ELSEVIER

Information Processing Letters 53 (1995) 55-59

Information Processing Letters

# Element perturbation problems of optimum spanning trees with two-parameter objectives [1]

Yung-Cheng Chang [a], Lih-Hsing Hsu [b,*]

[a] *Institute of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, 30050, Taiwan, ROC*
[b] *Department of Computer and Information Science, National Chiao Tung University, Hsinchu, 30050, Taiwan, ROC*

Communicated by K. Ikeda; received 24 November 1993

**Abstract**

Let $G = (V, E)$ be a graph. We associate with each edge $e_i \in E$ an ordered pair of rational numbers $(a_i, b_i)$. Let the weight of a spanning tree $T$, $w(T)$, be defined as $\sum_{e_i \in T} a_i + \prod_{e_i \in T} b_i$. A spanning tree $T$ in $G$ is called a $w$-optimum spanning tree if $w(T) \geqslant w(T')$ for all spanning trees $T'$ in $G$. The function $w$ is one instance in a class of two-parameter objectives. Hassin and Tamir proposed a unified approach for solving the class of two-parameter objective optimum spanning tree problems. Let $s$ be an objective in the class and $F_s(G)$ denote the weight of the $s$-optimum spanning tree of $G$. The element perturbation problem of the $s$-optimum spanning tree is to compute $F_s(G - e_k)$ for all $e_k \in E$. With Hassin and Tamir's approach, let $t_s(p, q)$ be the complexity of computing the $s$-optimum spanning tree where $p = |V|$ and $q = |E|$. In this paper, we present an approach to solve the element perturbation problem of the $s$-optimum spanning tree in $t_s(p, q)$.

*Keywords:* Combinatorial algorithms; Complexity; Spanning trees; Matroid

## 1. Introduction and notation

Most of the graph definitions used in this paper are standard (see, e.g., [1]). Here, we limit ourselves to defining the most commonly used terms and those that may produce confusion. $G = (V, E)$ is called a *graph* if $V$ is a finite set and $E$ is a subset of $\{(u, v) \mid (u, v)$ is an unordered pair of $V$, where $u \neq v\}$. We say $V$ is the *vertex set* of $G$ and $E$ is the *edge set* of $G$. Let $p = |V|$ and $q = |E|$. Let $\bar{E}$ be a subset of $E$. We use $G - \bar{E}$ to denote the graph $G' = (V, E - \bar{E})$. In particular, we use $G - e_k$ to denote the graph $G - \{e_k\}$, where $e_k \in E$. Graph $H = (V', E')$ is called a *subgraph* of $G$ if $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$. A subgraph $H = (V', E')$ of $G$ with $V' = V$ is called a *spanning subgraph* of $G$. A *spanning tree* of $G$ is a connected spanning subgraph of $G$ that contains exactly $p - 1$ edges. We also use $T$ to denote the spanning tree with edge set $T$. $D = (U, A)$ is called a *directed graph* if the vertex set $U$ is a finite set and the *arc set* $A$ is a subset of $\{[u, v] \mid [u, v]$ is an ordered pair of $U$, where $u \neq v\}$. We use $[u, v]$ to denote the arc incident from $u$ to $v$.

Let $G = (V, E)$ be a graph. We associate with each edge $e_i \in E$ an ordered pair of rational numbers $(a_i, b_i)$. For a subset $E'$ of $E$, define $A(E') = \sum_{e_i \in E'} a_i$ and $B(E') = \sum_{e_i \in E'} b_i$. Let $g$ be a real valued function defined in $R^2$. The problem of max-

* Corresponding author. Email: lhhsu@cc.nctu.edu.tw.

[1] This work was supported in part by the National Science Council of the Republic of China under contract NSC83-0208-M009-034.


*SSDI* 0020-0190(94)00173-1

imizing $g(A(T), B(T))$ over all spanning trees $T$ of $G$ has been discussed by Hassin and Tamir [3]. Finding a spanning tree $T$ which maximizes $(\sum_{e_i \in T} a_i)^2 + (\sum_{e_i \in T} b_i)^2$, $(\sum_{e_i \in T} a_i + \prod_{e_i \in T} b_i$ or $\prod_{e_i \in T} a_i + \prod_{e_i \in T} b_i$, respectively) can be transformed into the problem of maximizing $A(T)^2 + B(T)^2$ $(A(T) + \exp B'(T)$ with $B'(T) = \sum_{e_i \in T} \log b_i$ or $\exp A'(T) + \exp B'(T)$ with $A'(T) = \sum_{e_i \in T} \log a_i$ and $B'(T) = \sum_{e_i \in T} \log b_i$, respectively). Hassin and Tamir proposed a unified strategy which yields strongly polynomial algorithms for a class of two-parameter maximization problems that includes the above three objective functions. Note that the problem of minimizing the cost/reliability ratio over all spanning trees of a graph [2] is modeled as $\prod_{e_i \in T} b_i / \sum_{e_i \in T} a_i$ and also solved in [3].

Let $s(T) = g(A(T), B(T))$ be one of the above two-parameter objectives. The *weight* of a spanning tree $T$ is the objective value of $s(T)$. A spanning tree $T$ in $G$ is called an *s-optimum spanning tree* if $s(T) \geqslant s(T')$ for all spanning trees $T'$ of $G$. Finding the $s$-optimum spanning tree with respect to $s$ is called the $s$-MST problem. We use $t_s(p, q)$ to denote the time complexity of solving the $s$-MST problem with Hassin and Tamir's approach. Let $F_s(G)$ denote the weight of the $s$-optimum spanning tree of $G$ if $G$ is connected; otherwise, $F_s(G) = -\infty$. The *element perturbation problem* (EPP) of the $s$-optimum spanning tree, $s$-EPP for short, is to compute $F_s(G - e_k)$ for all $e_k \in E$. A naive method of solving the $s$-EPP is to repeatedly apply Hassin and Tamir's algorithm to compute $F_s(G - e_k)$ for every $e_k$. This approach takes $O(qt_s(p, q))$ time. However, $F_s(G - e_k) = f(G)$ if the edge $e_k$ is not in the $s$-optimum spanning tree. Thus, we can reduce the time to $O(pt_s(p, q))$. In this paper, we present a $t_s(p, q)$ algorithm for solving the $s$-EPP.

The study of the $s$-EPP is interesting for several reasons. First, the concept of the EPP can be generalized to combinatorial optimization problems. Recently, Hsu, Leu, and Sung discussed general EPP strategies [5]. If we solve the EPP of a combinatorial optimization problem, we can find the second optimal solution(s) and the critical element(s). A naive method of solving an EPP is to repeatedly solve the original problem. To avoid repeated execution of an algorithm, we extract and reuse information from what we have solved. Several EPPs can then be solved with the same time complexity as the original problem [4–7,9]. Algorithms that reuse information in this way are called *recycling algorithms* [5]. It has been observed that recycling strategies are problem-dependent. It is interesting to collect as many recycling strategies as possible. Hassin and Tamir's algorithm is a unified approach for a class of interesting problems. Thus, it is worthwhile to investigate the corresponding recycling strategy. Also, since the two-parameter maximum spanning tree problem is a topic in matroid theory [8], we expect that our approach may lead to some insights into matroid theory.

## 2. Previous work

To make this paper self-contained, we first review the basic strategy of Hassin and Tamir's approach as proposed in [3]. Assume that $(a_i, b_i) \neq (a_j, b_j)$ if $e_i \neq e_j$ to avoid degenerate cases. Let $T^*$ be the $s$-optimum spanning tree with respect to the objective function $s$. We call a spanning tree $T$ a *local optimal* spanning tree if there is no pair of elements $e_i, e_j \in E$ such that $e_i \in T$, $e_j \notin T$, and $T' = T - \{e_i\} + \{e_j\}$ is a spanning tree which yields a larger weight than $T$ does. Hassin and Tamir divided the $R^2$ plane into a number of cells and showed that all points in a cell produce at most one local optimal spanning tree. Obviously, the $s$-optimum spanning tree $T^*$ is one of these local optimal spanning trees. $T^*$ will be contributed by the unique cell in $R^2$ containing $(A(T^*), B(T^*))$.

Let $(A, B)$ be a point in $R^2$. Define a directed graph $D_{A,B}(G)$ with the vertex set being the edge set $E$ of $G$. In short, we use $D_{A,B}$ to denote $D_{A,B}(G)$. Let $e_i, e_j$ be distinct elements in $E$. $[e_i, e_j]$ is an arc in $D_{A,B}$ if and only if $g(A - a_i + a_j, B - b_i + b_j) > g(A, B)$. An equivalence in $R^2$ can be defined by $(A, B) \sim (C, D)$ if and only if $D_{A,B} = D_{C,D}$.

Let $e_i, e_j \in E$, $e_i \neq e_j$. Define the function $g_{ij}(A, B) = g(A - a_i + a_j, B - b_i + b_j) - g(A, B)$. The equivalence regions in $R^2$ are induced by the set of $q(q-1)$ $g_{ij}(A, B)$ functions. Define $R_{ij} = \{(A, B) \mid g(A - a_i + a_j, B - b_i + b_j) > g(A, B)\}$. Let $f : R^2 \to R^k$ be a mapping of $R^2$ into $R^k$. Further, let $T_{ij}$, $e_i, e_j \in E$, $e_i \neq e_j$ be a collection of subsets in $R^k$ such that $(A, B) \in R_{ij}$ if and only if $f(A, B) \in T_{ij}$ for $e_i, e_j \in E$, $i \neq j$. Suppose there exists a polynomial $h_{ij}(x_1, \ldots, x_k)$ such that

$T_{ij} = \{(x_1, \ldots, x_k) \mid h_{ij}(x_1, \ldots, x_k) > 0\}$. Then the number of equivalence regions is bounded by the number of topological components induced on $R^k$ by the set of polynomials $h_{ij}$. If $d$, the maximum degree of $h_{ij}$, and $k$, the dimension of $R^k$, are constant and independent of $q$, then it can be proved that the number of equivalence regions will be a polynomial in $q$.

For ease of exposition, we use the weight function $w(T) = \sum_{e_i \in T} a_i + \prod_{e_i \in T} b_i$ as the objective. In the $w$-MST problem, $g(A(T), B'(T)) = A(T) + \exp B'(T)$ with $A(T) = \sum_{e_i \in T} a_i$, $B'(T) = \sum_{e_i \in T} \log b_i$. Given a point $(A, B) = (A(T), B'(T))$, $R_{ij} = \{(A, B) \mid (A - a_i + a_j) + \exp(B - \log b_i + \log b_j) > A + \exp B\} = \{(A, B) \mid -a_i + a_j > (1 - b_j/b_i) \exp B\}$. Set $f(A, B) = \exp B$ and $T_{ij} = \{x \mid -a_i + a_j > x(1 - b_j/b_i)\}$. Hence, $(A, B) \in R_{ij}$ if and only if $f(A, B) \in T_{ij}$ for every pair of distinct edges $e_i, e_j \in E$. The number of equivalence regions is $O(q^2)$ for the $w$-MST problem. These regions correspond to the partition of the line induced by the points $d_{ij} = b_i(a_j - a_i)/(b_i - b_j)$, where $e_i, e_j \in E$ and $e_i \neq e_j$. Each equivalence region defines a $D_{A,B}$, where $(A, B)$ is a point in that region.

Let $E(D_{A,B})$ be the arc set of $D_{A,B}$. Let $T_1$ and $T_2$ be two distinct spanning trees of $G$. We say that $T_2$ is a $D_{A,B}$-*improvement* of $T_1$ if there exist $e_i \in T_1$ and $e_j \notin T_1$ such that $[e_i, e_j] \in E(D_{A,B})$ and $T_2 = T_1 - \{e_i\} + \{e_j\}$, i.e., $T_2$ is obtained from $T_1$ by a single edge swap. A spanning tree $T$ of $G$ is $D_{A,B}$-*optimal* if there exists no spanning tree $T'$ of $G$ which is a $D_{A,B}$-improvement of $T$. In [3], the following theorem is presented.

**Theorem 1.** *There is at most one $D_{A,B}$-optimal spanning tree of $G$ for every $(A, B)$ in $R^2$.*

We use $T_{A,B}$ to denote the $D_{A,B}$-optimal spanning tree if it exists. Let $\Gamma(D_{A,B})(e_i)$ be the set $\{e_j \mid [e_i, e_j] \in E(D_{A,B})\}$ for $e_i \in E$. Also let $X(D_{A,B})$ be the set $\{e_i \mid$ the two endpoints of $e_i$ in $G$ are on different connected components of the graph $H = (V, \Gamma_{A,B}(e_i))\}$. In short, we use $\Gamma_{A,B}(e_i)$ to denote $\Gamma(D_{A,B})(e_i)$ and $X_{A,B}$ to denote $X(D_{A,B})$. The following theorem is also from [3].

**Theorem 2.** *If the $D_{A,B}$-optimal spanning tree $T_{A,B}$ exists, then the edge set of $T_{A,B}$ is exactly $X_{A,B}$.*

This theorem states that a necessary condition for the existence of the $D_{A,B}$-optimal spanning tree is that $X_{A,B}$ forms a spanning tree. If $X_{A,B}$ forms a spanning tree, it is a *candidate solution*. It is suggested in [3] that we do not have to verify that the candidate solution is $D_{A,B}$-optimal. To reduce the computational complexity, it will suffice simply to find the candidate solution. The $s$-optimum spanning tree is a candidate solution which has maximum weight with respect to $s$. The following algorithm proposed in [3], Algorithm 1, finds $X_{A,B}$ and then tests whether it forms a candidate solution in a $D_{A,B}$.

**Algorithm 1.**

**Step 1.** Compute $\Gamma_{A,B}(e_x)$ for all $e_x \in E$.

**Step 2.** Set $X_{A,B} = \phi$.

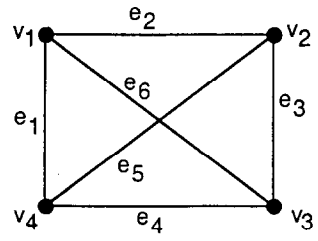
**Step 3.** For each $e_x \in E$ do the following:
If the two endpoints of $e_x$ are disconnected in $H = (V, \Gamma_{A,B}(e_x))$, set $X_{A,B} = X_{A,B} \cup \{e_x\}$.

**Step 4.** If $X_{A,B}$ does not form a spanning tree, stop and conclude that the $D_{A,B}$ has no $D_{A,B}$-optimal solution. Otherwise, $X_{A,B}$ forms the candidate solution.

Obviously, step 1 in Algorithm 1 takes $O(q^2)$ time. Step 3 needs $q$ tests to see if the endpoints of $e_x$ are not connected in $H = (V, \Gamma_{A,B}(e_x))$. Each test takes $O(q)$ time. Hence, step 3 takes $O(q^2)$ time. Step 4 is completed in $O(p)$ time. Hence, the complexity of Algorithm 1 is $O(q^2)$. For the weight function $w(T)$, we can find the $w$-optimum spanning tree by finding all candidate solutions among all $O(q^2)$ different $D_{A,B}$s. Then we select the candidate solution with maximum weight to be the $w$-optimum spanning tree. The whole process takes $O(q^4)$ time.

## 3. The EPP strategy

Although our EPP strategy is independent of the equivalence region construction, we use the $w$-EPP as an example to demonstrate our method. We want to extract some useful information when computing $F_w(G)$ and then reuse this information to compute $F_w(G - e_k)$s. Assume we are given a graph $G = (V, E)$ and each edge $e_i \in E$ is associated with an ordered pair of rational numbers $(a_i, b_i)$. Let $T^*$ be the $w$-optimum spanning tree of $G$ and $T^{e_k}$ be the $w$-optimum spanning tree of $G - e_k$.




| | $a_i$ | $b_i$ |
|---|---|---|
| $e_1$ | 40 | 7 |
| $e_2$ | 41 | 8 |
| $e_3$ | 42 | 9 |
| $e_4$ | 203 | 4 |
| $e_5$ | 202 | 3 |
| $e_6$ | 201 | 2 |

Fig. 1. A graph $G = (V, E)$ and the $(a_i, b_i)$ for every edge.

Fig. 1 is an example of a graph $G$ with four vertices. The $(a_i, b_i)$s are listed in the table. In this example, $T^* = \{e_4, e_5, e_6\}$, $T^{e_1} = T^{e_2} = T^{e_3} = T^*$, and $T^{e_4} = T^{e_5} = T^{e_6} = \{e_1, e_2, e_3\}$.

To apply Hassin and Tamir's algorithm, we need to construct $D_{A,B}$s for all equivalence regions in $R^2$. There are $O(q^2)$ different $D_{A,B}$s for $G$. When constructing a $D_{A,B}$, we create an arc $[e_i, e_j]$ if $g(A - a_i + a_j, B - \log b_i + \log b_j) > g(A, B)$, where $g(A(T), B'(T)) = A(T) + \exp B'(T)$. Let $D^{e_k}_{A,B}$ be $D_{A,B}(G - e_k)$. From the construction of a $D_{A,B}$, we know that if $[e_i, e_j]$ is an arc in $D^{e_k}_{A,B}$, $[e_i, e_j]$ must be an arc in $D_{A,B}$. We have the following lemma.

**Lemma 3.** *$D^{e_k}_{A,B}$ is the subgraph of $D_{A,B}$ induced by the vertex set $E - \{e_k\}$. Moreover, $\Gamma(D^{e_k}_{A,B})(e_i)$ is exactly $\Gamma_{A,B}(e_i) - \{e_k\}$, where $e_i \neq e_k$.*

With Lemma 3, we can easily obtain $D^{e_k}_{A,B}$ from $D_{A,B}$ for every $(A, B)$ in $R^2$ and $e_k \in E$. Fig. 2(a) is $D_{300,\log 40}$, where $G$ is the graph in Fig. 1. Fig. 2(b) is $D^{e_5}_{300,\log 40}$. Since the $O(q^2)$ $D_{A,B}$s cover all $(A, B)$s in $R^2$, the $D^{e_k}_{A,B}$s induced by $E - \{e_k\}$ from the $D_{A,B}(G)$s will also cover all $(A, B)$s.

Let $X^{e_k}_{A,B}$ be $X(D_{A,B}(G - e_k))$. We want to obtain $X^{e_k}_{A,B}$ from $X_{A,B}$. Note that the two endpoints of an $e_i$ in $X_{A,B}$ are on different connected components of the graph $H = (V, \Gamma_{A,B}(e_i))$. Let $e_i$ be an edge in $X^{e_k}_{A,B}$ but not in $X_{A,B}$. The two endpoints of $e_i$ are connected in $H = (V, \Gamma_{A,B}(e_i))$ but are not connected in $H' = (V, \Gamma_{A,B}(e_i) - \{e_k\})$. In other words, $e_k$ is a *cut edge* of the connected component in $H$ connecting the two endpoints of $e_i$. We use $R^{e_k}_{A,B}$ to denote the set $\{e_i \mid e_k$ is a cut edge of the connected component in $H$ connecting the endpoints of $e_i\}$. We have the following theorem.

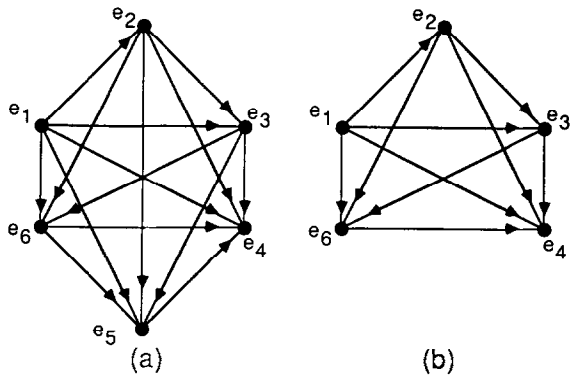



Fig. 2. (a) The $D_{300,\log 40}$ for $G$ in Fig. 1. (b) The $D^{e_5}_{300,\log 40}$ for $G - e_5$.

**Theorem 4.** $X^{e_k}_{A,B} = X_{A,B} - \{e_k\} \cup R^{e_k}_{A,B}$.

**Proof.** Obviously, $e_k \notin X^{e_k}_{A,B}$ since $e_k \notin D^{e_k}_{A,B}$. From the above discussion, if $e_i$ is an edge of $X^{e_k}_{A,B} - X_{A,B}$, $e_i \in R^{e_k}_{A,B}$. Suppose $e_i$ is in $X_{A,B}$ and $e_i \neq e_k$. Then the endpoints of $e_i$ are not connected in $H = (V, \Gamma_{A,B}(e_i))$. The endpoints of $e_i$ are not connected in $H' = (V, \Gamma_{A,B}(e_i) - \{e_k\})$, which is a subgraph of $H$. Hence, $e_i$ is in $X^{e_k}_{A,B}$. Therefore, $X^{e_k}_{A,B} = X_{A,B} - \{e_k\} \cup R^{e_k}_{A,B}$. □

In Fig. 2, $X_{300,\log 40}$ is $\{e_4, e_5, e_6\}$, which forms the $T^*$. $e_3$ is not in $X_{300,\log 40}$ because $\Gamma_{300,\log 40}(e_3) = \{e_4, e_5, e_6\}$ forms a connected component connecting the two endpoints of $e_3$ in $G$. From Theorem 4, $X^{e_5}_{300,\log 40} = \{e_4, e_5, e_6\} - \{e_5\} \cup \{e_3\}$, where $e_3$ is the only element in $R^{e_5}_{300,\log 40}$. Hence, $D^{e_5}_{300,\log 40}$ contributes the candidate solution $X^{e_5}_{300,\log 40} = \{e_3, e_4, e_6\}$ to compete for the $w$-optimum spanning tree of $G - e_5$.

Let $e_x \in E$ and $e_x = (u_x, v_x)$, where $u_x, v_x \in V$. The following algorithm, Algorithm 2, finds $R^{e_k}_{A,B}$ for every $e_k$ in $E$ with respect to $D_{A,B}$.

**Algorithm 2.**

**Step 1.** For each $e_y \in E$, set $R^{e_y}_{A,B} = \phi$.

**Step 2.** For each $e_x \in E$ do the following:

Construct $\Gamma_{A,B}(e_x)$. If the two endpoints of $e_x$ are connected in $H = (V, \Gamma_{A,B}(e_x))$, find every cut edge $e_y$ in the connected component that connects $u_x$ to $v_x$ and set $R^{e_y}_{A,B} = R^{e_y}_{A,B} \cup \{e_x\}$.

Step 1 in Algorithm 2 takes $O(q)$ time. The con-

struction of a $\Gamma_{A,B}(e_x)$ requires $O(q)$ time. The cut edges can be found with *depth-first search* techniques starting from $u_x$ or $v_x$ in $O(q)$ time [10]. Hence, step 2 can be performed in $O(q^2)$ for all $e_x \in E$. The loop in step 2 assigns the $e_x$ to all the $R^{e_k}_{A,B}$'s, where $e_k$ is a cut edge of the connected component in $H$ that connects $u_x$ to $v_x$. Hence, step 2 constructs $R^{e_k}_{A,B}$ for all $e_k \in E$.

Note that the $w$-EPP algorithm reuses the $X_{A,B}$ obtained from Algorithm 1. Even if the $X_{A,B}$ fails to form a candidate solution for $D_{A,B}$, the $X^{e_k}_{A,B}$ may become a candidate solution for $D^{e_k}_{A,B}$. After the execution of Algorithms 1 and 2 for $D_{A,B}$, the $X^{e_k}_{A,B}$s can also be found in $O(q^2)$ time by setting $X^{e_k}_{A,B} = X_{A,B} - \{e_k\} \cup R^{e_k}_{A,B}$ for all $e_k \in E$. We conclude that the $w$-EPP can be solved in $O(q^4)$ time.

Hassin and Tamir's unified approach for finding local solutions applies to a class of maximization problems that construct equivalence regions in various ways. Our Algorithm 2 for a $D_{A,B}$ is independent of the method of constructing the equivalence regions. Hence, our strategy can be applied to this entire class of problems. If we define the weight function $s(T)$ as $(\sum_{e_i \in T} a_i)^2 + (\sum_{e_i \in T} b_i)^2$, $\prod_{e_i \in T} a_i + \prod_{e_i \in T} b_i$, or $\prod_{e_i \in T} b_i / \sum_{e_i \in T} a_i$ and use Hassin and Tamir's approach, the time compexity is the same for solving the $s$-MST problem and the $s$-EPP.

## References


[1] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications* (Elsevier, New York, 1976).

[2] R. Chandrasekaran and A. Tamir, Polynomial testing of the Query "Is $a^b > c^d$?" with application on finding a minimal cost reliability ratio spanning tree, *Discrete Appl. Math.* **9** (1984) 117–123.

[3] R. Hassin and A. Tamir, Maximizing classes of two-parameter objectives over matroids, *Math. Oper. Res.* **14** (1989) 362–375.

[4] L.H. Hsu, R.H. Jan, Y.C. Lee, C.N. Hung and M.S. Chern, Finding the most vital edge with respect to minimum spanning tree in weighted graphs, *Inform. Process. Lett.* **39** (1991) 277–281.

[5] L.H. Hsu, S.C. Leu and T.Y. Sung, Second optimal solutions, critical elements and recycling algorithms, submitted for publication.

[6] C.N. Hung, L.H. Hsu and T.Y. Sung, The most vital edges for matchings in a bipartite graph, *Networks* **23** (1993) 309–313.

[7] N. Katoh, T. Ibaraki and H. Mine, An efficient algotithm for $K$ shortest simple paths, *Networks* **12** (1982) 411–427.

[8] J. Lee and J. Ryan, Matroid applications and algorithms, *ORSA J. Comput.* **4** (1992) 70–98.

[9] K. Malik, A.K. Mittal and S.K. Gupta, The $k$ most vital arcs in the shortest path problem, *Oper. Res. Lett.* **8** (1989) 223–227.

[10] R.E. Tarjan, Depth-first search and linear graph algorithms, *SIAM J. Comput.* **1** (1972) 146–160.