ELSEVIER

Information Processing Letters 66 (1998) 81–85

Information Processing Letters

On the minimum label spanning tree problem

Sven O. Krumke *, Hans-Christoph Wirth [1]

Department of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany

Received 12 November 1997; revised 18 February 1998
Communicated by H. Ganzinger

Abstract

We study the *Minimum Label Spanning Tree Problem*. In this problem, we are given an undirected graph whose edges are labeled with colors. The goal is to find a spanning tree which uses as little different colors as possible.

We present an approximation algorithm with logarithmic performance guarantee. On the other hand, our hardness results show that the problem cannot be approximated within a constant factor. © 1998 Elsevier Science B.V.

Keywords: Algorithms; NP-hardness; Approximation algorithms; Spanning tree

1. Introduction

Problems of finding spanning trees optimizing some measure have been extensively studied in the literature. Typical measures include the weight or the diameter of the tree. In communication network design, it is also often desirable to obtain a tree that is "most uniform" in some specified sense.

Motivated by this observation, in [2] Chang and Leu introduce the minimum label spanning tree problem. In this problem, we are given a graph with colored edges, and one seeks to find a spanning tree with the least number of colors possible.

Chang and Leu proved the NP-hardness and provided an exponential time algorithm for the problem. They also gave two heuristics to obtain (possibly suboptimal) solutions in polynomial time and evaluated the quality of their algorithms by experimental studies. They observe that the first of their heuristics is often far off from the optimum, while the second one performs well in practice.

In this paper, we confirm their results by theoretical performance analysis. We give an example that shows that the first heuristic might perform essentially as badly as possible. On the other hand, we prove a logarithmic performance guarantee for the second heuristic. We also show that there cannot exist a polynomial time constant factor approximation for the minimum label spanning tree problem unless P = NP.

2. Problem definition and preliminaries

The minimum label spanning tree problem can be defined formally as follows:

Definition 1 (*Minimum label spanning tree problem, MLST*). Let $G = (V, E)$ be a connected undirected graph and $c : E \to \mathbb{N}$ be an edge labeling/coloring function. A K-colored spanning tree (V, T) is a spanning tree of G such that the number of used colors $|\{c(e) \mid e \in T\}|$ does not exceed K. A minimum

* Corresponding author. Email: krumke@informatik.uni-wuerzburg.de.

[1] Email: wirth@informatik.uni-wuerzburg.de.

PII S0020-0190(98)00034-9

Input: A graph $G = (V, E)$, and edge labels $c : E \to \{1, \ldots, m\}$
1 Let $C \leftarrow \emptyset$ the set of used colors
2 **repeat**
3 Let H be the subgraph of G restricted to edges with colors from C
4 Contract each connected component in H to a supernode
5 **for all** $i \in \{1, \ldots, m\} \setminus C$ **do**
6 Determine the number of connected components when inserting all edges with color i in H
7 **end for**
8 Choose color i with smallest resulting number of components: $C \leftarrow C \cup \{i\}$
9 **until** H is connected

Fig. 1. Algorithm 1. Approximation algorithm for MLST.

label spanning tree is a K-colored spanning tree with minimum K.

As mentioned before, the problem has been shown to be NP-hard [2]. Thus, we are interested in approximation algorithms. Recall that an approximation algorithm with performance α for a minimization problem Π is a polynomial time algorithm with the following property: for every instance I of Π with optimal solution value $\mathrm{OPT}(I)$, the algorithm finds a solution of value at most $\alpha \cdot \mathrm{OPT}(I)$ (cf. [5]).

The first heuristic proposed in [2] works as follows: starting with an arbitrary spanning tree, the algorithm considers each non-tree edge once and checks whether the number of used colors can be reduced by swapping the edge for a tree edge on the induced cycle. The edge to remove from the tree is chosen as an edge whose color appears least in the current tree.

The following example shows that the number of colors in the resulting tree might be as large as $n - 1$ on an n-node graph which has a 2-colored spanning tree: choose a star shaped graph where each of the $n - 1$ edges has a different color. Then, make the graph complete by adding edges of a new color. If the algorithm starts with the star as the initial spanning tree, no swaps will be performed and the algorithm terminates with a tree of $n - 1$ colors. On the other hand, the edges of the new color and one additional edge of the star can be used to construct a spanning tree that uses only 2 colors. Thus the heuristic is off by a factor of $\Omega(n)$ which is as bad as possible.

3. Approximation algorithm and performance guarantee

In this section we present a polynomial time algorithm which calculates an approximate solution of the MLST problem and prove a logarithmic performance guarantee.

Outline of the algorithm. We first give a sketch of the algorithm, which is essentially the second heuristic proposed in [2]. See Algorithm 1 (Fig. 1) for a detailed description. The algorithm starts with an empty set of edges. Then, it iteratively selects one color and inserts all edges of that color in the graph until the graph is connected. Hereby the algorithm tests all unused colors and chooses a color in such a way that the decrease in the number of connected components is as large as possible.

Running time. Let $G = (V, E)$ be the input graph. Denote by $n = |V|$ and $m = |E|$ the number of vertices and edges, respectively. Further, let l be the number of colors. We show that the algorithm can be implemented to run in time $O(m + l\,\alpha(m, n) \cdot \min\{ln, m\})$, where α is the inverse of Ackerman's function.

The implementation uses data structures for disjoint sets [3]. Recall that, by using the path compression and union-by-rank heuristic for t Union-Find operations on s elements, one obtains a running time of $O(t\,\alpha(t, s))$.

We first do a preprocessing step to reduce the number of edges of each color to at most $n-1$. Let $E_c \subseteq E$ be the set of edges of color c. We compute for each color c the connected components of (V, E_c) and then replace E_c by the edge set E'_c of a spanning forest of that graph. This procedure can be carried out by depth-first search (see, e.g., [3]) in time

$$O\left(\sum_{c=1}^{l}(n+|E_c|)\right) \subseteq O(ln+m).$$

Notice that the edge sets E_c and E'_c induce the same components on G and therefore we can restrict the algorithm to the smaller sets E'_c.

In each iteration, the algorithm has to test at most l colors, where for each color we have to determine the number of connected components when all edges of the specified color are added to the current graph. For each yet unselected color c, this can be achieved by $O(|E'_c|)$ Union-Find operations on the set of nodes of H. This results in a total time per iteration of

$$O\left(\sum_{c=1}^{l}|E'_c|\alpha(|E'_c|,n)\right) \subseteq O\left(\alpha(m,n)\cdot\sum_{c=1}^{l}|E'_c|\right).$$

Since on the one hand $|E'_c| \leqslant n-1$ for each color c, and on the other hand $\sum_{c=1}^{l}|E_c| \leqslant m$, we obtain that the work in each iteration can be bounded by $O(\alpha(m,n)\cdot\min\{ln,m\})$.

Using the fact that the number of iterations is at most l, yields a total running time as stated above.

Performance guarantee. We now prove a logarithmic performance guarantee of the algorithm stated above. At first we show that in each iteration there is a color which reduces the number of connected components almost by a constant factor.

Lemma 2. *Let H be a connected graph with r nodes which has a p-colored spanning tree. Then there is a color i such that H, restricted to edges of color i, has no more than $r(1-1/p)+1/p$ connected components.*

Proof. Let T be the p-colored spanning tree. Then T contains $r-1$ edges, and there is a color i such that the number of i-colored edges in T is at least $\lceil (r-1)/p \rceil$. Remove all edges from H and then reinsert all tree edges of color i one by one. Each time inserting one edge, the number of connected components decreases by one, since otherwise the edges would form a cycle in the tree T. Therefore, the resulting graph has at most $r-\lceil (r-1)/p \rceil = \lfloor r(1-1/p)+1/p \rfloor$ connected components. □

Let p be the number of colors in the optimal spanning tree. Define $f(n) := n(1-1/p)+1/p$. Then, by Lemma 2 and the fact that the algorithm chooses the best possible color in each iteration, it follows that $f^k(n)$ is an upper bound on the number of connected components after iteration k. Here, f^k denotes the k-fold iteration of f.

Lemma 3. *Let $f(n) = n(1-1/p)+1/p$ for some fixed $p \geqslant 1$. Then,*

$$f^k(n) \leqslant n\left(1-\frac{1}{p}\right)^k + \frac{k}{p}.$$

Proof. We establish the claim by induction on k. The claim is trivial for $k=0$. Assume that it is correct for k. Then,

$$\begin{aligned} f^{k+1}(n) &= f^k(n)\left(1-\frac{1}{p}\right)+\frac{1}{p} \\ &\leqslant n\left(1-\frac{1}{p}\right)^k\left(1-\frac{1}{p}\right)+\frac{k}{p}\left(1-\frac{1}{p}\right)+\frac{1}{p} \\ &\leqslant n\left(1-\frac{1}{p}\right)^{k+1}+\frac{k+1}{p}\left(1-\frac{1}{p}\right). \end{aligned}$$ □

The following lemma can be shown by elementary analysis:

Lemma 4. *For $n, p \in \mathbb{N}$ we have:*

$$n(1-1/p)^k \leqslant p \quad \text{if } k \geqslant p\ln n/p.$$

Proof. Since the expression given on the left hand side of the inequality is decreasing in k, it suffices to show that $n(1-1/p)^k \leqslant p$ for $k = p\ln np$. Taking the natural logarithm on both sides and plugging in the value of k, we obtain that this is equivalent to

$$p\ln\frac{n}{p}\ln\frac{p}{p-1} \geqslant \ln\frac{n}{p} \iff p\ln\frac{p}{p-1} \geqslant 1.$$

The function $p \mapsto p\ln p/(p-1)$ is non-increasing and has limit of 1 for $p \to \infty$. Thus, $p\ln p/(p-1) \geqslant 1$ for all $p > 1$, and the claim follows. □

We now are able to bound the number of iterations the algorithm performs.

Lemma 5. *The number of iterations is at most*

$$p \ln \frac{n}{p} + p + \ln \frac{n}{p}.$$

Proof. By Lemmas 3 and 4, after $k = \lceil p \ln n/p \rceil$ iterations the number of connected components has fallen from n to at most $k' = \lfloor p + \ln n/p \rfloor$. Since the number of connected components decreases by at least one in each iteration, the algorithm terminates after no more than $k' - 1$ additional iterations. □

Taking these results, the performance guarantee of the algorithm follows:

Theorem 6. *The algorithm described above has a performance guarantee of* $2 \ln n + 1$ *for the* MLST *problem. More precisely, if G is an n-node graph which has a p-colored spanning tree, then the algorithm finds a spanning tree of G which has at most*

$$p \ln \frac{n}{p} + p + \ln \frac{n}{p} \leqslant p(2 \ln n + 1)$$

colors.

4. Hardness result

In this section we show that the MLST problem cannot be approximated within a constant factor.

Theorem 7 (Non-approximability of MLST). *Unless* P = NP, *for any constant α there is no α-approximation algorithm for* MLST.

Proof. We give a reduction from MINSETCOVER. An instance of MINSETCOVER is given by a finite set $Q = \{q_1, \ldots, q_s\}$ of ground elements and a family $F = \{Q_1, \ldots, Q_l\}$ of subsets of Q. The problem consists of finding a covering $C \subseteq F$ of Q with minimum number $|C|$ of sets. The problem MINSETCOVER is known to be NP-hard [4, Problem SP5]. Moreover, there is a constant $\eta > 1$ such that, unless P = NP, MINSETCOVER cannot be approximated within a factor of $\eta \ln |Q|$ [1].

Given an instance of MINSETCOVER, for each $j = 1, \ldots, l$ we construct an instance I_j of MLST as follows: The node set of graph I_j consists of Q (the element nodes), F (the set nodes), and an additional node r (the root). The root is connected to each set node via an edge of color j. Further, the graph contains an edge (q_ν, Q_μ) of color μ between an element node q_ν and a set node Q_μ if and only if $q_\nu \in Q_\mu$. Observe that the construction can be performed in polynomial time.

It is easy to see that there is a covering $C \subseteq F$ of Q with no more than K subsets if and only if there is a $j \in \{1, \ldots, l\}$ such that instance I_j contains a spanning tree with edges of no more than K different colors.

Moreover, given a spanning tree of instance I_j with k different colors, we can easily obtain a set cover of size at most k in the following way: if color μ appears in the tree, then choose set Q_μ to be in the cover. (Note that if color j merely appears in edges adjacent to the root, we can remove set Q_j from the cover and obtain a valid cover of size $k - 1$.)

For the instance I of MINSETCOVER, let OPT(I) be the minimum number of sets which form a covering. As noted above, there is an instance I_j of MLST containing a spanning tree of no more than OPT(I) colors. Let A be an approximation algorithm for MLST with constant approximation ratio α. When running A on all instances $I_1, \ldots, I_l$ constructed as described above, A finds on instance I_j a spanning tree of no more than $\alpha \cdot \mathrm{OPT}(I)$ colors. Thus, if we take the best solution over all runs of the algorithm, we obtain an α-approximation for the MINSETCOVER instance I. This is a contradiction to the results of [1]. □

Acknowledgements

The authors wish to thank Dorit Hochbaum for suggesting a far more efficient implementation of the algorithm than in the original version of the paper.

References

[1] S. Arora, M. Sudarn, Improved low-degree testing and its applications, in: Proc. 29th Annual ACM Symposium on the Theory of Computing (STOC'97), 1997, pp. 485–496.

[2] R.-S. Chang, S.-J. Leu, The minimum labeling spanning trees, Inform. Process. Lett. 63 (1997) 277–282.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, 1990.
[4] M.R. Garey, D.S. Johnson, Computers and Intractability (A Guide to the Theory of NP-Completeness), W.H. Freeman, New York, 1979.
[5] D.S. Hochbaum (ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing Company, Boston, MA, 1997.