ELSEVIER

Discrete Applied Mathematics 113 (2001) 319–327

DISCRETE APPLIED MATHEMATICS

Note

Finding the k most vital edges with respect to minimum spanning trees for fixed k

Weifa Liang *

Department of Computer Science, The Australian National University, Canberra ACT 0200, Australia

Received 10 June 1999; received in revised form 28 September 2000; accepted 10 January 2001

Abstract

Assume that $G(V,E)$ is a weighted, undirected, connected graph with n vertices. The k most vital edge problem with respect to a minimum spanning tree is to find a set S^* of k edges from E such that the removal of the edges in S^* results in the greatest increase in the weight of the minimum spanning tree in the resulting graph $G(V,E-S^*)$. In this paper, an improved algorithm for the problem with fixed k, $k \geqslant 2$, has been presented. The proposed algorithm runs in time $O(n^k\alpha((k+1)(n-1),n))$, which improves a previously known result by an $O(n/\alpha((k+1)(n-1),n))$ factor, where α is a functional inverse of Ackermann's function which grows very slow. The parallel version of the algorithm takes $O(\log n \log\log n)$ time using $O(n^k/\log n)$ processors on a CREW PRAM.

Keywords: Combinatorial algorithms; Minimum spanning trees; Most vital edges; Network optimization; Parallel algorithms

1. Introduction

Let $G(V,E)$ be an undirected, weighted, connected graph with a vertex set V and an edge set E, where $m=|E|$, $n=|V|$. Associated with each edge $e \in E$, there is a real valued weight $w(e)$. The *minimum spanning tree* (MST) problem is to find a spanning tree of G such that the sum of the edge weights of the spanning tree is minimized. This problem has been well studied in the past two decades [2,11]. The best sequential algorithm for it takes $O(m \log\beta(m,n))$ time [11] where $\beta(m,n) = \min\{i : \log^{(i)} n \leqslant m/n\}$ ($\log^{(i)}$ is the iterated logarithm so that $\log^{(i+1)} = \log\log^{(i)}$ for $i > 0$). The best parallel algorithms for it require $O(\log n)$ time using $O(m+n)$ processors on an EREW PRAM [4] for sparse graphs, and $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on a CREW PRAM [3] for dense graphs. On the other hand, there are also several fast randomized sequential and parallel algorithms for the problem [5,15,6]. However, the

* Tel.: +61-2-6249-3019; fax: +61-2-6249-0010.
E-mail address: wliang@cs.anu.edu.au (W. Liang).

0166-218X/01/$ - see front matter
PII: S0166-218X(01)00189-5

deterministic simulation of these randomized algorithms may take more time than that of the best known deterministic sequential and parallel algorithms.

One closely related problem is the k *most vital edge problem* with respect to an MST of G (the k most vital edge problem for short), which is to find an edge subset $S^* \subseteq E$ with $|S^*| = k$ to maximize the weight $w(MST(G(V, E - S^*)))$ of the MST in the resulting graph $G(V, E - S^*)$. The k most vital edge problem has many practical applications including the design of robust telecommunications networks and distributed computing [10–14]. Obviously, $k \leqslant \lambda \leqslant \lceil m/n \rceil$, where λ is the edge connectivity of G. Otherwise, the resulting graph is disconnected and no MST exists after deleting all the edges in a minimum cut of G with any other $k - \lambda$ edges. Therefore, in this paper we assume that G is $(k+1)$-edge-connected at least when dealing with the k most vital edge problem. Without loss of generality, we also assume that the weight associated with each edge in G is distinct and hence the MST or the minimum spanning forest (MSF for short) of graph $G(V, E - S)$ is unique for every $S \subseteq E$.

The *single most vital edge problem*, a special case with $k = 1$, has been extensively studied in the literature [12–14,16,19]. In the sequential environment, Hsu et al. [12] and Iwano and Katoh [14] proposed algorithms for it. In the parallel computing environment, Hsu et al. [13], Shen [19], and Liang and Shen [16] presented parallel algorithms for it. However, for a general k, Frederickson and Solis-Oba [10] have shown that the k most vital edge problem is NP-complete. Instead, they presented an approximate algorithm for it. Their algorithm requires $O(\min\{km \log n + k^2\, n \log n, km \log^2 n\})$ time, and the solution delivered is $\Omega(1/\log k)$ times of the optimal. Shen [20] explored this problem by giving an exact algorithm and a randomized, approximate algorithm. His exact algorithm needs $O(n^k m \log \beta(m, n))$ time when k is fixed. Liang and Shen [16] improved Shen's result by presenting an $O(n^{k+1})$ time algorithm for fixed k.

In this paper, we improve the result in [16] further by giving an $O(n^k \alpha((k+1)(n-1), n))$ algorithm, which improves in time by a factor of $O(n/\alpha((k+1)(n-1), n))$, where $\alpha((k+1)(n-1), n)$ is almost constant in practice [7, p. 453] and grows very slow. We also show that the proposed algorithm can be parallelized, and the parallel version requires $O(\log n \log\log n)$ time using $O(n^k/\log n)$ processors on a CREW PRAM.

The remainder of the paper is organized as follows. Section 2 introduces the notations and notions related to our problem. Section 3 presents an improved algorithm for the k most vital edge problem with fixed k. Section 4 concludes our discussions.

2. Preliminaries

Consider an unweighted, undirected graph G. Let T_1' be a maximal spanning forest of G and T_i' be a maximal spanning forest of graph $G_i = G - \cup_{j=1}^{i-1} T_j'$ for $i > 1$. Denote by $U_i' = \cup_{j=1}^{i} T_j'$ the union of the maximal spanning forests $T_1', T_2', \ldots, T_i'$. Then, the graph U_k' is called a *sparse k-edge-connectivity certificate* of G, and U_k' is l-edge-connected if and only if G is l-edge-connected, for any integer l with $0 \leqslant l \leqslant k$ [17,18]. This property always holds no matter whether G is a simple graph or not. In [16] a weighted

version of the above certificate is introduced, which is defined as follows. Let G be a weighted, undirected graph, T_1 be the MST/MSF of G, and T_i be the MST/MSF of $G_i = G - \cup_{j=1}^{i-1} T_j$ for $i > 1$. The union $U_k = \cup_{j=1}^{k} T_j$ of $T_1, T_2, \ldots, T_k$, is called *the sparse, weighted k-edge-connectivity certificate* of G. U_{k+1} has the following property.

Lemma 1 (Liang and Shen [16]). *If $e \in E - S$ is not an edge in U_{k+1}, then e is not an edge in the MST of graph $G(V, E - S)$ for any $S \subseteq E$, where $|S| \leqslant k$.*

Lemma 1 says that the k most vital edge problem on G is exactly equivalent to the k most vital edge problem on the sparse, weighted $(k+1)$-edge-connectivity certificate U_{k+1} of G. As a result, U_{k+1} instead of G, will be used to find the k most vital edges in G.

Let V_1 and V_2 be an arbitrary partition of the vertex set V of G, $V_1 \cup V_2 = V$ and $V_i \neq \emptyset$, $i = 1, 2$, and $Q = (V_1 \times V_2) \cap E$. It is easy to show that the minimum weighted edge in Q is an edge in the MST/MSF of G. Now assume that T is the MST of G and $e = (u, v)$ is an edge in T. Define the *ith minimum weighted replacement edge* $r_i(e)$ of e as follows. The vertex set V is partitioned into two subsets W and $V - W$ including u and v, respectively after removing edge e from T. Let $Q' = (W \times (V - W)) \cap E - \{e\}$, then $r_i(e)$ is the ith minimum weighted edge in Q'. Obviously $w(r_i(e)) < w(r_j(e))$ if $1 \leqslant i < j \leqslant |Q'|$.

Lemma 2. *Let $E(T) = \{e_1, e_2, \ldots, e_{n-1}\}$ be the edge set of the MST T. Assume that $r_j(e_i)$, $e_i \in E(T)$, is defined as above, then $r_j(e_i) \in U_{k+1}$ for all i and j, $1 \leqslant i \leqslant n-1$ and $1 \leqslant j \leqslant k$.*

Proof. For any edge $e \in E(T)$ and $e = (u, v)$, if we can show that $r_l(e) \in U_{l+1}$ for all l, $1 \leqslant l \leqslant k$, the lemma then follows. Assume that $r_j(e)$ is the first edge which is not in U_{j+1}, i.e., $r_1(e) \in U_2$, $r_2(e) \in U_3$, $\ldots, r_{j-1}(e) \in U_j$ but $r_j(e) \notin U_{j+1}$. Now consider the graph $G' = G - U_j$. Obviously edge $r_i(e)$ does not appear in G' for all i, $0 \leqslant i \leqslant j - 1$, assuming $r_0(e) = e$. Let T_{j+1} be the MST/MSF of G'. Assume that T_v and T_u are the subtrees containing v and u after deletion of e from T and V_1 and V_2 are the vertex sets of T_v and T_u. It is easy to see that $V_i \neq \emptyset$, $i = 1, 2$ and $V_1 \cup V_2 = V$. Let Q' be an edge set in G' in which the endpoints of the edges are in V_1 and V_2 respectively. Then $r_j(e)$ is the minimum weighted edge in Q' because the edges $e, r_1(e), r_2(e), \ldots, r_{j-1}(e)$ do not appear in G', and $r_j(e) \notin U_j$. By the property of the MST of G', $r_j(e)$ must be in T_{j+1}. Therefore, it must be in $U_{j+1} = U_j \cup T_{j+1}$. □

Given the MST T of G with fixed k, in the following we show that all $r_j(e_i)$s can be computed in time $O(n^2)$, for all i and j, $e_i \in E(T)$ and $1 \leqslant j \leqslant k$.

Lemma 3. *Let T be the MST of G, and $E(T) = \{e_1, e_2, \ldots, e_{n-1}\}$ be the edge set of T. The calculation of $r_j(e_i)$ for all i and j can be done in $O(n^2)$ time, where $1 \leqslant i \leqslant n - 1$, $1 \leqslant j \leqslant k$ with fixed k.*

Proof. For a given k, U_{k+1} can be found in $O((k+1)n^2) = O(n^2)$ time because finding an MST/MSF in an n-vertex graph takes $O(n^2)$ time and U_{k+1} is the union of the $(k+1)$ MSTs/MSFs. U_{k+1} instead of G then will be used to compute all $r_j(e_i)$ by Lemma 2, $1 \leqslant i \leqslant n-1$, $1 \leqslant j \leqslant k$. The algorithm proceeds as follows. For an edge $e=(u,v) \in E(T)$, the computation of all $r_j(e)$, $1 \leqslant j \leqslant k$, can be obtained by deleting e from T. As a result, T becomes two subtrees containing u and v, respectively. Label the vertices in each subtree by a unique identification, which takes $O(n)$ time. Let W and $V-W$ be the vertex sets containing u and v. Choose the jth minimum weighted edge from $Q'=(W \times (V-W)) \cap U_{k+1} - \{e\}$, which is exactly $r_j(e)$ by Lemma 2. This can be done in time $O(n)$ because $|Q'| \leqslant |U_{k+1}| < (k+1)n$ with fixed k. Therefore, it takes $O(k(k+1)n) = O(n)$ time to compute all $r_j(e)$, $1 \leqslant j \leqslant k$. The computation of all $r_j(e_i)$, $1 \leqslant i \leqslant n-1$ and $1 \leqslant j \leqslant k$, can be done in time $O(n^2)$. □

Despite the computation of all $r_j(e_i)$ taking $O(n^2)$ time, the computation of all $r_1(e_i)$ for all $e_i \in E(T)$ takes less time by utilizing a result in the design of an algorithm for the sensitivity analysis of minimum spanning trees, which is due to Dixon et al. [8] when G is a sparse graph.

Lemma 4 (Dixon et al. [8], Dixon and Tarjan [9]). *Let T be the MST of a weighted, connected graph $G(V,E)$ and $E(T)$ be the edge set of T, then all $r_1(e_i)$, $e_i \in E(T)$, can be found in time $O(\alpha(m,n)(m+n))$ at most, or in time $O(\alpha(m,n)\log n)$ using $O((m+n)/\log n)$ processors on a CREW PRAM, where $1 \leqslant i \leqslant n-1$, $|V|=n$ and $|E|=m$.*

Note that Lemma 4 is derived from [8,9] directly. In the design of the algorithm for sensitivity analysis of minimum spanning trees due to Dixon et al., one important component is to compute $r_1(e_i)$ for all $e_i=(u_i,v_i) \in E(T)$, which was defined as $b(u_i,v_i)$ in [8]. The time upper bound of their algorithm is $O((m+n)\alpha(m,n))$. Therefore, the time used for computing all $r_1(e_i)$ is no more than $O((m+n)\alpha(m,n))$, $1 \leqslant i \leqslant n-1$. Later Dixon and Tarjan [9] gave a parallel version of the sensitivity analysis algorithm. Their parallel algorithm requires $O(\alpha(m,n)\log n)$ time and $O((m+n)/\log n)$ processors on a CREW PRAM. Thus, the computation of all $r_1(e_i)$ can be done in the same amount of time and using the same number of processors on a CREW PRAM, $1 \leqslant i \leqslant n-1$.

3. Finding the k Most Vital Edges

In this section, we deal with the k most vital edge problem by proposing an improved algorithm for it. Before we proceed, let us recall the following fact. Let T be the MST of $G(V,E)$ and $E(T)=\{e_1,e_2,\ldots,e_{n-1}\}$ be the edge set of T. Assume that S^* is the set of k most vital edges in G whose deletion results in the maximum increase in the weight of the MST in $G(V,E-S^*)$. Then, $E(T) \cap S^* \neq \emptyset$. Assume that all $r_j(e_i)$ have been calculated for each tree edge $e_i \in E(T)$, for all i and j, $1 \leqslant i \leqslant n-1$ and

$1 \leqslant j \leqslant k$. We now can see that there is an i such that $|S^* \cap E(T)| = i$, $1 \leqslant i \leqslant k$, which can be further classified into two cases: (i) $|S^* \cap E(T)| = i$ with $1 \leqslant i < k$; and (ii) $|S^* \cap E(T)| = k$. We start by dealing with the case (i) $|S^* \cap E(T)| = i$ with $1 \leqslant i < k$ first.

Lemma 5. *Let S^* be the set of the k most vital edges and T be the MST in G. If $|S^* \cap E(T)| = i$ with $1 \leqslant i < k$, then the k most vital edges in G can be found in $O(n^{i+1})$ time.*

Proof. Assume that each tree edge $e \in E(T)$ has been assigned a unique number between 1 and $n-1$. There are $\binom{n-1}{i}$ combinations of i tree edges among $n-1$ tree edges. For each of the combinations of the i tree edges, assume that a copy of T is available. Delete the i tree edges in the combination from T, then T becomes a forest containing $i+1$ trees (or connected components). For each of the trees in the forest, label the vertices in a tree by a unique label (usually use the minimum vertex index of the tree to label all vertices in it). After that, a multi-graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$ is constructed, where each node in $\mathscr{V}$ corresponds to a tree in the forest and there is an edge between two nodes if there is an edge in U_{k+1} whose two endpoints are in the two trees. For any two nodes in $\mathscr{G}$, it is possible that there are many edges connected them, but only the first $(k-i+1)$th minimum weighted edges between them in U_{k+1} are kept, and all the other edges between them are useless and will be removed. As results, $\mathscr{G}$ is a multi-graph which contains no more than $i(i+1)(k-i+1)/2$ edges and $i+1$ nodes. It is easy to show that the other $k-i$ most vital edges in G must be a subset of these $i(i+1)(k-i+1)/2$ edges if the i tree edges are part of the k most vital edges in G. Since k is fixed, $\mathscr{G}$ is a graph with constant nodes and edges. Therefore, finding the $k-i$ most vital edges from $\mathscr{G}$ can be done in constant time, and their corresponding edges in G are the most vital edges. The construction of $\mathscr{G}$ takes $O(n)$ time because $|U_{k+1}| = O(n)$. For each combination of i tree edges among the $n-1$ tree edges, the other $k-i$ most vital edges (non-tree edges) can be found in $O(n)$ time. Thus, the i tree edges and the $k-i$ non-tree edges form a candidate of the solution. There are $\binom{n-1}{i}$ such candidates. Finally, a candidate that maximizes the weight of the MST in the resulting graph after deleting the k edges in the candidate from G will be chosen. Thus, the total time for finding the k most vital edges is $O(\binom{n-1}{i}n) = O(n^{i+1})$. □

We then consider the case (ii) $|S^* \cap E(T)| = k$. This means that k tree edges among the $n-1$ tree edges are the k most vital edges. Let $S^* = \{e_1^*, e_2^*, \ldots, e_{k-1}^*, e_k^*\}$, then $e_i^* \in E(T)$ for all $1 \leqslant i \leqslant k$. Assume that $e_1^*, e_2^*, \ldots, e_{k-1}^*$ have already been found, the problem is to find e_k^*. Let $T_{e_1^*, e_2^*, \ldots, e_{k-1}^*}$ be the MST in $G(V, E - \{e_1^*, e_2^*, \ldots, e_{k-1}^*\})$. Then, e_k^* is a tree edge in $T_{e_1^*, e_2^*, \ldots, e_{k-1}^*}$ whose deletion results in the maximum increase in the weight of the MST in the resulting graph. To find e_k^*, the algorithm of Dixon et al. [8] for finding the 1st minimum weighted replacement edge $r_1'(e)$ for each edge e in $T_{e_1^*, e_2^*, \ldots, e_{k-1}^*}$ can be applied. Assume that e' is in $T_{e_1^*, e_2^*, \ldots, e_{k-1}^*}$ and $e' \in E(T)$

such that

$$w(r_1'(e')) - w(e') = \max_{e \in E(T) \text{ and } e \notin \{e_1^*, e_2^*, \ldots, e_{k-1}^*\}} \{w(r_1'(e)) - w(e)\}.$$

Clearly, $e' = e_k^*$. We now prove the following lemma.

Lemma 6. *Let S^* be the set of the k most vital edges and T be the MST in G. If $|S^* \cap E(T)| = k$, then the k most vital edges in G can be found in $O(n^k \alpha((k+1)(n-1), n))$ time.*

Proof. From the discussion above, if the k most vital edges are the k tree edges of T, then the k most vital edge problem can be reduced to the single most vital edge problem. That is, for each combination of $k-1$ tree edges among the $n-1$ tree edges, the MST T_S of graph $G(V, E-S)$ is constructed, where S is the set of the $k-1$ tree edges in the combination. Clearly, all other tree edges in $E(T)-S$ must be in T_S, For each $e' \in E(T)-S$, find the 1st minimum weighted replacement edge $r_1'(e')$ of e' in T_S. Finally, find an $e'' \in E(T)-S$ to maximize the weight of the MST in $G(V, E-S-\{e''\})$. Thus, $S \cup \{e''\}$ forms a candidate of the final solution. There are $\binom{n-1}{k-1}$ such candidates because there are $\binom{n-1}{k-1}$ ways to choose the set S. From all the candidates, a candidate with maximizing the weight of the MST in the resulting graph after deletion of the edges in the candidate will be the solution of the problem.

We now analyze the time complexity of finding S^*. Assume that the MST T of G is given. For a given combination S of $k-1$ tree edges in $E(T)$, the minimum spanning tree T_S of $G(V, E-S)$ can be constructed in $O(n)$ time because T_S is built based on the information supplied by the MST T of G and U_{k+1}, while the number of edges in U_{k+1} is no more than $(k+1)(n-1)$. Finding the 1st minimum weighted replacement edges of all tree edges in T_S takes $O(n\alpha((k+1)(n-1), n))$ time by Lemma 4, using $U_{k+1} - S$. It takes $O(n)$ time to find an edge $r_1'(e'')$ from the $n-1$ 1st minimum weighted replacement edges which maximizes the weight of the MST in the resulting graph by deleting its corresponding tree edge $e'' \in E(T)$. Thus, finding a candidate $S \cup \{e''\}$ takes $O(n\alpha((k+1)(n-1), n))$ time. Therefore, the total time for finding S^* is $O(\binom{n-1}{k-1} n\alpha((k+1)(n-1), n)) = O(n^k \alpha((k+1)(n-1), n))$ if the k most vital edges in G are the tree edges. □

Now we are ready to give the detail of the proposed algorithm for finding the k most vital edges of G. Let $e_1^*, e_2^*, \ldots, e_{k-1}^*, e_k^*$ be the k most vital edges in G and $W_{\max}$ be the weight of the MST in the resulting graph after deleting the k most vital edges. The proposed algorithm is presented in Fig. 1.

The correctness of the proposed algorithm is justified by the discussion above; and its time complexity is analyzed in Lemmas 5 and 6. We, therefore, have the following theorems.

Theorem 1. *Given a weighted, connected, undirected graph $G(V, E)$, the k most vital edge problem with respect to minimum spanning trees can be solved in time $O(n^k \alpha((k+1)(n-1), n))$ with fixed k ($\geqslant 2$).*

Procedure **Find_k_Vital_Edges**(G, V, E)
1. find the MST T and the U_{k+1} of G;
2. W_{max}:=$w(MST)$; /* the weight of the MST T of G */
3. **for** each edge $e_i \in E(T)$ **do**
   compute the edge $r_j(e_i)$ using the subgraph U_{k+1} of G, $j = 1, 2, \ldots, k$;
   endfor;
4. **for** i:=1 **to** $k-1$ **do**
   $N_{n-1,i}$:=$\binom{n-1}{i}$;
5. **for** j:=1 **to** $N_{n-1,i}$ **do**
   5.1 $W_{i,j} = w(MST)$; /* initial assignment of the MST of the resulting graph */
   /* after deleting the k most vital edges from G. Let $e_{j_1}, e_{j_2}, \ldots, e_{j_i}$ be */
   /* the i tree edges whose combinatorial numbering is j. */
   5.2 $W_{i,j}$:=$W_{i,j} - \sum_{l=1}^{i} w(e_{j_l})$;
   5.3 construct the multi-graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$;
   5.4 find the MST T' of $\mathscr{G}$;
   5.5 find the $k-i$ most vital edges from $\mathscr{G}$;
   /* Let $w(\mathscr{G})$ be the weight of the MST of the graph after deleting */
   /* the $k-i$ most vital edges from $\mathscr{G}$ and let $e_1'', e_2'', \ldots, e_{k-i}''$ be */
   /* the corresponding $k-i$ edges in G */
   5.6 $W_{i,j}$:=$W_{i,j} + w(\mathscr{G})$;
   5.7 **if** $W_{max} < W_{i,j}$ **then**
   5.7.1 W_{max}:=$W_{k,j}$;
   5.7.2 **for** l:=1 **to** i **do** $e_j^* = e_{j_l}$ **endfor**;
   5.7.3 **for** l:=$i+1$ **to** k **do** $e_j^* = e_l''$ **endfor**
   endif;
   endfor;
   endfor;
6. M:=$\binom{n-1}{k-1}$;
7. **for** j:=1 **to** M **do**
   7.1 $W_{k,j} = w(MST)$; /* the initial weight of the MST of the resulting graph */
   /* after deleting the k most vital edges from G. Let $e_{j_1}, e_{j_2}, \ldots, e_{j_{k-1}}$ be */
   /* the $k-1$ tree edges whose combinatorial numbering is j. */
   7.2 construct the MST T' of the resulting graph after deleting $e_{j_1}, e_{j_2}, \ldots, e_{j_{k-1}}$ from G;
   7.3 $W_{k,j}$:=$w(T')$;
   7.4 find the 1st minimum weighted replacement edge $r_1'(e)$ of each edge e in T',
   using Dixon's algorithm on graph $U_{k+1} - \{e_{j_1}, e_{j_2}, \ldots, e_{j_{k-1}}\}$;
   7.5 Let $e'' \in E(T')$ be an edge in $E(T)$ which satisfies
   $w(r_1'(e'')) - w(e'') = \max_{e \in E(T) \text{ and } e \notin \{e_{j_1}, e_{j_2}, \ldots, e_{j_{k-1}}\}} \{w(r_1'(e)) - w(e)\}$.
   7.6 $W_{k,j}$:=$W_{k,j} + w(r_1'(e'')) - w(e'')$;
   7.7 **if** $W_{max} < W_{k,j}$ **then**
   7.7.1 W_{max}:=$W_{k,j}$;
   7.7.2 **for** l:=1 **to** $k-1$ **do** $e_j^* = e_{j_l}$ **endfor**;
   7.7.3 $e_k^* = e''$
   endif;
   endfor.

Fig. 1. The algorithm for the k most vital edge problem

Proof. By Lemmas 5 and 6, the Theorem follows. □

Theorem 2. *Given a weighted undirected graph $G(V,E)$, the k most vital edge problem with respect to a minimum spanning tree can be solved in time $O(\log n \log\log n)$ using $O(n^k/\log n)$ processors on a CREW PRAM with fixed k ($\geqslant 2$).*

Proof. The proposed algorithm **Find_k_Vital_Edges** can be parallelized easily. The computational complexity of the parallel version is analyzed as follows. Step 1 requires $O(\log n)$ time and $O(m+n)$ processors on an EREW PRAM [4]. Step 2 takes $O(1)$ time using a processor. Step 3 takes $O(\log n)$ time using $O(n^2/\log n)$ processors, whose implementation detail is as follows. For every edge $e \in E(T)$, make a copy of T. Let this copy be T_e rooted at r, delete e from T_e, using the algorithm in [1] to label each subtree. This can be done in $O(\log n)$ time using $O(n/\log n)$ processors on an EREW PRAM. Label the endpoints of each edge in U_{k+1} by the subtree identifications they belong to, which requires $O(1)$ time and $O(|U_{k+1}|) = O(n)$ processors on a CREW PRAM. Find the 1st, 2nd, …, kth minimum weighted replacement edges from the set consisting of edges in U_{k+1} whose endpoints are labeled by different subtree identification. It is obvious that this step can be done in $O(\log n)$ time using $O(n/\log n)$ processors on an EREW PRAM. So, finding all $r_j(e)$ requires $O(\log n)$ time using $O(n^2/\log n)$ processors on a CREW PRAM for all $e \in E(T)$ and $j = 1, \ldots, k$. Steps 4 and 5 can be implemented in parallel. That is, we compute the k most vital edges in parallel. For every i and j, Step 5.3 requires $O(\log n)$ time and $O(n/\log n)$ processors. All other steps within Step 5 takes constant time using $O(1)$ processors. Thus, the total time for Step 4 is $O(\log n)$ and the total number of processors are $O(\sum_{i=1}^{k-1} \sum_{j=1}^{N_{n-1,i}} c''n/\log n) = O(n^k \log n)$ where c'' is constant. Step 7 can also be implemented in parallel. For each j, Step 7.2 takes $O(\log n)$ time and $O(n/\log n)$ processors because the information of the MST T can be used. Step 7.4 takes $O(\alpha((k+1)(n-1), n)\log n)$ time and $O(((k+1)(n-1)+n)/\log n) = O(n/\log n)$ processors by Lemma 4. Step 7.5 takes $O(\log n)$ time and $O(n/\log n)$ processors; and Steps 7.6 and 7.7 take $O(1)$ time and $O(1)$ processors at most. Therefore, the total time for Step 7 is $O(\alpha((k+1)(n-1), n)\log n) = O(\log n \log\log n)$ and the total number of processors is $O(Mn/\log n) = O(n^k/\log n)$ because $\log\log n > \alpha((k+1)(n-1), n)$ when n is enough large. Thus, the algorithm requires $O(\log n \log\log n)$ time and $O(n^k/\log n)$ processors. □

4. Conclusions

In this paper, an improved algorithm for finding the k most vital edges with respect to a minimum spanning tree has been suggested for fixed k, $k \geqslant 2$. The proposed algorithm runs in time $O(n^k \alpha((k+1)(n-1), n))$, which improves a previously known result by a factor of $O(n/\alpha((k+1)(n-1), n))$ where α is a slow growing function which is constant in most practical cases. It is also shown that the proposed algorithm can be parallelized easily. Its parallel version requires $O(\alpha((k+1)(n-1), n)\log n)$ time and $O(n^k/\log n)$ processors on a CREW PRAM.

Acknowledgements

We appreciate an anonymous referee for his/her invaluable suggestions and comments which help us improve the paper's quality and presentation.

References

[1] K. Abrahamson, N. Dadoun, D.G. Kirkpatrick, T. Przytycka, A simple parallel tree contraction algorithm J. Algorithms 10 (1989) 287–302.
[2] A. Aho, J. Hopcroft, J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.
[3] F.Y. Chin, J. Lam, I.N. Chen, Efficient parallel algorithms for some graph problems, Comm. ACM 25 (1982) 659–665.
[4] K.W. Chong, Y. Han, T.W. Lam, On the parallel time complexity of undirected connectivity and minimum spanning trees, Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, January, 1999, pp. 225–234.
[5] R. Cole, P.N. Klein, R.E. Tarjan, A linear work parallel algorithm for minimum spanning trees, Proceedings of the Sixth Annual ACM Symposium on Parallel Algorithms and Architectures, June 1994, pp. 11–15.
[6] R. Cole, P.N. Klein, R.E. Tarjan, Finding minimum spanning forests in logarithmic time and linear work, Proceedings of the eighth Annual ACM Symposium on Parallel Algorithms and Architectures, June 1996, pp. 243–250.
[7] T.H. Cormen, C.N. Leiserson, R.L. Rivest, Intorduction to Algorithms, MIT Press, Cambridge, MA, 1998.
[8] B. Dixon, M. Rauch, R.E. Tarjan, Verification and sensitivity analysis of minimum spanning trees in linear time, SIAM J. Comput. 21 (1992) 1184–1192.
[9] B. Dixon, R.E. Tarjan, Optimal Parallel Verification of Minimum Spanning Trees in Logarithmic Time, Lecture Notes in Computer Science, Vol. 805, Springer, Berlin, 1994, pp. 13–22.
[10] G.N. Frederickson, R. Solis-Oba, Increasing the weight of minimum spanning trees, Proceedings of the seventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 1996, pp. 539–546.
[11] H.N. Gabow, Z. Galil, T. Spencer, R.E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, Combinatorica 6 (2) (1986) 109–122.
[12] L. Hsu, R. Jan, Y. Lee, C. Hung, Finding the most vital edge with respect to minimum spanning tree in a weighted graph, Inform. Proc. Lett. 39 (1991) 277–281.
[13] L. Hsu, P. Wang, C. Wu, Parallel algorithms for finding the most vital edge with respect to minimum spanning tree Parallel Comput. 18 (1992) 1143–1155.
[14] K. Iwano, N. Katoh, Efficient algorithms for finding the most vital edge of a minimum spanning tree, Inform. Proc. Lett. 48 (1993) 211–213.
[15] D.R. Karger, P.N. Klein, R.E. Tarjan, A randomized linear-time algorithm to find minimum spanning trees, J. ACM 42 (1995) 321–328.
[16] W. Liang, X. Shen, Finding the k most vital edges in the minimum spanning tree problem, Parallel Comput. 23 (1997) 1889–1907.
[17] H. Nagamochi, T. Ibaraki, A linear time algorithm for finding a sparse k-connected subgraph of a k-connected graph, Algorithmica 7 (1992) 583–596.
[18] H. Nagamochi, T. Ibaraki, Computing edge-connectivity in multigraphs and capacitated graphs, SIAM J. Discrete Math. 5 (1992) 54–66.
[19] H. Shen, Improved parallel algorithms for the most vital edge of a graph with respect to minimum spanning tree, Technical Report, Department of Computer Science, Abo Akademi University, Finland, 1995.
[20] H. Shen, Finding the k most vital edges with respect to minimum spanning tree, Acta Inform. 36 (1999) 405–424.