A Note on Finding Optimum Branchings

P. M. Camerini
L. Fratta
F. Maffioli
Centro Studi per le Telecomunicazioni Spaziali of CNR
and
Istituto di Elettrotecnica ed Elettronica,
Politechnico di Milano,
Milano, Italy

ABSTRACT

The subject of this note is Tarjan's algorithm for finding an optimum branching in a directed graph. Two errors are pointed out, namely (i) an incorrect claim involving branching uniqueness, and (ii) an imprecise way of updating edge values in each iteration. These two inaccuracies do not affect the basic validity of the algorithm. It is shown here that they may be fixed via a simple modification, which leaves unchanged the overall time and space performances.

In the paper on "Finding Optimum Branchings" by R.E. Tarjan [8], an efficient implementation of the algorithm [2,5,6] for computing an optimum branching in a directed graph G = (V,E) is described. Algorithm BRANCH of [8] constructs a subgraph G(H) = (V,H) of G. When the algorithm is completed, it is claimed that an optimum branching of G can be extracted from H via a depth first search. This claim is based upon lemma 2, which states that there is always a unique simple path in G(H) leading from any vertex v of any root component S of G(H) to any vertex w of the weakly connected component W containing S. However, lemma 2 is incorrect, as shown by the following counterexample.

Let us apply algorithm BRANCH to the graph G in fig. 1(a), where numbers on edges are values. Fig. 1(b), (c) and (d) represent a possible sequence of graphs G(H) obtained after each execution of step G8. If we take S = {1,2,3,4} = W, v=1 and w=3 in G(H) of fig. 1(d), there are *two* simple paths (of different values) from v to w. Hence, after deleting edge (4,1) from G(H), a depth first search on the resulting graph cannot guarantee to extract from it an optimum branching of G. Nevertheless, since the results of [6,7] imply that the final set H does contain an optimum branching of G, the method for finding G(H) is valid, but the incorrectness of lemma 2 requires that we update and memorize, after each update of H, the nested shrinking structure of G(H). This allows us to perform a backward expansion of the root components, thus correctly recovering an optimum branching. Many bookkeeping mechanisms could be utilized to this purpose. One of them, having the desirable property of not affecting the complexity evaluations of [8], is here outlined and discussed in detail in [3].

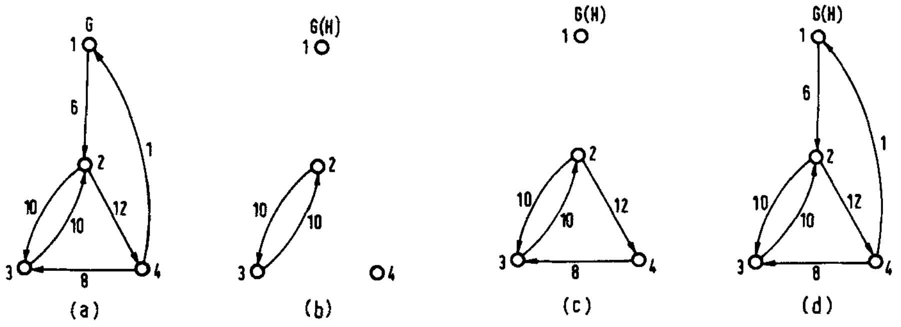

Fig. 1

At the beginning of BRANCH, a forest F is initialized to be an empty forest, with no *nodes* and no *arcs*. Each time an edge (u,v) is added to H in BRANCH, a new node is also added to F, so that the nodes of F are maintained to represent the edges in H. The arcs of F are constructed as follows. Whenever (u,v) is chosen to enter a root component S containing more than one vertex, let (x_1,y_1), (x_2,y_2),...,(x_k,y_k) be the sequence of edges determined in BRANCH, step G5, such that edge (x_k,y_k) was added to H to form S. Then, for each i, $1 \leq i \leq k$, an arc is added to F, directed from node (u,v) to node (x_i,y_i), so that (x_i,y_i) is a *child* of (u,v) in F and (u,v) is the *parent* of (x_i,y_i). In case S contains a unique vertex, a pointer $\lambda(v)$ = (u,v) is set from v to the *leaf* (u,v). During the construction of F we also need to keep track of the set N of all *root nodes* of F, i.e. the nodes which have no parent.

Fig. 2 illustrates the final forest F obtained in the example of fig. 1. Note that, since H does not contain more than 2n-2 edges (n=|V|) [8], the total number of nodes and arcs of F is O(n). Therefore the computation of F does not increase the complexity of BRANCH.

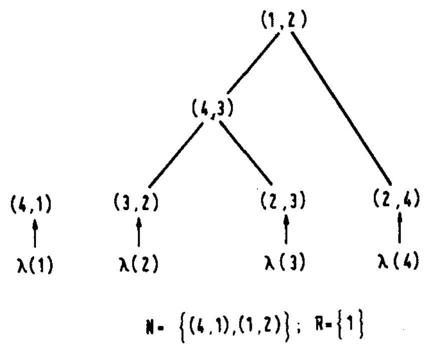

Fig. 2

Once the algorithm is completed, an optimum branching of G can be obtained from F in the following way. Let R be the set of root vertices which algorithm ROOT would select if applied to all root components of G(H). (R = $\{min(i) \mid i \in rset\}$: see section 3 of [8].) Initialize a void set B, initialize N to the set of root nodes of F and repeat steps L1-L3 below until R=N=∅.

Algorithm LEAF:

L1: If R ≠ ∅, delete a root vertex v from R, else pick any root node (w,v) ∈ N and add it to B.

L2: Identify the (possibly trivial) path P in F leading from a root node to the leaf (u,v) = λ(v).

L3: Delete from F all nodes of P and all arcs directed out of these nodes (this step updates the set N of the root nodes).

The identification of step L2 can be easily made by tracing P in the child-to-parent direction, until a root node is found.

As it can be seen in the example of fig. 1 and 2, the final set B obtained by algorithm LEAF is {(1,2),(2,3),(2,4)} which identifies an optimum branching of G and shows, for this example, the correctness of the above algorithm. Its formal proof is obtained by induction and can be found in [3] for the case of spanning arborescences. Since algorithm LEAF consists essentially of visiting each node of F exactly once, its complexity is O(n). Hence all the complexity results given in [8] are still valid.

For what concerns the version of algorithm BRANCH described in section 3, it has to be pointed out that c(x,y) and c(i,j) should refer to values updated as in step G7, rather than the original values of the given digraph. This further difficulty can be overcome by noting [9] that the only values which are used by the algorithm are those of edges either about to be added to H, or already in H but not in a strong component. An edge is added to H only when it is returned by a call on MAX. By using an idea of [1], MAX can be easily modified to return the updated value of the edge, as well as the edge itself. (See [4] for a discussion of how to implement MAX, ADD and QUNION.) Once an edge is added to H, further updates do not affect its value.(Step G7 only updates unexamined edges.)

REFERENCES

1. Aho, A.V., J.E. Hopcroft and J.D. Ullman, "On Finding Optimum Ancestors in Trees", *SIAM J. Comput.*, 5, 1976, pp. 115-132.

2. Bock, F., "An Algorithm to Construct a Minimum Directed Spanning Tree in a Directed Network", *Developments in Operations Research,* Gordon and Breach, New York, 1971, pp. 29-44.

3. Camerini, P.M., L. Fratta and F. Maffioli, "The K Best Spanning Arborescences of a Network",to be published.

4. Cheriton, D. and R. Tarjan, "Finding Minimum Spanning Trees", *SIAM J. Comput.* 5, 1976, pp. 724-742.

5. Chu, Y.J. and T.H. Lin, "On the Shortest Arborescence of a Directed Graph", *Sci. Sinica,* 14, 1965, pp. 1396-1400.

6. Edmonds, J., "Optimum Branchings", *Jour. of Research of The National Bureau of Standards,* 71B, 1967, pp. 233-240.

7. Karp, R.M., "A Simple Derivation of Edmonds' Algorithm for Optimum Branchings", *Networks,* 1, 1971, pp. 265-272.

8. Tarjan, R.E., "Finding Optimum Branchings", *Networks,* 7, 1977, pp. 25-35.

9. Tarjan, R.E., private communication.