BIT 32 (1992), 403–412.

THE MOST VITAL EDGES WITH RESPECT TO THE NUMBER OF SPANNING TREES IN TWO-TERMINAL SERIES-PARALLEL GRAPHS

RONG-HONG JAN, LIH-HSING HSU and YUEH-YING LEE

Department of Information and Computer Science, National Chiao Tung University Hsinchu 30050, Taiwan, R.O.C.

Abstract.

A set E' of k edges in a multigraph $G = (V, E)$ is said to be a k most vital edge set (k-MVE set) if these edges being removed from G, the resultant graph $G' = (V, E - E')$ has minimum number of spanning trees. The problem of finding a k-MVE set for two-terminal series-parallel graphs is considered in this paper. We present an $O(|E|)$ time algorithm for the case $k = 1$, and an $O(|V|^k + |E|)$ time algorithm for arbitrary k.

C.R. Category: G.2.2.

Keywords: most vital edges, spanning trees, two-terminal series-parallel graphs.

1. Introduction and definitions.

In many applications the network designer may want to know which edges in the network are most important to him; if these edges are removed from the network there will be a significant decrease in its performance. Such edges are called the most vital edges in a network. Several papers [2, 4, 7] have been presented to find the most vital edges. However, they only address the effect on the shortest path or the maximum flow in the network. For example, the problem considered in [4] is to find a set of edges in a network whose removal from the network results in the greatest increase in the length of the shortest path between two specified nodes. In this paper, we will consider the effect on the number of spanning trees in the network.

The number of spanning trees is an important parameter in design and analysis of network reliability [3]. A computer communication network can be modelled as a graph with p vertices and q edges; vertices represent computers in the network, and edges represent communication links. Assume that the vertices are perfectly reliable and each edge may work (or fail) independently with the probability ρ (or $1 - \rho$).

Received March 1991. Revised December 1991.

Therefore, if ρ is close to zero, the all-terminal reliability $R(G)$ of the network G can be estimated by

$$R(G) \sim \tau(G)\rho^{p-1}(1-\rho)^{q-p+1}$$

where $\tau(G)$ is the number of spanning trees of G.

Most graph-theoretic terms used in this paper are standard (e.g., [6]). Here, we limit ourselves to defining the most commonly used terms and those that may produce confusion. Let $G = (V, E)$ be a *multigraph*, where V is the *vertex set* of G and E is the *edge set* of G. Let $p = |V|$ and $q = |E|$. A *spanning subgraph* of $G = (V, E)$ is a subgraph $H = (V, E')$, where $E' \subset E$. A *simple graph* is a graph G with the multiplicity of every edge in G equal to one.

A spanning subgraph H of a connected graph G is called a *spanning tree* of G if H forms a tree. There are usually many spanning trees in a connected graph. We use $\tau(G)$ to denote the number of spanning trees in G. Let E' be a subset of E in graph $G = (V, E)$. We use $G - E'$ to denote the graph $G' = (V, E - E')$. In particular, we use $G - e$ to denote the graph $G - \{e\}$. An edge e^* is called a *most vital edge* in graph G, if $\tau(G - e^*) \leq \tau(G - e)$ for all edges $e \in E$. The problem of finding such an edge is called the 1-MVE problem. A subset E^* of E with $|E^*| = k$ is called a *k most vital edge set* (k-MVE set) of G if $\tau(G - E^*) \leq \tau(G - E')$ for all $E' \subset E$ with $|E'| = k$. The problem of finding such a subset is called the k-MVE problem.

In this paper, both the 1-MVE problem and k-MVE problem are considered for a special class of graphs called two-terminal series-parallel graphs [5]. The formal description of two-terminal series-parallel graphs will be given in the next section. Many NP-complete problems in general graphs can be solved by polynomial-time algorithms in this class of graphs [8, 11]. We are going to present an $O(q)$ time algorithm to solve the 1-MVE problem and an $O(p^k + q)$ time algorithm to solve the k-MVE problem.

2. Two-terminal series-parallel graphs

Two-terminal series-parallel graphs are derived from the concept of series-parallel networks (SPN). Series-parallel networks often occur in practice since they correspond to boolean formulas (with serial connection implementing logical-AND and parallel connection implementing logical-OR). For example, an SPN shown in Fig. 1, corresponds to the boolean formula $[a \vee b] \wedge [c \vee (d \wedge (e \vee f)) \vee (g \wedge (h \vee i \vee j))]$. An SPN can be represented by a two-terminal series-parallel graph, an edge-labelled graph with two distinguished vertices called *terminals*. Formally, a *two-terminal series-parallel* (TTSP) *graph* of *type k*, where $k \in \{L, P, S\}$ (which mean *leaf*, *parallel* and *series*), is recursively defined as follows.

1. A graph G, which contains only one edge labelled by e_x with its two endpoints as terminals, is a TTSP graph of type L.
2. If the TTSP graphs $G_i, i = 1, 2, \ldots, h$, with terminals (a_i, b_i) are not of type P, then

by identifying all a_i together, we get a new vertex a and identifying all b_i together, we get a new vertex b. The resultant graph G is a TTSP graph of type P with two terminals (a, b).

3. If the TTSP graphs G_i, $i = 1, 2, \ldots, h$, with terminals (a_i, b_i) are not of type S, then by identifying terminals b_i and a_{i+1} together, for $i = 1, 2, \ldots, h-1$, we get a graph G with terminals (a_1, b_h). The resultant graph G is a TTSP graph of type S with two terminals (a_1, b_h).

For example, Fig. 2(a) shows a TTSP graph representing the SPN shown in Fig. 1.

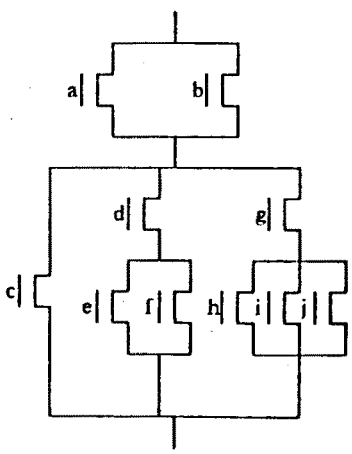


Figure 1 An S-type series-parallel network

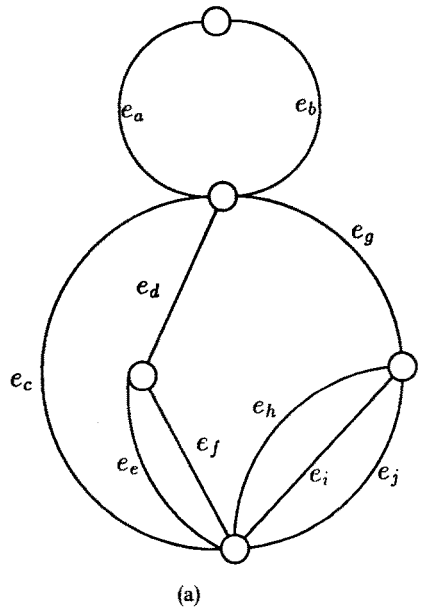
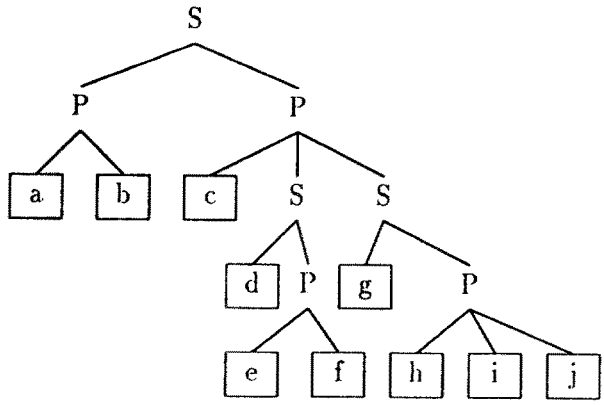


Figure 2(a) The TTSP graph of Figure 1

For convenience we describe a TTSP graph G with a tree structure, $T(G)$, called the *parsing tree* of G. For example, Fig. 2(b) is the parsing tree of the TTSP graph G shown in Fig. 2(a). Every node x in $T(G)$, along with all of its descendant nodes induces a subtree T_x which describes a TTSP subgraph G_x of G. Each leaf node labelled by x in $T(G)$ corresponds to an L-type TTSP subgraph which contains only one edge e_x in the TTSP graph G. Every internal node x is labelled by S or P according to the type of the TTSP subgraph G_x. If the node x in $T(G)$ is a child node of the root r, then G_x is called a *child TTSP subgraph* of G.

Now, the procedure for evaluating $\tau(G)$ for the TTSP graph G is described. In order to compute the value of $\tau(G)$, we introduce a spanning subgraph, called bush [10], of the TTSP graph G. Let $G = (V, E)$ be a TTSP graph with two terminals a and b. A spanning subgraph of G is called a *bush* of G if it is composed of two disjoint,



(b)

Fig. 2(b) The parsing tree of Figure 2(a)

rooted trees: one rooted at terminal a and the other rooted at terminal b. For example, Fig. 3 shows two bushes of the TTSP graph G in Fig. 2(a). Let $\tau'(G)$ denote the number of bushes in G.

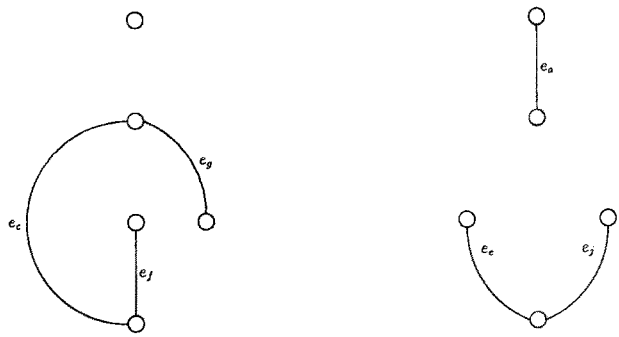


Figure 3 Two of bushes of TTSP graph G, shown in Figure 2(a)

Let G be a TTSP graph. Obviously, $\tau(G) \geq 1$ and $\tau'(G) \geq 1$. Moreover, if G is of type L, $\tau(G) = \tau'(G) = 1$. If G is of type S with child TTSP subgraphs $G_1, G_2, \ldots, G_h$, then the union of any spanning tree of G_i, for $i = 1, 2, \ldots, h$, forms a spanning tree of G. On the other hand, a bush of G can be divided into a bush of G_i for some fixed i, and spanning trees for all G_j with $j \neq i$. Thus, $\tau(G) = \prod_{i=1}^{h} \tau(G_i)$ and $\tau'(G) = \sum_{i=1}^{h} [\tau'(G_i) \prod_{j=1, j\neq i}^{h} \tau(G_j)] = \tau(G) \sum_{i=1}^{h} (\tau'(G_i)/\tau(G_i))$. If G is of type P with child TTSP

subgraphs $G_1, G_2, \ldots, G_h$, then the union of any bush of G_i, for $i = 1, 2, \ldots, h$, forms a bush of G. Moreover, any spanning tree of G can be decomposed into a spanning tree of G_i for some fixed i, and bushes for all G_j with $j \neq i$. Thus, $\tau'(G) = \prod_{i=1}^{h} \tau'(G_i)$ and $\tau(G) = \sum_{i=1}^{h} [\tau(G_i) \prod_{j=1, j\neq i}^{h} \tau'(G_j)] = \tau'(G) \sum_{i=1}^{h} (\tau(G_i)/\tau'(G_i))$.

Therefore, the values of $\tau(G)$ and $\tau'(G)$ can be determined by computing $\tau(G_x)$ and $\tau'(G_x)$ level by level for every node x in the parsing tree $T(G)$. If the input form for the TTSP graph is a link-list structure, the graph can be translated into its parsing tree in $O(q)$ time by the algorithm proposed by Valdes, Tarjan and Lawler [9]. The following theorem is obtained.

THEOREM 1. *1.* $\tau(G) = \tau'(G) = 1$ *if* G *is of type L.*

2. If G *is of type S with child TTSP subgraphs* $G_1, G_2, \ldots, G_h$, *then*

$$\tau(G) = \prod_{i=1}^{h} \tau(G_i) \quad \text{and} \quad \tau'(G) = \tau(G) \sum_{i=1}^{h} \frac{\tau'(G_i)}{\tau(G_i)}.$$

3. If G *is of type P with child TTSP subgraphs* $G_1, G_2, \ldots, G_h$, *then*

$$\tau'(G) = \prod_{i=1}^{h} \tau'(G_i) \quad \text{and} \quad \tau(G) = \tau'(G) \sum_{i=1}^{h} \frac{\tau(G_i)}{\tau'(G_i)}.$$

4. Given the TTSP graph G, *there is an* $O(q)$ *time algorithm that evaluates* $[\tau(G_x), \tau'(G_x)]^t$ *for every node* x *in parsing tree* $T(G)$.

3. The most vital edge.

To be direct, we may compute $\tau(G - e)$ for every edge e in the TTSP graph G. If the graph $G - e$ is not a TTSP graph, then $\tau(G - e) = 0$. Otherwise, $\tau(G - e)$ can be found by theorem 1. The most vital edge in G is thus easily obtained. However, this method takes $O(q^2)$ time. In this section, a linear time algorithm for finding the most vital edge in a TTSP graph will be presented.

First, given an L-type TTSP graph G which contains only the edge e, the value of $\tau(G - e)$ is 0 and the value of $\tau'(G - e)$ is 1. Then, for any TTSP graph G with an edge e_i, $\tau(G - e_i)$ and $\tau'(G - e_i)$ can be determined recursively by the following equations.

LEMMA 1. *Given the TTSP graph* G *with the child TTSP subgraphs* $G_1, G_2, \ldots, G_h$, *for each edge* $e_i \in G_i$, $i = 1, 2, \ldots, h$, we have:

1. If G *is of type S, then*

$$\begin{bmatrix} \tau(G - e_i) \\ \tau'(G - e_i) \end{bmatrix} = \frac{\tau(G)}{\tau(G_i)} \begin{bmatrix} 1 & 0 \\ \dfrac{\tau'(G)}{\tau(G)} - \dfrac{\tau'(G_i)}{\tau(G_i)} & 1 \end{bmatrix} \begin{bmatrix} \tau(G_i - e_i) \\ \tau'(G_i - e_i) \end{bmatrix} \tag{1}$$

2. If G is of type P, then

$$\begin{bmatrix} \tau(G-e_i) \\ \tau'(G-e_i) \end{bmatrix} = \frac{\tau'(G)}{\tau'(G_i)} \begin{bmatrix} 1 & \dfrac{\tau(G)}{\tau'(G)} - \dfrac{\tau(G_i)}{\tau'(G_i)} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau(G_i-e_i) \\ \tau'(G_i-e_i) \end{bmatrix}. \tag{2}$$

PROOF. 1. Let G be of type S with the TTSP subgraphs $G_1, G_2, \ldots, G_h$ and let edge e_i be an edge in child subgraph G_i. The removal of edge e_i from G influences neither the number of spanning trees nor the number of bushes in any other child subgraph $G_j, j = 1, 2, \ldots, h$, and $j \neq i$.

(a) If the subgraph G_i contains only the edge e_i, then $\tau(G_i) = \tau(G_i - e_i) = 0$, $\tau'(G_i - e_i) = 1$, and $\tau(G - e_i) = 0$. The right hand side of equation (1) is equal to $[0, \tau(G)]^t$. Note that

$$\tau'(G-e_i) = \tau'(G_i-e_i) \prod_{j=1, j\neq i}^{h} \tau(G_j) + \sum_{s=1, s\neq i}^{h} \left[\tau'(G_s)\tau(G_i-e_i) \prod_{j=1, j\neq s,i}^{h} \tau(G_j) \right]$$

$$= \prod_{j=1, j\neq i}^{h} \tau(G_j) = \tau(G).$$

Thus, the left hand side of (1) is also equal to $[0, \tau(G)]^t$.

(b) If subgraph G_i contains more than one edge, then $\tau(G_i - e_i) \neq 0$. By Theorem 1,

$$\tau(G-e_i) = \tau(G_i-e_i) \prod_{j=1, j\neq i}^{h} \tau(G_j) = \tau(G_i-e_i)\frac{\tau(G)}{\tau(G_i)}$$

and

$$\tau'(G-e_i) = \tau(G-e_i)\left\{ \left[\sum_{j=1, j\neq i}^{h} \frac{\tau'(G_j)}{\tau(G_j)} \right] + \frac{\tau'(G_i-e_i)}{\tau(G_i-e_i)} \right\}$$

$$= \tau(G_i-e_i)\cdot\frac{\tau(G)}{\tau(G_i)} \left\{ \left[\sum_{j=1}^{h} \frac{\tau'(G_j)}{\tau(G_j)} - \frac{\tau'(G_i)}{\tau(G_i)} \right] + \frac{\tau'(G_i-e_i)}{\tau(G_i-e_i)} \right\}$$

$$= \frac{\tau(G)}{\tau(G_i)} \left\{ \tau(G_i-e_i) \left[\frac{\tau'(G)}{\tau(G)} - \frac{\tau'(G_i)}{\tau(G_i)} \right] + \tau'(G_i-e_i) \right\}.$$

Using matrix form, equation (1) holds.

2. Similary, we can obtain equation (2). ■

With this observation, we define the *τ-evaluation matrix* M_x for each node x, whose parent is node y, in the parsing tree $T(G)$ as:

$$M_x = \frac{\tau(G_y)}{\tau(G_x)} \begin{bmatrix} 1 & 0 \\ \dfrac{\tau'(G_y)}{\tau(G_y)} - \dfrac{\tau'(G_x)}{\tau(G_x)} & 1 \end{bmatrix} \text{ if the parent node } y \text{ is of type S,}$$

$$M_x = \frac{\tau'(G_y)}{\tau'(G_x)} \begin{bmatrix} 1 & \dfrac{\tau(G_y)}{\tau'(G_y)} - \dfrac{\tau(G_x)}{\tau'(G_x)} \\ 0 & 1 \end{bmatrix}$$ if the parent node y is of type P and

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$ if node x is the root of $T(G)$.

Given a leaf node x in $T(G)$, it is clear that $[\tau(G_x - e_x), \tau'(G_x - e_x)]^t = [0, 1]^t$. Let $x = x_0, x_1, x_2, \ldots, x_n = r$ be the unique path from x to the root r. Then,

$$\begin{bmatrix} \tau(G - e_x) \\ \tau'(G - e_x) \end{bmatrix} = M_{x_n} M_{x_{n-1}} \cdots M_{x_0} \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

With this equation, the 1-MVE problem of the TTSP graph can be solved by $\tau(G - e^*) = \min\{\tau(G - e_x) | e_x \text{ is a leaf node in } T(G)\}$. Obviously, this method takes $O(q^2)$ time, so we will introduce another data structure called a *preserving matrix* to improve the time complexity of the algorithm. The preserving matrix P_x of node x in the parsing tree $T(G)$ can be determined recursively as follows.

$P_x = M_x$, if node x is the root node in $T(G)$,

$P_x = P_y \cdot M_x$ if node x is a non-root node with its parent node y,

where M_x is the τ-evaluation matrix of node x and P_y is the preserving matrix of node y. It is clear that if x is a leaf node in $T(G)$ and $x = x_0, x_1, x_2, \ldots x_n = r$ be the unique path from x to the root r, then

$$\begin{bmatrix} \tau(G - e_x) \\ \tau'(G - e_x) \end{bmatrix} = M_{x_n} M_{x_{n-1}} \cdots M_{x_0} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = P_{x_0} \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

With this preserving matrix, the following algorithm can be used to solve the 1-MVE problem of the TTSP graph in linear time.

Algorithm 1

Step 1 Evaluate $[\tau(G_x), \tau'(G_x)]^t$ for each node x in the parsing tree $T(G)$ by postorder traversal [1].

Step 2 For every non-root node x with its parent y, compute the τ-evaluation matrix M_x and thus, get the preserving matrix $P_x = P_y \cdot M_x$ by preorder traversal [1] in the parsing tree $T(G)$.

Step 3 Compute $\tau(G - e_x)$ for every leaf node x using

$$\begin{bmatrix} \tau(G - e_x) \\ \tau'(G - e_x) \end{bmatrix} = P_x \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Step 4 Choose the most vital edge from the results obtained in step 3. That is, choose e^* such that $\tau(G - e^*) = \min\{\tau(G - e) | e \text{ is an edge in TTSP graph } G\}$.

Obviously, algorithm 1 takes $O(q)$ time.

THEOREM 2. *The 1-MVE problem for the TTSP graph can be solved in $O(q)$ time.*

4. The k most vital edge set.

In this section, the k-MVE problem is discussed. First, we will use $k = 2$ and 4 as examples, and then we will extend them to the general case. Note that the 2-MVE set does not necessarily contain the 1-MVE set. For example, the 1-MVE set for the TTSP graph shown in Fig. 2(a) is $\{e_d\}$ while the 2-MVE set is $\{e_a, e_b\}$.

If $E^* = \{e_1, e_2\}$ is a 2-MVE set for TTSP graph G, edge e_1 is obviously the most vital edge in the subgraph $G - e_2$. Hence, a direct way of finding a 2-MVE set of G is to find a 1-MVE set in the subgraph $G - e$ for each e in the TTSP graph G, respectively. Since there are $O(q)$ edges and each takes $O(q)$ time by using Algorithm 1, it will take $O(q^2)$ time to find a 2-MVE set in a TTSP graph. However, the time complexity can be reduced to $O(p^2 + q)$. It is known that every TTSP graph is connected. Thus, $p \leqq q + 1$. Since there may exist a lot of parallel edges connecting any two vertices, p may be much smaller than q.

Let $G = (V, E)$ be a TTSP graph. The *reduced weighted TTSP graph* $R_G = (V_R, E_R)$ of G is a simple graph, where $V = V_R$ and vertices x, y in R_G are joined by an edge if and only if they are joined by some edges in the TTSP graph G. Moreover, the edge multiplicity for edge $e = (u, v)$ in G, denoted as $m(e)$, is assigned to the corresponding edge $e = (u, v)$ in R_G. Similarly, we can associate the multiplicity of edges in G with the corresponding leaf node of the parsing tree $T(R_G)$ of R_G. For example, Fig. 4(a) is the reduced weighted TTSP graph R_G of TTSP graph G shown in Fig. 2(a). Fig. 4(b) is the parsing tree $T(R_G)$ of R_G. Let $G - ke$, where $k \leq m(e)$, denote the graph $\hat{G} = (V, E - \hat{E})$ in which the set $\hat{E}$ contains k copies of the edge e.

Observe that every subset E' of E with $|E'| = 2$ is composed of two edges, which are either parallel edges in G (i.e., two such edges represented by one edge in graph R_G), or non-parallel edges in G (i.e., two such edges represented by two edges in graph R_G). We can solve a 2-MVE problem in a reduced weighted TTSP by the following algorithm.

Algorithm 2

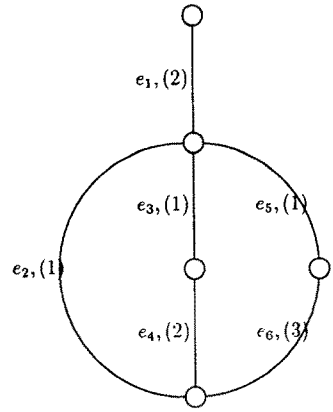
Step 1 Transfer the parsing tree $T(G)$ of the TTSP graph G to the parsing tree $T(R_G)$ of the reduced weighted TTSP graph R_G.

Step 2 Evaluate $[\tau(R_{G_x}), \tau'(R_{G_x})]^t$ for each node x in the parsing tree $T(R_G)$ by postorder traversal.

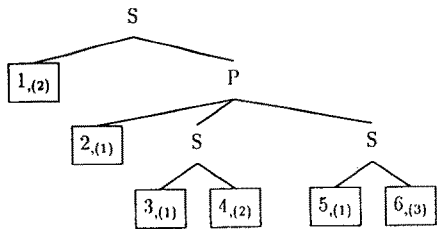
Step 3 Compute $\tau(G - 2e)$ for every edge e in R_G with an edge multiplicity of at least two.

Step 4 For each graph R_{G-e_i}, $e_i \in E_R$, compute $\tau((G - e_i) - e_j)$ where $e_j \in E_R$ and $e_j \neq e_i$.

Step 5 Choose the 2-MVE set from the results obtained in step 3 and step 4. That is, choose edge set $E^* = \{e_1^*, e_2^*\}$ such that $\tau(G - E^*) = \min\{\tau(G - E') | E' \subset E$ and $|E'| = 2\}$.



$e_x,(y)$: y indicates the corresponding edge multiplicity of edge e_x
(a)



$x_{(y)}$: y indicates the corresponding edge multiplicity of leaf node x
(b)

Figure 4(a) The reduced weighted TTSP graph of Figure 2(a)
(b) The parsing tree of Figure 4(a)

The evaluation of $[\tau(R_{G_x}), \tau(R_{G_x})]^t$ in the parsing tree $T(R_G)$ is the same as the evaluation of $[\tau(G_x), \tau'(G_x)]^t$ in Theorem 1. The only difference is $[\tau(R_{G_x}), \tau'(R_{G_x})]^t = [m(e_x), 1]^t$ if the graph R_{G_x} is of type L and $m(e_x)$ is its multiplicity. Similarly, the evaluation of $[\tau(G - e_x), \tau'(G - e_x)]^t$ in the parsing tree $T(R_G)$ is $[\tau(G - e_x), \tau'(G - e_x)]^t = M_{x_n} M_{x_{n-1}} \dots M_{x_0}[m(e_x) - 1, 1]^t$, if $x_0, x_1, x_2, \dots, x_n = r$ is the unique path from leaf node x to the root r in the parsing tree $T(R_G)$. Moreover, $[\tau(G - ke_x), \tau'(G - ke_x)]^t = M_{x_n} M_{x_{n-1}} \dots M_{x_0}[m(e_x) - k, 1]^t$, where $m(e_x) \geq k$. In Algorithm 2, step 1 takes $O(q)$ time. Since $R_G = (V, E_R)$ is a planar graph, $|E_R| = O(p)$. Thus, steps 2 and 3 take $O(p)$ time and step 4 takes $O(p^2)$ time. The algorithm takes $O(p^2 + q)$ time total.

Now, we will discuss the case $k = 4$. It is observed that every edge subset E' with $|E'| = 4$ is composed of four edges in the TTSP graph G, which may be four parallel edges, three parallel edges together with another edge, two parallel edges together with another two parallel edges, two parallel edges together with two non-parallel

edges or four non-parallel edges. In short, we can write $4 = 4 = 3 + 1 = 2 + 2 = 2 + 1 + 1 = 1 + 1 + 1 + 1$. Using the same idea as in the case $k = 2$, the time complexity for solving a 4-MVE problem is $O(p^4 + p^3 + 2p^2 + p + q) = O(p^4 + q)$.

As in the discussion above, a k-MVE problem can also be solved in a similar way. The time complexity is $O(d_k p^k + d_{k-1} p^{k-1} + \ldots + d_2 p^2 + d_1 p + q)$, where d_i is the number of integer partitions of integer k being divided into exactly i parts. Let π_i denote the number of ordered integer partitions of integer k being divided into exactly i parts. For example, $d_3 = 1$ and $\pi_3 = 3$, when $k = 4$ and $i = 3$. Note that $4 = 2 + 1 + 1$ when the order of 1 and 2 is ignored and $4 = 2 + 1 + 1 = 1 + 2 + 1 = 1 + 1 + 2$ when the order is taken into account. Obviously, $d_i \leq \pi_i$. Since $\pi_i = C(k-1, i-1) \leq C(k, i)$, where $C(m, n)$ is the binomial coefficient, $d_k p^k + d_{k-1} p^{k-1} + \ldots + d_2 p^2 + d_1 p + q \leq (p+1)^k + q$. Therefore, the k-MVE problem can be solved in $O(p^k + q)$ time.

5. Conclusions.

In this paper, an $O(q)$ time algorithm for solving a 1-MVE problem in a TTSP graph is presented, where q is the number of edges in the graph. A reduction method is also proposed to solve a k-MVE problem in $O(p^k + q)$ time, where p is the number of vertices in the graph.

REFERENCES

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, 54–55, Addison-Wesley, Reading, MA, 1974.
2. M. O. Ball, B. L. Golden and R. V. Vohra, *Finding the most vital arcs in a network*, Operations Research Letters, 8, 73–76 (1989).
3. C. J. Colbourn, *The Combinatorics of Network Reliability*, 49–53, Oxford University Press, Oxford, England, 1987.
4. H. W. Corley and D. Y. Sha, *Most vital links and nodes in weighted networks*, Operations Research Letters, 1, 157–160 (1982).
5. R. J. Duffin, *Topology of series-parallel networks*, Journal of Mathematical Analysis and Applications, 10, 303–318 (1965).
6. R. Gould, *Graph Theory*, Benjamin/Cummings, Menlo Park, CA, 1988.
7. S. H. Lubore, H. D. Ratliff and G. T. Sicilia, *Determining the most vital links in a flow network*, Naval Research Logistics Quarterly, 17, 497–502 (1970).
8. H. D. Ratliff and A. S. Rosenthal, *Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem*, Operations Research, 507–521 (1983).
9. J. Valdes, R. Tarjan, and E. L. Lawler, *The recognition of series parallel digraphs*, SIAM Journal on Computing, 11, 298–313 (1982).
10. J. A. Wald and C. J. Colbourn, *Steiner trees, partial 2-trees, and minimum IFI networks*, Networks, 13, 159–167 (1983).
11. Y. L. Wang, R. S. Chang and P. R. Chang, *The tour problems in two-terminal series-parallel graphs*, to appear in BIT.