SIAM J. COMPUT.
Vol. 4, No. 3, September 1975

ON FINDING AND UPDATING SPANNING TREES AND SHORTEST PATHS*

P. M. SPIRA† AND A. PAN‡

Abstract. We consider one origin shortest path and minimum spanning tree computations in weighted graphs. We give a lower bound on the number of analytic functions of the input computed by a tree program which solves either of these problems equal to half the number of worst-case comparisons which well-known algorithms attain. We consider the work necessary to update spanning tree and shortest path solutions when the graph is altered after the computation has terminated. Optimal or near-optimal algorithms are attained for the cases considered. The most notable result is that a spanning tree solution can be updated in $O(n)$ when a new node is added to an n-node graph whose minimum spanning tree is known.

Key words. spanning trees, shortest paths, lower bounds on computation, graph computations

1. Synopsis of results. Dijkstra [2] has given an algorithm to find all shortest paths from a single origin in a directed graph with positive arc weights and Prim [1] has given an algorithm to find a minimal spanning tree in an undirected graph. We discuss the optimality of these algorithms in the sequel and show that no program whose unit operation is the evaluation and testing for positivity of an analytic function of the weights can better these algorithms by more than a factor of two. We then consider the problem of updating previous shortest path and minimum spanning tree solutions when parameters of the graph are changed. We consider what must be done when nodes are added or deleted and when weights on arcs are increased or decreased. We obtain lower bounds and optimal or near optimal algorithms for these problems in terms of how many analytic functions of the weights must be considered.

2. Definitions and preliminaries. Let G be an n-node with d_{ij} the distance from node i to node j so that G is undirected if $d_{ij} = d_{ji}$ for all i and j.

DEFINITION 2.1. An *analytic tree program* T is one defined by a rooted tree. Each internal node and the root are labeled by analytic functions, and each leaf is labeled by an answer—the output of the program. Computation begins at the root. At each node the analytic function is evaluated and the next node visited is the left or right successor of the present node. Computation terminates when a terminal node is reached. The *depth of* T, $d(T)$, is the length of the longest branch.

DEFINITION 2.2. Let $l_1, \cdots, l_m$ be linear maps from R^d to R, where R is the real numbers, and let $C \subseteq R^d$ be a convex set. Let $L^+ = \{x \in R^d : l_i(x) \geqq 0, 1 \leqq m\}$. A *complete analytic proof of* L^+ *on* C is a matrix

$$\mathscr{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ \vdots & \vdots & & \vdots \\ p_{r1} & p_{r2} & \cdots & p_{rk} \end{bmatrix},$$

* Received by the editors January 1, 1974, and in revised form July 26, 1974. This research was supported in part by the U.S. Army Research Office, Durham, under Contract DA-ARO-D-31-124-71-6174 and in part by the National Science Foundation under Grant GJ-356 04.

† Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California at Berkeley, Berkeley, California 94720.

‡ Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California at Berkeley, Berkeley, California. Now at Bell Laboratories, Naperville, Illinois 60540.

where each $p_{ij}:R^d \to R$ is analytic and such that $x \in L^+ \Leftrightarrow \exists i, 1 \leqq i \leqq k$, with $p_{ij}(x) \geqq 0, 1 \leqq j \leqq k$. We call k the *width of $\mathcal{P}$.*

The reason for defining a complete proof is that any lower bound on the width of a complete proof for L^+ is a lower bound on the depth of a tree program. In fact, we have the following lemma.

LEMMA 2.3. *Let $l_1, \cdots, l_m$ be linear maps from R to R. Let $C \subseteq R^d$. Let k be the minimum width of any analytic proof of L^+ on C. Let T be a program which, given any point $x \in C$, determines whether or not $x \in L^+$. Then $d(T) \geqq k$.*

Proof. The proof is direct from the definition of complete analytic proof. Q.E.D.

We shall use in all our lower bound proofs the following theorem.

THEOREM 2.4 (Rabin). *Let $l_1, \cdots, l_m$ be linear forms from R^d to R, with $m \leqq d$. Let $C \subseteq R^d$ contain a point for any given of the 3^m possible $+, 0, -$ sign conditions of the l_i. Then any complete analytic proof of L^+ on C has width at least m.*

This theorem says that under the given hypotheses, the easiest thing to do to verify that a point $x \in C$ is in L^+ is to compute $l_1(x), \cdots, l_m(x)$ and see if they are all nonnegative.

3. Spanning trees.

Prim's [1] well-known procedure finds the minimum spanning tree in an undirected graph. There are two types of comparisons employed. The first type finds the closest unconnected node to the set of nodes already connected. This closest node becomes a connected node. The second type compares for each unconnected node the distance to it via the last connected node and the distance to it which was minimal before the last node was connected. If the algorithm is properly programmed by introducing a tree of depth $\lceil \log_2 (n - k) \rceil$ for the arcs from the kth node brought in, then it will take between $\frac{1}{2}(n - 1) \cdot (n - 2)$ and $(n - 1) \cdot (n - 2)$ comparisons, depending upon the number of new arcs brought into consideration in the second type of comparison. We show that any analytic tree program will have depth at least $\frac{1}{2}(n - 1) \cdot (n - 2)$ for this problem. In fact, more strongly, we have the next theorem.

THEOREM 3.1. *Let T be an analytic tree program which, given a complete undirected weighted graph and $n - 1$ arcs, determines whether or not these arcs form a minimum spanning tree. Then $d(T) \geqq \frac{1}{2}(n - 1) \cdot (n - 2)$.*

Proof. Let D be the set of $n - 1$ arcs. Let $d = \max \{d_{ij}$: the arc from node i to node j is in $D\}$. Then the arcs of D form a minimum spanning tree if and only if they form a tree and $d_{ij} \geqq d$ for each i and j such that the arc from i to j is not in D. But this is a set of $\frac{1}{2}(n - 1) \cdot (n - 2)$ inequalities which satisfy Rabin's hypothesis. Q.E.D.

We now discuss updating minimum spanning tree solutions when graph parameters are changed. First we consider adding a new node to the graph.

THEOREM 3.2. *Let an n-node weighted undirected graph G be given, together with $n - 1$ arcs known to be a minimum spanning tree. Let an $(n + 1)$-st node be added to G, together with at least two arcs connecting it to the original n nodes. Then any analytic tree program to compute the minimum spanning tree of the new graph has depth at least n.*

Proof. Consider the case in which there are two arcs from the new node which, together with the given minimum spanning tree, form a cycle of length $n + 1$.

Then the new minimum spanning tree will contain each of these arcs except that arc with the maximum weight. Q.E.D.

The reader can easily construct an $O(n \log n)$ algorithm to update the minimum spanning tree if he or she notes the fact that the only eligible arcs are the $n - 1$ arcs now in the tree and the at most n new arcs connected to the new node. In fact, there is an $O(n)$ algorithm which we now present. Also, the algorithm uses storage proportional to n.

THEOREM 3.3. *There is an algorithm to update the minimum spanning tree of an n-node graph to which a new node has been added which uses $O(n)$ comparisons and $O(n)$ storage.*

Proof. We give the algorithm. The input to the algorithm is the set of arcs in the old tree and the set of arcs to the new node. All arcs appear with their weights.

ALGORITHM.

1. Find minimum weight arc incident upon each node.
2. Find the connected components of the set of arcs found in step 1.
3. Find the minimum arc between each pair of trees found in step 2 such that there is at least one such arc.
4. Collapse each tree found in step 2 to a new node, and go to step 1 if there is more than one such node.

Step 1 requires at most $4n$ comparisons. Step 2 is linear in n if we use Tarjan's [4] connected components algorithm. Step 3 can be done by processing each edge not found in step 1 once and uses linear storage. To see this, note that there can be no more than one arc between any two trees unless one of them contains the newly added node, or there would have been a cycle in the original spanning tree. So we only need to process arcs that go to the component containing the new node and hence use linear storage. In the process we will throw out all nonminimal connecting arcs, so that step 4 is trivial. When we return to step 1, we have the original problem on at most half as many nodes. Hence for a constant c, we have a recursion for the work, $F(n)$, given by

$$F(n) \leqq F(n/2) + cn,$$

so that $F(n) \leqq 2cn$. Q.E.D.

We note that Johnson and Simon [5] have independently discovered an entirely different $O(n)$ algorithm for this problem.

The rest of the results on updating spanning trees are now stated as Theorem 3.4.

THEOREM 3.4. *Let G be an n-node undirected weighted graph whose minimum spanning tree is specified. Then*:

(i) If the value of a tree arc is increased any analytic tree program to update the minimum spanning tree has depth at least $n/4$ for n even and $(n^2 - 4)/4$ for n odd. Furthermore there is an algorithm using this many comparisons in the worst case.

(ii) If the value of a nontree arc is decreased in weight, then an algorithm using $n - 1$ comparisons in the worst case will yield the new minimum spanning tree and no analytic tree program with depth less than $n - 1$ can solve this problem.

(iii) If a node is deleted from the graph together with all of its arcs, then an analytic tree program to update the solution has depth at least $\frac{1}{2}(n-2)\cdot(n-3)$ (although it will usually be easier than this).

Proof. (i) Consider all arcs running between the two subtrees formed by deleting the arc whose weight has increased. Then the new tree will be the union of the subtrees and the connecting arc of minimum weight. If the subtrees have i and $n-i$ nodes, there are $n(n-i)$ such arcs. Hence the result follows.

(ii) The arc of decreased weight is in the new tree if and only if it is no longer the maximum weight arc in the cycle it forms when added to the old minimum spanning tree.

(iii) The worst case occurs when the deleted node was a root of degree $n-1$ of the old tree. Then no old information is useful. Q.E.D.

4. Shortest paths. In this section we discuss finding and updating shortest paths from a single origin in positively weighted directed graphs (digraphs). Dijkstra's [2] procedure for finding a shortest path from a root to every other node in an n-node graph requires between $\frac{1}{2}(n-1)\cdot(n-2)$ and $(n-1)\cdot(n-2)$ comparisons. Similar considerations apply as in the spanning tree problem. Also, similarly to Theorem 3.1, we have Theorem 4.1.

THEOREM 4.1. *Let T be an analytic tree program which verifies that a tree rooted at node* 1 *specifies a shortest path from node* 1 *to each other node in a positively weighted digraph. Then $d(T) \geqq \frac{1}{2}(n-1)\cdot(n-2)$.*

Proof. Let D_{ij} be the shortest distance from node 1 to node j in the given tree for each $1 < j \leqq n$. Assume with no loss of generality that $D_{12} \leqq D_{13} \leqq \cdots \leqq D_{in}$. Then for each $1 \leqq i \leqq j \leqq n$ such that d_{ij} is not in the proposed shortest path tree, we must verify that $d_{ij} \geqq D_{ij} - D_{ij}$ and this set of $\frac{1}{2}(n-1)\cdot(n-2)$ inequalities cannot be proven by an analytic proof of width less than $\frac{1}{2}(n-1)\cdot(n-2)$. Q.E.D.

In contrast to the case of spanning trees, when a new node is added, it requires an $O(n^2)$ algorithm to update the solution. In fact, the updating problems we considered for shortest paths all require $O(n^2)$ steps.

THEOREM 4.2. *Let G be an n-node positively weighted digraph for which a shortest path tree from node* 1 *to each other node is specified. Then*:

(i) *If a new node is added, any analytic tree program to update the solution will have depth at least $\frac{1}{2}(n-1)\cdot(n-2)$.*

(ii) *If a node is deleted, any analytic tree program for updating the set of paths will have depth at least $\frac{1}{2}(n-2)\cdot(n-3)$.*

(iii) *If the weight of some arc in a path is increased, any updating program will have depth at least $\frac{1}{2}(n-2)\cdot(n-3)$.*

(iv) *If the weight of some arc in a path is decreased, the minimum depth of an updating program is at least $\frac{1}{2}(n-2)\cdot(n-3)$.*

(v) *If the weight of an arc not in the shortest path tree is decreased, then any analytic tree program to update the solution has depth at least $\frac{1}{2}(n-2)\cdot(n-3)$.*

Proof. (i) Consider the case in which

$$d_{ij} = 1, \qquad 1 \leqq j \leqq n,$$

$$d_{ij} < \frac{1}{n}, \quad \text{all other } i \text{ and } j \text{ with } 1 \leqq i \neq j \leqq n+1,$$

$$d_{i,n+1} = \min\{d_{ij} : 1 \leqq i \neq j \leqq n+1\}.$$

Then the old tree had a direct arc from node 1 to each other node, but the new tree will not use any of these arcs. The new solution will have a direct path only from node 1 to node $n + 1$, and an entirely new solution for the rest of G which will entail finding a shortest path from node $n + 1$ to each other node.

(ii) Consider the case

$$\begin{aligned} &d_{12} = 1, \\ &d_{2j} = 1, \qquad 3 \leqq j \leqq n, \\ &d_{ij} > 2, \quad \text{all other } i \text{ and } j. \end{aligned}$$

Then if node 2 is deleted, an entirely new problem must be solved on nodes $1, 3, \cdots, n$.

(iii) Take

$$\begin{aligned} &d_{12} = 1, \\ &d_{2j} = 1, \qquad 3 \leqq j \leqq n, \\ &d_{ij} > 2, \qquad i \neq 2, \quad j \neq 2, \\ &d_{i2} > \sum_{j \neq 2} d_{ij}. \end{aligned}$$

Then the original solution is to go from node 1 to node 2 and thence directly to each other node. Now let d_{12} increase to be the maximum of all weights, and we must solve a new problem from node 1 to nodes 3 through n.

(iv) Let

$$\begin{aligned} &d_{ij} = 1, \qquad 1 < j \leqq n, \\ &d_{ij} < \frac{1}{n}, \quad \text{all other } i \text{ and } j, \end{aligned}$$

and now let d_{12} decrease to be the minimum weight arc. So we must solve a shortest path problem from node 2 to each other node.

(v) Let

$$\begin{aligned} &d_{12} = 1, \\ &d_{2j} = 1, \qquad 2 < j, \\ &d_{ij} > 2, \qquad j \neq 2, \\ &d_{ij} < \frac{1}{n}, \quad \text{all other } i \text{ and } j. \end{aligned}$$

Now let d_{13} decrease to $1/n$. Then we must solve a new problem from node 3 to nodes $2, 4, \cdots, n$. Q.E.D.

5. Further considerations. In this concluding section we make several further remarks about shortest paths and spanning trees. Firstly, there is an algorithm for shortest paths or for the spanning tree problem which uses an average of $\frac{1}{2}n^2 + O(n \log^2 n)$ comparisons. To see this, let G be a graph in which the weights are chosen independently from any probability distribution which has zero probability of yielding negative values. Then Spira's [6] algorithm for the all shortest path problem can be adapted to either of the above problems to yield an algorithm which uses $\frac{1}{2}n^2 + O(n \log^2 n)$ comparisons on the average. Secondly, we have discussed updating where only the answer to the problem considered is retained. It seems likely that if intermediate information in obtaining the original solution is kept, improvements will be possible. We have not investigated this. Thirdly, we have not considered sparse graphs. A major open problem is whether there are $O(E)$ algorithms for these computations in the case where E, the number of edges actually present, is small.

Acknowledgment. We acknowledge a helpful discussion with Professor Shimon Even concerning Theorem 3.3.

REFERENCES

[1] R. C. PRIM, *Shortest interconnection network and some generalizations*, Bell System Tech. J., 36 (1967), pp. 1389–1401.

[2] E. W. DIJKSTRA, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[3] M. O. RABIN, *Proving simultaneous positivity of linear forms*, J. Comput. System Sci., 6 (1972), pp. 639–650.

[4] R. TARJAN, *Depth-first and linear graph algorithms*, this Journal, 1 (1972), pp. 146–160.

[5] R. JOHNSON AND JANOS SIMON, Private communication.

[6] P. M. SPIRA, *A new algorithm for finding all shortest paths in a graph of positive arcs in average time $O(n^2 \log^2 n)$*, this Journal, 2 (1973), pp. 28–32.