 Unbounded approximation. We show two examples for which FSA**, FSM** or SMC** generates solutions to W-3-ECA(M) such that APP/OPT fails to be bounded by constants. For the graphs shown by solid lines in Figs. 7 and 8, suppose that they represent G_s' and that all costs except those specified in the figures are very large, where M>5 in Fig. 7. For Fig. 7, an optimum solution is shown in halftone lines and OPT=5. FSA** finds a solution with total cost APP=2M+5, and APP/OPT=2M/5+1. 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.

For Fig. 8, we have OPT=2 (the edge (a,g)), while SMC** generates a solution shown by halftone lines and APP=|V|-1(=6). Hence APP/OPT=(|V|-1)/2 (=3).

It is observed in our experimentation that FSM** produces good solutions for data to which SMC** gives unbounded solutions, and vice versa. HBD** is proposed in order to overcome these unbounded cases: it finds an optimum solution to each of these two cases, and good approximate ones are produced in our experiment.

6. CONCLUDING REMARKS

This paper proposed four approximation algorithms GS-A, GS-M, GS-S and GS-H for 3-ECA-SV(M), and both theoretical and experimental evaluation of their approximate solutions are given. The following (1) through (3) are left for future research:

(1) theoretical evaluation of HBDM, which is being continued (our conjecture is APP/OPT≤2);

(2) proposing approximation algorithms for k-ECA-SV(M);

(3) providing more experimental results on 3-ECA-SV(M) with $\lambda+\delta=3$ and $\delta \ge 2$.

REFERENCES

1. Eswaran, K.P. and Tarjan, R.E., Augmentation problems,

SIAM J.Comput, 5 (1976), pp.653-655.
2. Frank, A., Augmenting graphs to meet edge connectivity requirements, SIAM J. Discrete Mathematics, Vol. 5, No. 1(February 1992), pp.25-53.
3. Frederickson, G.N. and Ja'ja', J., Approximation algorithms for several graph augmentation problems, SIAM J.Comput., 10 (1981), pp.270-283.
4. Gabow., H.N., Applications of a poset representation to edge connectivity and graph rigidity, Proc. 32nd IEEE Symp. Found. Comp. Sic., 812-821(1991).
5. Gabow., H.N., Galil, Z., Spencer, T. and Tarjan, R.E., Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, Combinatorica, 6(2) (1986), pp.109-122.
6. Galil, Z. and Italiano, G.F., Reducing edge connectivity to vertex connectivity, SIGACT NEWS, 22 (1991), pp.57-61.
7. Mashima, T., Taoka, S. and Watanabe, T., Approximation algorithms for the k-edge-connectivity augmentation problem, Tech. Rep. IEICE, COMP92-24(July 1992), pp.11-20.
8. Nagamochi, H. and Ibaraki, T., A linear time algorithm for computing 3-edge-connected components in a multigraph, Japan J. Industrial and Applied Math., Vol.9, No.7(June 1992), pp.163-180.
9. Naor, D., Gusfield, D. and Martel, C., A fast algorithm for optimally increasing the edge-connectivity, Proc. 31st Annual IEEE Symposium on Foundations of Computer Science(1990), pp.698-707.
10. Taoka, S., Watanabe, T. and Onaga, K., A linear time algorithm for computing all 3-edge-connected components of an multigraph, Trans. IEICE, Vol.E75-A, No.3(March 1992), pp.410-424.
11. Tarjan, R.E., Data Structures and Network Algorithms, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, PA (1983).
12. Ueno, S., Kajitani, Y., and Wada, H., The minimum augmentation of trees to k-edge-connected graphs, Networks, 18(1988), pp.19-25.
13. Watanabe, T., Higashi, Y. and Nakamura, A., An approach to robust network construction from graph augmentation problems, Proc. of 1990 IEEE Int. Symp. on Circuits and Systems (May, 1990), pp. 2861-2864.
14. Watanabe, T, Higashi, Y. and Nakamura, A., Construction robust networks by means of graph augmentation problems, Trans. IEICE of Japan, Vol. 73-A, No. 7 (July 1991), pp.1242-1254. (Also see Electronics and Communications in Japan, Part 3, Vol. 74, No. 2 (1991), pp.79-96.)
15. Watanabe, T., Mashima, T. and Taoka, S., The k-edge-connectivity augmentation problem of weighted graphs, Proc. 3rd International Symposium on Algorithms and Computation(ISAAC'92)(Dec. 1992), to appear.
16. Watanabe, T. and Nakamura, A., Edge-connectivity augmentation problems, Journal of Computer and System Sciences, 35 (1987), pp.96-144.
17. Watanabe, T., Narita, T. and Nakamura, A., 3-Edge-connectivity augmentation problems, Proc. 1989 IEEE ISCAS(1989), pp.335-338.
18. Watanabe, T., Taoka, S. and Mashima, T., Approximation algorithms for the 3-edge-connectivity augmentation problem of graphs, Proc. 1st IEEE Asia-Pacific Conference on Circuits and Systems(APCCAS'92)(Dec. 1992), to appear.
19. Yamakado, M., and Watanabe, T., A Linear-Time Augmenting Algorithm for the 3-Edge-Connectivity Augmentation Problem, IEICE of Japan, Tech. Research Rep. COMP90-57(November, 1990), pp. 41-49.
20. Watanabe, T., Yamakado, M. and Onaga, K., A linear-time augmenting algorithm for 3-edge-connectivity augmentation problems, Proc. 1991 IEEE ISCAS(1991), pp.1168-1171.
21. Watanabe, T. and Yamakado, M., A linear time algorithm for smallest augmentation to 3-edge-connect a graph, Trans. IEICE of Japan, submitted(September, 1992).

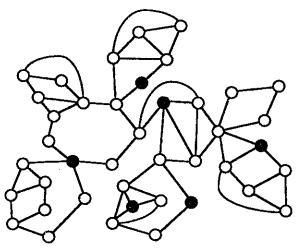

Fig. 1. A graph G'=(V,E') with $\lambda(G')=2$ and $\Gamma \subseteq V$, where vertices of Γ are denoted as black spots.

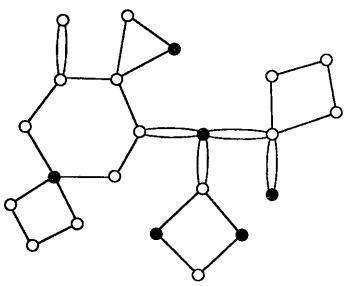

Fig. 2. The cactus $G_S'=(V_1,E_1')$ and $V_a \subseteq V_1$, where vertices of Va are denoted as black spots.

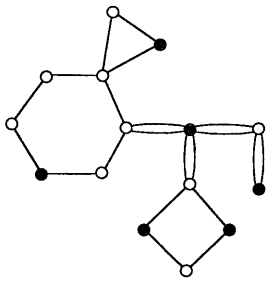

Fig.3. The cactus $G_2'=(V_2,E_2')$ and $V_a \subseteq V_2$.

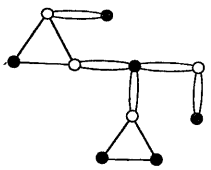

Fig. 4. The condensation $G_3'=(V_3,E_3')$ of G_2' and $V_a \subseteq V_3$.

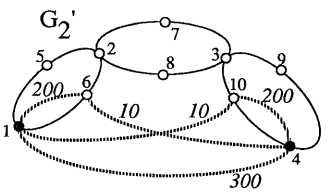

Fig. 5. Another example of a cactus $G_2'=(V_2,E_2')$ with $V_a=\{1,4\}$, having a crossing pair $\{\{1,10\},\{6,4\}\}$.

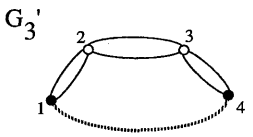

Fig. 6. The condensation G_3' of G_2' shown in Fig. 5.

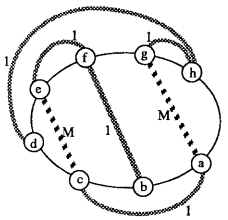

Fig. 7. An example for which each of GS-A and GS-M generates a solution whose total cost cannot be bounded by a constant.

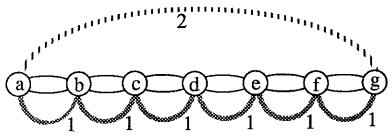

Fig. 8. An example for which GS-S generates a solution whose total cost cannot be bounded by a constant.

Table 1. A part of our experimental results for data with |V|=20: (1) type C; (2) type T, where optimum solutions are sought by exhaustive search. The columns, cost, AP/OP and time denote total costs of solutions, ratio of costs APP/OPT and CPU time in 1/60 second, respectively.

(1) type C

#	|V|	|E'|	|Γ|	GS-A			GS-M			GS-S			GS-H			OPT	
				cost	AP/OP	time	cost	AP/OP	time	cost	AP/OP	time	cost	AP/OP	time	cost	time
25	20	26	2	3	1	5	3	1	4	3	1	3	3	1	4	3	7
			4	9	1	5	9	1	5	9	1	4	9	1	6	9	95
			6	24	1	7	24	1	8	24	1	5	24	1	7	24	2882
			8	25	1.1905	6	23	1.0952	5	23	1.0952	5	22	1.0476	7	21	218
			10	28	1.12	7	26	1.04	7	25	1	5	26	1.04	6	25	304
			12	28	1.12	7	26	1.04	7	25	1	5	26	1.04	6	25	287
			14	38	1.1875	11	36	1.125	10	36	1.125	6	35	1.0938	10	32	1022
			16	42	1.1053	11	38	1	10	42	1.1053	7	39	1.0263	8	38	1276
			18	54	1.0588	13	51	1	11	51	1	7	51	1	10	51	6609
30	20	26	2	4	1	4	4	1	4	4	1	4	4	1	3	4	5
			4	4	1	5	4	1	3	4	1	2	4	1	3	4	5
			6	17	1	5	25	1.4706	4	29	1.7059	3	25	1.4706	5	17	13
			8	18	1	5	18	1	4	18	1	2	18	1	4	18	20
			10	30	1.6667	5	18	1	4	30	1.6667	4	18	1	5	18	41
			12	30	1.6667	4	18	1	5	30	1.6667	3	18	1	5	18	41
			14	30	1	5	30	1	5	30	1	4	30	1	4	30	78
			16	38	1	8	38	1	7	38	1	4	38	1	7	38	60
			18	38	1	8	38	1	6	38	1	4	38	1	6	38	60

(2) type T

#	|V|	|E'|	|Γ|	GS-A			GS-M			GS-S			GS-H			OPT	
				cost	AP/OP	time	cost	AP/OP	time	cost	AP/OP	time	cost	AP/OP	time	cost	time
22	20	36	2	3	1	5	3	1	5	3	1	4	3	1	4	3	7
			4	4	1	5	5	1.25	4	5	1.25	4	5	1.25	6	4	21
			6	8	1.3333	5	6	1	5	6	1	5	6	1	5	6	14
			8	33	1.1	6	30	1	6	30	1	5	33	1.1	7	30	521
			10	33	1	8	33	1	7	33	1	6	33	1	7	33	885
			12	12	1	7	12	1	6	12	1	5	12	1	6	12	27
			14	67	1.0469	9	64	1	10	64	1	6	67	1.0469	10	64	18384
			16	53	1.0816	11	55	1.1224	11	49	1	8	58	1.1837	11	49	1143
			18	70	1.0606	13	66	1	12	66	1	8	75	1.1364	12	66	7379
30	20	36	2	1	1	4	1	1	5	1	1	4	1	1	3	1	7
			4	1	1	4	1	1	4	1	1	4	1	1	4	1	7
			6	4	1	6	4	1	5	5	1.25	5	4	1	6	4	60
			8	7	1	7	7	1	8	8	1.1429	6	7	1	6	7	475
			10	8	1	8	8	1	8	9	1.125	6	8	1	7	8	1097
			12	10	1.1111	8	9	1	7	10	1.1111	6	9	1	7	9	3386
			14	12	1.2	9	10	1	8	12	1.2	6	10	1	9	10	16003
			16	12	1.0909	11	11	1	10	13	1.1818	7	11	1	10	11	14388
			18	12	1.0909	12	11	1	11	13	1.1818	9	11	1	12	11	7700

Table 2. Distribution of error=(AP/OP-1)×100(%) for 500 data of type C in (1) or for those of type T in (2); average of the ratio cost(*)/OPT over 500 data for each of types C and T in (3), where * is any one of GS-A, GS-M, GS-S and GS-H. In each column (except the leftmost) in (1) and (2), the total number of cases with the corresponding error is on the left and its ratio on the right.

(1) type C

err.	GS-A		GS-M		GS-S		GS-H	
err.=0%	219	43.80%	342	68.40%	236	47.20%	295	59.00%
0%<err.≤5%	24	4.80%	30	6.00%	38	7.60%	63	12.60%
5%<err.≤10%	30	6.00%	42	8.40%	35	7.00%	47	9.40%
10%<err.≤15%	24	4.80%	25	5.00%	35	7.00%	32	6.40%
15%<err.	203	40.60%	61	12.20%	156	31.20%	63	12.60%

(2) type T

err.	GS-A		GS-M		GS-S		GS-H	
err.=0%	256	51.20%	424	84.80%	349	69.80%	358	71.60%
0%<err.≤5%	22	4.40%	20	4.00%	11	2.20%	28	5.60%
5%<err.≤10%	25	5.00%	15	3.00%	28	5.60%	39	7.80%
10%<err.≤15%	55	11.00%	26	5.20%	44	8.80%	45	9.00%
15%<err.	142	28.40%	15	3.00%	68	13.60%	30	6.00%

(3) type C and type T

	GS-A	GS-M	GS-S	GS-H	#data
type C	1.1694	1.0547	1.1475	1.0575	500
type T	1.1021	1.0173	1.0524	1.0322	500
all	1.1357	1.0360	1.0999	1.0448	1000

Table 3. A part of our experimental results for data with $180 \leq |V| \leq 200$: (1) type C; (2) type T, where the columns cost, ratio and time denote total costs of solutions, cost(*)/cost(GS-M) and CPU time in 1/60 second, respectively, where * denotes any one of GS-A, GS-M, GS-S and GS-H.

(1) type C

#	|V|	|E'|	|Γ|	GS-A			GS-M			GS-S			GS-H		
				cost	ratio	time	cost	ratio	time	cost	ratio	time	cost	ratio	time
105	180	256	18	10	1.1111	40	9	1	34	10	1.1111	31	9	1	35
			36	22	1.1	83	20	1	84	21	1.05	63	20	1	91
			54	20	1	83	20	1	86	24	1.2	65	20	1	105
			72	40	1.0256	195	39	1	222	43	1.1026	149	39	1	355
			90	51	1.0625	203	48	1	227	54	1.125	166	48	1	436
			108	59	1.0926	227	54	1	264	58	1.0741	181	54	1	507
			126	65	1.1207	250	58	1	286	62	1.069	196	59	1.0172	443
			144	75	1.2097	382	62	1	381	70	1.129	260	62	1	445
			162	78	1.2188	355	64	1	458	71	1.1094	269	65	1.0156	426
112	200	235	20	18	1	595	18	1	614	22	1.2222	531	18	1	787
			40	39	1.1818	808	33	1	851	36	1.0909	739	32	0.9697	1001
			60	52	1.1818	1012	44	1	1135	49	1.1136	985	43	0.9773	1292
			80	58	1.1837	1123	49	1	1261	54	1.102	1048	50	1.0204	1804
			100	75	1.2712	1220	59	1	1438	67	1.1356	1158	59	1	1450
			120	87	1.2794	1373	68	1	1646	76	1.1176	1313	67	0.9853	1580
			140	99	1.2532	1342	79	1	1757	86	1.0886	1222	79	1	1786
			160	110	1.2791	1435	86	1	1740	94	1.093	1275	86	1	1655
			180	125	1.2376	1557	101	1	1971	112	1.1089	1366	102	1.0099	1783

(2) type T

#	|V|	|E'|	|Γ|	GS-A			GS-M			GS-S			GS-H		
				cost	ratio	time	cost	ratio	time	cost	ratio	time	cost	ratio	time
110	180		18	19	1.1176	70	17	1	60	20	1.1765	53	17	1	67
			36	38	1.0556	109	36	1	102	38	1.0556	79	36	1	132
			54	65	1.0656	202	61	1	202	61	1	144	61	1	278
			72	78	1.0986	283	71	1	316	71	1	214	72	1.0141	365
			90	94	1.1605	370	81	1	471	82	1.0123	285	83	1.0247	451
			108	108	1.1489	613	94	1	677	97	1.0319	491	94	1	763
			126	129	1.1944	788	108	1	947	114	1.0556	673	109	1.0093	837
			144	150	1.2295	1008	122	1	1602	127	1.041	874	123	1.0082	1060
			162	172	1.2374	1348	139	1	1999	150	1.0791	1194	144	1.036	1557
115	200		20	18	1	82	18	1	76	18	1	63	18	1	127
			40	35	1.0606	161	33	1	149	34	1.0303	115	33	1	217
			60	50	1.087	256	46	1	274	46	1	187	46	1	385
			80	66	1.2	363	55	1	374	58	1.0545	278	55	1	396
			100	84	1.2353	523	68	1	638	71	1.0441	430	69	1.0147	571
			120	98	1.2564	701	78	1	955	80	1.0256	587	78	1	788
			140	117	1.3295	1016	88	1	1314	96	1.0909	854	90	1.0227	1081
			160	127	1.3093	1263	97	1	1718	104	1.0722	1109	101	1.0412	1419
			180	142	1.3786	1666	103	1	2141	112	1.0874	1547	108	1.0485	1790

Table 4. Average of the ratio cost(*)/cost(GS-M) over 2120 data for each of types C and T,where * denotes any one of GS-A, GS-M, GS-S and GS-H.

type C and type T

	GS-A	GS-M	GS-S	GS-H	#data
type C	1.1203	1.0000	1.1016	1.0058	2120
type T	1.1320	1.0000	1.0483	1.0127	2120
all	1.1262	1.0000	1.0749	1.0092	4240

Table 5. Distribution of the ratio cost(*)/cost(GS-M) concerning 2120 data for each type: type C in (1) and type T in (2). Each column (except the leftmost) shows total number of data on the left and its ratio on the right.

(1) type C

R=cost(*)/cost(GS-M)	GS-A		GS-M		GS-S		GS-H	
R<1	45	2.12%	0	0.00%	63	2.97%	196	9.25%
R=1	452	21.32%	2120	100.00%	436	20.57%	1346	63.49%
1<R≤1.05	164	7.74%	0	0.00%	273	12.88%	462	21.79%
1.05<R≤1.10	373	17.59%	0	0.00%	486	22.92%	76	3.58%
1.10<R	1086	51.23%	0	0.00%	862	40.66%	40	1.89%

(2) type T

R=cost(*)/cost(GS-M)	GS-A		GS-M		GS-S		GS-H	
R<1	28	1.32%	0	0.00%	70	3.30%	53	2.50%
R=1	409	19.29%	2120	100.00%	708	33.40%	1258	59.34%
1<R≤1.05	80	3.77%	0	0.00%	441	20.80%	656	30.94%
1.05<R≤1.10	321	15.14%	0	0.00%	590	27.83%	115	5.42%
1.10<R	1282	60.47%	0	0.00%	311	14.67%	38	1.79%