ELSEVIER

Information Processing Letters 63 (1997) 277-282

Information Processing Letters

The minimum labeling spanning trees

Ruay-Shiung Chang *, Shing-Jiuan Leu

Department of Information Management, National Taiwan Institute of Technology, Taipei, Taiwan, ROC

Received 19 August 1996
Communicated by T. Asano

Abstract

One of the fundamental problems in graph theory is to compute a minimum weight spanning tree. In this paper, a variant of spanning trees, called the minimum labeling spanning tree, is studied. The purpose is to find a spanning tree that tries to use edges that are as similar as possible. Giving each edge a label, the minimum labeling spanning tree is to find a spanning tree whose edge set consists of the smallest possible number of labels. This problem is shown to be NP-complete even for complete graphs. Two heuristic algorithms and an exact algorithm, based on the A^*-algorithm, are presented. According to the experimental results, one of the heuristic algorithms is very effective and the exact algorithm is very efficient. © 1997 Elsevier Science B.V.

Keywords: Graph theory; Spanning trees; NP-complete; Analysis of algorithms

1. Introduction

Computing a *minimum weight spanning tree* (MWST) is one of the fundamental and classic problems in graph theory. Given an undirected graph G with nonnegative weight on each edge, the MWST of G is the tree spanning G having the minimum total edge weight among all possible spanning trees [1]. This problem and its related problems, k smallest spanning tree [5], edge update of minimum spanning tree [10], minimum diameter spanning tree [8], min-max spanning tree [2], most and least uniform spanning trees [3,7] and so on, have been intensely studied. Minimum weight spanning trees have applications in many areas, including network design, VLSI, and geometric optimization [4,11].

In this paper, the minimum labeling spanning tree is defined and studied. The purpose is to construct a spanning tree using edges which are as similar as possible. For example, in communication networks, there are many different types of communication medium, such as optic fiber, cable, microwave, telephone line and so on [12]. A communication node may communicate with different nodes by choosing different types of communication medium. Given a set of communication networks nodes, the problem we are interested is to find a spanning tree (a connected communication network) that uses as few types of communication lines as possible. This spanning tree will reduce the construction cost and complexity of the network. This problem can be formulated as a graph problem. Given a graph $G = (V,E)$ and a labeling function $L(e)$ for all edges $e \in E$ where vertices represent communication nodes. Edges and their labelings represent the communication lines and their types respec-

* Corresponding author. Email: rschang@cs.ntit.edu.tw.

PII S0020-0190(97)00127-0

tively. The objective is to find a spanning tree which uses the smallest number of different types of edges labels. Define L_T to be the set of different labels in edges for a spanning tree T. The *minimum labeling spanning tree* (MLST) problem is formally defined as follows.

Problem (*MLST problem*). Given a graph $G = (V, E)$ and a labeling function $L(e)$ for all $e \in E$, find a spanning tree T of G such that $|L_T|$ is minimized.

Reducing from the *minimum covering* problem [6], it is shown that the MLST problem is NP-complete even when restricted to complete graphs. Two heuristic algorithms and an algorithm to obtain an optimal solution are proposed. The optimal solution algorithm is based on the A^*-algorithm [9] concept. Experimental results indicate that the A^*-algorithm is quite efficient and one of the heuristics gives quite good answers.

The rest of this paper is organized as follows. In Section 2, The MLST problem is shown to be NP-complete even when restricted to complete graphs. In Section 3, two heuristic algorithms for the MLST problem and one algorithm that will produce an optimal solution are proposed. Experimental results for the algorithms are given in Section 4. Finally, conclusions and some open problems are discussed in Section 5.

2. NP-complete proof

First, a decision version, called the *bounded labeling spanning tree* (BLST) problem, for the MLST problem is defined and shown to be NP-complete.

Problem (*BLST problem*). Given a graph $G = (V, E)$ and a labeling function $L(e)$ for all $e \in E$ and a positive integer K, is there a spanning tree T for G such that $|L_T| \leqslant K$.

Lemma 1. *BLST is* NP*-complete.*

Proof. It is easy to see that BLST $\in$ NP since a nondeterministic algorithm need only guess a subset of edges and check in polynomial time whether these edges connect all vertices and $|L_T|$ is the appropriate size.

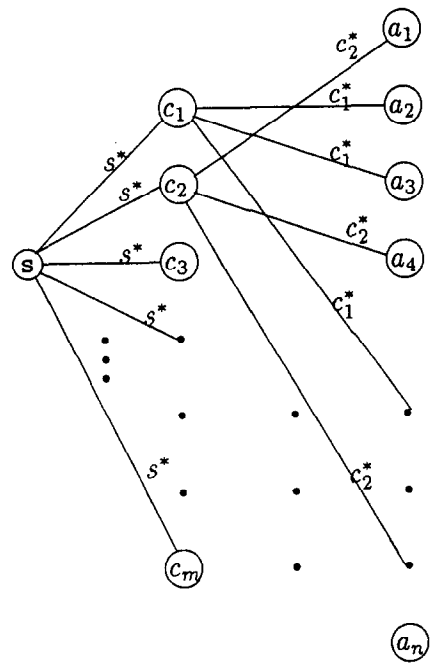

Fig. 1. Construction of G in the proof of Lemma 1.

It is proved by transforming the minimum covering problem to the BLST problem. Let $S = \{a_1, a_2, a_3, \ldots, a_n\}$ and $C_1 = \{a_{i_1}, a_{j_1}, a_{k_1}\}$, $C_2 = \{a_{i_2}, a_{j_2}, a_{k_2}\}$, $C_3 = \{a_{i_3}, a_{j_3}, a_{k_3}\}$, ..., and $C_m = \{a_{i_m}, a_{j_m}, a_{k_m}\}$ be subsets of S. The NP-complete minimum covering problem is to find a minimum number of subsets to cover all the element in S. From S and the C_i's, we will construct a graph $G = (V, E)$ such that there is a cover for S with $K - 1$ subsets if and only if G has MLST with K labels.

The construction of G is as follows (Fig. 1). G contains the following vertices: the element nodes $\{a_1, a_2, a_3, \ldots, a_n\}$, the subset nodes $\{C_1, C_2, C_3, \ldots, C_m\}$ and a special node s. The edges set contains the special edges $\{(s, C_i) \mid i = 1, \ldots, m\}$ and the covering edges $\{(C_p, a_{l_p}) \mid p = 1, \ldots, m, l = i, j,$ or k, where a_{l_p} is in the C_p subset$\}$. All the special edges have label s^*. The covering edges have label C_p^* for $p = 1$ to m depending on which C_p this edge connects. It can be shown that the construction can be accomplished in polynomial time and G has an MLST with K labels if and only if a minimum covering of size $K - 1$ exists. □

Corollary 2. *The MLST problem is* NP*-complete for complete graphs.*

Proof. The proof is similar to that of Lemma 1. Except in Fig. 1, all missing edges are added to make it complete. Each newly added edge is assigned a unique label. □

3. Algorithms for minimum labeling spanning tree

Given a graph $G = (V,E)$, with a label on each edge, the MLST problem is to find a spanning tree which uses the smallest number of different types of edge labels. As shown in Section 2, the MLST problem is NP-complete. In this section, two heuristic algorithms and an exact algorithm for finding an optimal solution for the MLST problem are proposed.

3.1. Heuristic algorithms

The first heuristic algorithm is based on the edge replacement concept. At first, find an arbitrary spanning tree. Then, in order to reduce the spanning tree's total number of edge labels, for each nontree edge, test if the tree after this nontree edge is included and a tree edge deleted will have the possibility of obtaining smaller number of labelings. The detail of this heuristic algorithm is described as follows.

Algorithm 1: Edge Replacement Algorithm

Input: A labeling graph $G = (V,E)$, with $|V| = n$, $|E| = m$ and $|L| = l$, where L is the set of possible labels for all edges.
Output: A spanning tree.

1. Find an arbitrary spanning tree of G.
2. For every nontree edge i
3. Let *label*(i) be the label of edge i. If *label*(i) has not appeared in the spanning tree, go to Step 7.
4. Find the cycle, say C, which is created by adding the edge i.
5. Let *current_label_count* be the number of times that *label*(i) appears in C.
   Let *minimum_label* be the label with the least number of appearances in C.
   Let *minimum_count* be the number of appearances of *minimum_label*.
6. If *current_label_count* > *minimum_count* and *label*(i) ≠ *minimum_label*, then put edge i into spanning tree and remove a tree edge, whose label is equal to *minimum_label*.
7. End (for i).

Theorem 3. *The time complexity of Algorithm* 1 *is* $O(mn)$.

Proof. At Step 2, there are $m-(n-1)$ nontree edges. We then spend $O(n)$ to find a cycle at Step 4. Therefore, this algorithm takes $O(mn)$. □

The second heuristic algorithm tries to construct the spanning tree gradually, each time selecting a label such that edges with this label cover as many uncovered vertices as possible. We repeat this procedure until all vertices are covered. The detail of this algorithm is described as follows.

Algorithm 2: Maximum Vertex Covering Algorithm

Input: A labeling graph $G = (V,E)$, with $|V| = n$, $|E| = m$ and $|L| = l$, where L is the set of possible labels for all edges.
Output: A spanning tree.

1. Let $H = (V,\phi)$ be the subgraph of G, which hasn't any edge.
2. *While* H is not connected
3. Find a unused label l such that edges with label l cover as many uncovered vertices as possible. If there are more than one candidate, select one randomly.
4. Add edges whose labels are l into the subgraph H.
5. End (for *while*).
6. Find an arbitrary spanning tree of H.

Theorem 4. *The time complexity of Algorithm* 2 *is* $O(lmn)$, *where* l *is the total number of different labels.*

Proof. At most, the *while* loop will take $O(n)$ times. We spend $O(lm)$ to find the maximum covering label at Step 3. Since Step 2 is the dominating step, the time complexity of this algorithm is $O(lmn)$. □

3.2. An optimal solution algorithm

The algorithm is based on the A^*-algorithm [9]. Basically, an A^*-algorithm is a tree-searching algorithm. It always selects a least cost node to expand for minimization problems. We first explain how to evaluate the node-cost-estimation function f. For any node x, $f(x)$ consists of two parts, $g(x)$ and $h(x)$, where $g(x)$ = the cost of the current search path from root node r to x with $g(r) = 0$ and $h(x)$ = an estimate of the cost of the best path from x to a goal node, where $h(\textit{goal node}) = 0$.

The A^*-algorithm states that if $h(x)$ is an underestimate, then the first goal node reached will be an optimal solution. In our problem, $g(x)$ is just the number of labels that have been used so far. $h(x)$ is calculated as follows.

(1) Let $l_1, l_2, \ldots, l_k$ be the unused labels and e_i be the number of edges with label l_i for $1 \leqslant i \leqslant k$. Without loss of generality, assume $e_1 \geqslant e_2 \geqslant \cdots \geqslant e_k$.
(2) Let $H = (V_c, E_c)$, where E_c consists of the edges whose labels have been selected so far and V_c consists of the vertices covered by edges in E_c.
(3) Let H_1 be the subgraph of H with an arbitrary edge of each fundamental cycle in H removed. (H_1 is a forest.)
(4) Let *edge_needed* = $(n-1) - (\#$ of edges in $H_1)$.
(5) $h(x)$ = the smallest j such that $\sum_{i=1}^{j} e_i \geqslant$ *edge_needed*.

It is obvious that $h(x)$ is an underestimate and can be computed in polynomial time.

Having described the cost function, we next present the A^*-algorithm.

Algorithm 3: An exact algorithm for the MLST problem

Input: A graph $G = (V, E)$ where each edge has a label in L and $|V| = n$, $|E| = m$, and $|L| = l$.
Output: A spanning tree with minimum number of tree edge labels.

1. Put the root node r on OPEN. /* OPEN is the storage place for all generated but unexpanded nodes. */
2. If OPEN is empty, exit with failure. /* Since a solution always exists, this will never be executed. */

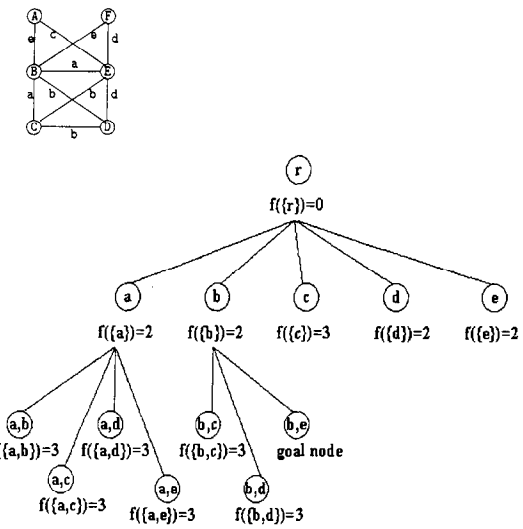

Fig. 2. The whole tree searched by the A^*-algorithm for the graph in the top-left of the figure.

3. Remove from OPEN and place on CLOSED a node n for which f is minimum. If there are more than one node with the same minimum f value, the latest generated node will be selected. /* CLOSED is the storage place for the expanded nodes. */
4. If n is a goal node (a spanning subgraph is formed), goto Step 8.
5. Otherwise expand n. If there are k unselected labels, then n has k children, one for each unselected label.
6. For each child n' of n:
   If n' is not already on OPEN or CLOSE, calculate $h(n')$ and $f(n') = g(n') + h(n')$ where $g(n') = g(n) + 1$ and $g(r) = 0$. Put n' into OPEN.
7. Goto Step 2.
8. Find a spanning tree of the subgraph.
9. End.

Applying the algorithm to the graph in the top-left of Fig. 2, the whole expanded tree is as shown in Fig. 2.

4. Experimental results

In this section, we study the performance of the proposed heuristic algorithms, Algorithms 1 and 2, and the MLST algorithm, Algorithm 3. These algo-

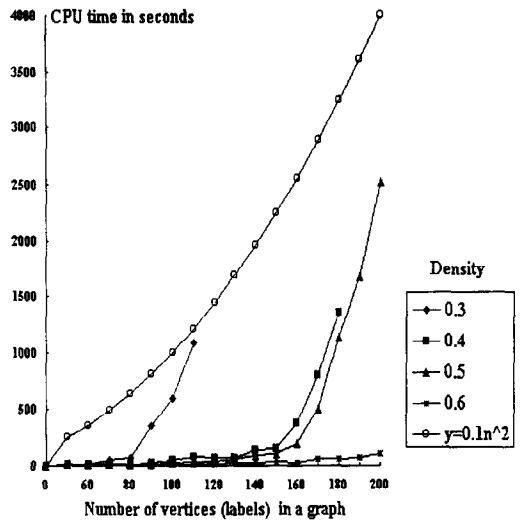

Fig. 3. The MLST algorithm's CPU time used in the graph with density 0.3, 0.4, 0.5 and 0.6., and the curve $y = 0.1n^2$.

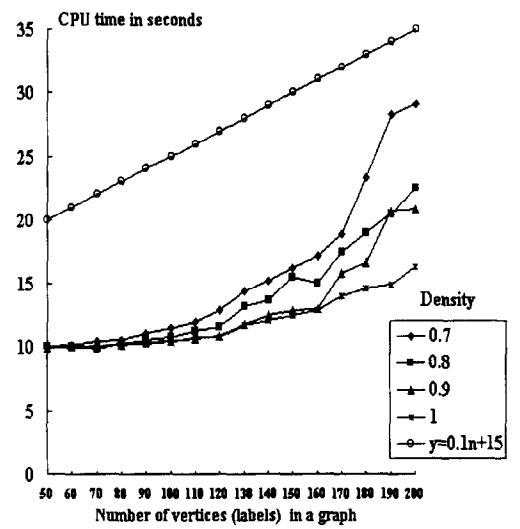

Fig. 4. The MLST algorithm's CPU time needed in the graph with density 0.7, 0.8, 0.9 and 1 and the curve of $y = 0.1n + 15$.

rithms are implemented using the C language and run on a SUN sparc-20 machine. Figs. 3 and 4 show the performance of the exact MSLT algorithm, Algorithm 3. Fig. 3 shows the relation of the CPU time used in seconds for various number of vertices for small density graphs. The number of labels is equal to the number of vertices. We also draw the curve $y = 0.1n^2$ for comparison. Fig. 4 shows the same simulation for large density graphs. The graph with density 1 is a complete graph. Those curves are always under the curve $y = 0.1n + 15$ and the CPU time needed is never more than 30 seconds. Fig. 5 shows that Heuristic 1 is not stable and Heuristic 2 is very effective.

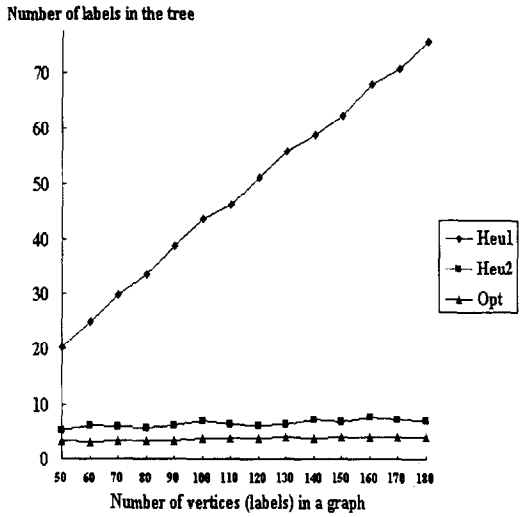

Fig. 5. Performance comparison with graphs of density 0.4.

5. Conclusion

The concept of the minimum labeling spanning tree is to find a spanning tree such that the spanning tree edges are as similar as possible. It is shown that the MLST problem is NP-complete, and two heuristic algorithms and an exact algorithm based on the A^*-algorithm are proposed. Possible future researches are to apply the minimum labelings definitions to other weighted optimization problems. For example, the minimum labeling shortest path problem, the minimum labeling maximum matching problem, the minimum labeling cut set problem and so on.

References

[1] B. Brassard and P. Bratley, *Algorithmics Theory and Practice* (Prentice-Hall, Englewood Cliffs, NJ, 1988).

[2] P.M. Camerini, The min-max spanning tree problem and some extensions, *Inform. Process. Lett.* 7 (1) (1978) 10–14.

[3] P.M. Camerini, F. Maffioli, S. Martello and P. Toth, Most and least uniform spanning trees, *Discrete Appl. Math.* 15 (1986) 181–197.

[4] W.-T. Chen and N.-F. Huang, The strongly connecting problem on multihop packet radio networks, *IEEE Trans. Comm.* 37 (3), pp. 293–295.

[5] D. Eppstein, Finding the k smallest spanning trees, *BIT* 32 (1992) 237–248.

[6] M.R. Garey and D.S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).

[7] Z. Galil and B. Schieber, On finding most uniform spanning trees, *Discrete Appl. Math.* 20 (1988) 173–175.

[8] J.-H. Ho, D.T. Lee, C.-H. Chang and C.K. Wong, Minimum diameter spanning trees and related problems, *SIAM J. Comput.* 20 (1991) 987–997.

[9] N.J. Nillson, *Principles of Artificial Intelligence* (Tioga, Palo Alto, CA, 1981).

[10] X. Shen and W. Liang, A parallel algorithm for multiple edge undates of minimum spanning trees, in: *Proc. 7th Internat. Parallel Processing Symp.* (1993) 310–317.

[11] R. Simha and B. Norahari, Single path routing with delay considerations, *Computer Network ISDN Systems* 24 (1992) 405-419.

[12] A.S. Tanenbeum, *Computer Networks* (Prentice-Hall, Englewood Cliffs, NJ, 1989).