Yugoslav Journal of Operations Research
12 (2002), Number 1, 1-10

FINDING MINIMAL BRANCHINGS WITH A GIVEN NUMBER OF ARCS

Dragoš CVETKOVIĆ

Faculty of Electrical Engineering
University of Belgrade, Belgrade, Yugoslavia

Mirjana ČANGALOVIĆ

Faculty of Organizational Sciences
University of Belgrade, Belgrade, Yugoslavia

Abstract: We describe an algorithm for finding a minimal s-branching (where s is a given number of its arcs) in a weighted digraph with an asymetric weight matrix. The algorithm uses the basic principles of the method (previously developed by J. Edmonds) for determining a minimal branching in the case when the number of its arcs is not specified in advance. Here we give a proof of the correctness for the described algorithm.

Key words: Combinatorial optimization, weighted graphs, minimal branching.

1. INTRODUCTION

We consider an arc weighted digraph G without loops and with an asymmetric weight matrix. As usual, the *weight* $d(H)$ of any subgraph H of G is defined to be the sum of weights of arcs of H.

We start with some specific definitions.

Definition 1. *A tree in which each edge is directed (so that it becomes an arc) is called a* **directed tree**.

Definition 2. *A directed tree is called an* **arborescence** *if the following conditions are fulfilled:*

1. *There is a node x with no entering arcs;*
2. *Exactly one arc enters each of the other nodes.*

Node x is called the **root**.

Definition 3. *A digraph whose weakly connected components are arborescences is called a* **branching**. *A branching with s arcs is called an s-***branching**.

We consider the problem of finding a minimal s-branching in G, i.e. a branching with minimal weight. An algorithm for this problem has been developed in [6] (see also [8], pp. 59-61) and we describe it here. The algorithm uses the basic principles of a method for determining a minimal branching (without a specification of the number of arcs in it) in a weighted digraph usually accredited to Edmonds [10], [13], [15], [17] although it was previously discovered in [4] and later independently in [2]. (Some efficient implementations of this method are given in [3], [12], [16]). Edmonds' algorithm cannot of course be directly applied to the problem of determining a minimal s-branching, but it was a starting point in developing our algorithm. In fact, our algorithm is identical to the one given by Edmonds, except that it stops when the number of the chosen arcs is equal to s. This is true in any greedy algorithm when a limit is put on the number of elements. However, Edmond's algorithm is not greedy since it may change the arcs selected in previous steps. Here we give a proof of the correctness for our algorithm. The proof cannot be immediately derived from the existing correctness proofs of Edmond's algorithm (see [10], [13], [15]), as the fixed number of arcs requires some delicate additional considerations.

An incomplete version of our paper has been presented in [7].

The motivation for considering the problem of finding a minimal s-branching is related to a kind of the traveling salesman problem (TSP). Consider the m traveling salesmen problem (m-TSP) with mk cities and with an asymmetric weight matrix where each of the m salesmen should visit the given number of k cities, while their tours should be disjoint. The problem of finding a minimal $m(k-1)$-branching can be used as a relaxation in a branch-and-bound procedure for solving m-TSP [8]. Note that the above version of the m-TSP cannot be solved by a standard transformation [1] which reduces the standard m-TSP to the ordinary TSP. Our m-TSP is similar to the so-called *clover leaf problem* [9] which has not been much studied in the literature.

2. THE ALGORITHM

The algorithm for finding a minimal s-branching is based on a *modification* M (defined in [10]) which transforms the weighted digraph G into a weighted digraph G' in the following way:

Let C be a cycle in G. (By cycle we always mean a directed cycle). An arc which does not belong to C, but enters (leaves) a node of C, is called the *entering* (*leaving*) arc of C. The digraph G' is obtained from G by contracting all nodes of C into a node v, called a *supernode*. In G' all arcs of G not incident to a node of C are kept, all arcs from C are removed, while for all entering and leaving arcs of C, their endnodes belonging to C are replaced by v.

For each arc (x,v) in G' a new weight is determined as

$$p_{C,y}(x,v) = d(x,y) + d(p,q) - d(z,y) \tag{1}$$

where (x,y) is the corresponding entering arc of C, (p,q) an arc in C having the maximal weight, (z,y) the arc of C which enters the same node as (x,y), while $d(x,y), d(p,q), d(z,y)$ are their weights in G, respectively (see Figure 1). Weights of all the other arcs from G' are the same as in G. It is obvious that G' could have multiple arcs, i.e. it could be a multi-digraph. Therefore, the expression *digraph*, used in all further considerations, includes the possibility that multiple arcs exist.

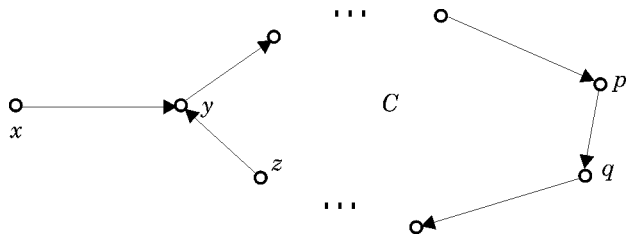

Figure 1.

The algorithm uses a concept of minimal arcs. For each internal node v of G (i.e. a node with an entering arc in G) *the minimal arc* of v is one of its entering arcs with the minimal weight.

Remark 1. If v has more than one entering arc with the same minimal weight, the minimal arc of v can be any of them.

Remark 2. As the minimal arc is unique for each internal node of G, then obviously an arbitrary set of minimal arcs, which do not form any cycle, induces a branching in G.

Now a short description of the algorithm for finding a minimal s-branching is given:

The algorithm

Initialization $i = 0,\ s_0 = 0,\ A_0 = \varnothing,\ G_0 = G$

Phase 1 (forward phase)

Step 1. Determine the set R_i of all minimal arcs from G_i which do not belong to A_i. If $|R_i| < s - s_i$, then STOP: there is no s-branching in G.

Step 2. Order all minimal arcs from R_i according to their nondecreasing weights.

Step 3. Choose minimal arcs from R_i, one by one, respecting their ordering, until either

– the first $s - s_i$ chosen arcs do not form, mutually or with arcs from A_i, any cycle in G_i: These arcs and arcs from A_i determine a branching T_i; go to Phase 2.

or

– the last chosen arc forms, with the previously chosen arcs from R_i or with arcs of A_i, a cycle C_i in G_i.

Step 4. Transform G_i into a digraph G_{i+1} by modification (M), contracting all nodes of C_i into a supernode v_i. Determine A_{i+1} as the set of all minimal arcs, chosen in Step 3, which belong to G_{i+1}. Update s_{i+1} as

$$s_{i+1} = |A_{i+1}| + \sum_{j=0}^{i} (m_j - 1)$$

where m_j is the total number of nodes in the cycle C_j. Set $i = i + 1$ and go to Step 1.

Phase 2 (backward phase).

Step 5. If $i = 0$, then STOP: $T \equiv T_0$ is a minimal s-branching in G.

Step 6. A branching T_{i-1} in the digraph G_{i-1} is formed as follows: T_{i-1} contains all arcs from T_i not incident to v_{i-1} and also arcs corresponding to the arcs of T_i leaving v_{i-1}.

– If T_i includes the minimal arc of v_{i-1}, then T_{i-1} contains the corresponding entering arc (x, y) of C_{i-1} and all arcs of C_{i-1}, except (z, y) entering the same node as (x, y).

– If T_i does not include the minimal arc of v_{i-1}, then T_{i-1} contains all arcs of C_{i-1}, except an arc of the maximal weight. Set $i = i - 1$ and go to Step 5.

Remark 3. In the case when the weights of some minimal arcs from G_i are mutually equal, the set R_i and the ordering of their elements (Steps 1 and 2 of Phase 1) need not be determined in a unique way.

Remark 4. It can be easily estimated that the numerical complexity of the algorithm is $O(n^2)$, where n is the number of nodes of the digraph. In fact, the most time consuming parts of the algorithm are the determination of all minimal arcs (which obviously has complexity $O(n^2)$) and their sorting (with complexity no more than $O(n^2)$).

Remark 5. The problem of finding a minimal s-branching becomes NP-hard when the resulting branching is required to be connected (i.e. to consist of a simple arborescence spanning $s + 1$ out of n nodes in the digraph). This follows from the NP-hardness of the k-cardinality tree problem proved in [11].

3. THE ALGORITHM CORRECTNESS PROOF

It is well-known that the problem of finding a minimal branching can be formulated as the weighted matroid intersection problem [14], [5]. Namely, branchings are common independent sets for the forest matroid of G and for the matroid of subgraphs of G having indegrees at most 1. (However, branchings for themselves do not constitute a matroid). The correctness of our algorithm follows from the correctness of the weighted matroid intersection algorithm, in particular, from a theorem (Theorem 9.1 of [14], i.e. Theorem 8.24 of [5]) justifying a procedure for extending a maximum weight common independent set of cardinality k to the one of cardinality $k+1$. However, our algorithm avoids some steps present in the general algorithm and has a lower complexity ($O(n^2)$ instead of $O(n^4)$ in general case). In addition we offer an elementary correctness proof using graph theoretical terminology, thus avoiding more general structures of the matroid theory.

According to the algorithm definition, it is obvious that the branching T, if it is obtained in Step 5 of Phase 2, has s arcs (if there is no s-branching in G, the algorithm stops at Step 1 of Phase 1). We show there that T is a minimal s-branching in G. Our proof is an elaboration of the one outlined in [6].

First, several necessary lemmas should be proved.

Lemma 1. *Let minimal arcs of all internal nodes in a weighted graph G be ordered according to nondecreasing weights. If the first k minimal arcs do not form any cycle in G, then they induce a minimal k-branching of G.*

The proof is straightforward and thus omitted.

Let $G_0 \equiv G, G_1, ..., G_q$, $q \geq 0$ be digraphs, considered in Phase 1, and $T_0, T_1, ..., T_q$ be the corresponding branchings, generated in Phase 2 of the algorithm.

For $q = 0$ it follows, directly from Lemma 1, that $T \equiv T_0$ is a minimal s-branching in G. Therefore we shall further suppose that $q \geq 1$.

Let H_i be a subgraph of G_i, $i \in \{0,1,...,q-1\}$, induced by all arcs entering nodes of the cycle C_i (formed in Step 3 of Phase 1). For each (x,y) from H_i, which is an entering arc of C_i, $p_i(x,y)$ denotes a value, defined by (1) in the modification (M), i.e.

$$p_i(x,y) = d(x,y) + d(u,w) - d(z,y),$$

where (u,w) is an arc of C_i with the maximal weight and (z,y) belongs to C_i and enters the same node as (x,y). Let G_i^* be a subgraph of G_i, induced by all arcs of G_i which do not belong to H_i.

Now the following lemmas can be proved:

Lemma 2. *Each arc of* G_i^* *not entering any internal node of* T_i*, has weight not smaller that the weight of any arc in* C_i.

Proof: Each arc of G_i^*, not entering any internal node of T_i, enters either a node which does not belong to T_i or a root of T_i not belonging to C_i. It is sufficient to prove that for each such node y weight $d(x,y)$ of its minimal arc (x,y) is not smaller than weights of arcs in C_i.

For $i=q-1$, weight $d(x,y)$ cannot obviously be smaller than weights of arcs in C_i. (Otherwise, arc (x,y) would be chosen in Step 3 of Phase 1 before closing the cycle C_i and, as (x,y) would not be included into any supernode during Phase 1, it would belong to T_i, i.e. node y would represent an internal node of T_i).

Let us consider the case when $i<q-1$ for $q>1$ and let us assume that $d(x,y)$ be smaller than a weight of at least one arc in C_i. Then arc (x,y) would be chosen in Step 3 of Phase 1 before closing cycle C_i. As (x,y) does not belong to T_i, this arc would be included to a cycle C_j, $j\in\{i+1,\ldots,q-1\}$ of a digraph G_j and then excluded from T_j in Step 6 of Phase 2 such that either

(a) T_j contains the entering arc of C_j corresponding to node y and all arcs from C_j except (x,y); or

(b) T_j does not contain any entering arc of cycle C_j, but contains all its arcs except arc (x,y) of the maximal weight.

Consequently, in G_i there would be a path P from node y to node x belonging to T_i and consisting of arcs which either belong to C_j or they are contracted into supernodes of C_j.

Now we shall derive the following contradictions:

If case (a) occurred, node y would obviously be an internal node of T_i, which leads to contradiction.

The weight of the arc (x,y) in G_j would be the same as in G_i and therefore in case (b) $d(x,y)$ would not be smaller than weights of all the other arcs from C_j. Moreover, if C_j contained a supernode (formed after closing C_i and before closing C_j), $d(x,y)$ would not be smaller than weights of all arcs from G_i contracted into this supernode. (It follows from the fact that, always when a new supernode is obtained by contracting a cycle, new weights of its entering arcs are not smaller than weights of the arcs in this cycle).

Consequently, weights of arcs in path P would not be greater than $d(x,y)$. It means that there would be a cycle in G_i (composed of (x,y) and P) which would be closed in Step 3 of Phase 1 before C_i, which leads to contradiction. ♦

Lemma 3. *If T_{i+1} is a minimal branching in G_{i+1}, then T_i is a minimal branching in G_i, $i \in \{0,1,...,q-1\}$, where minimal branchings are considered with respect to the number of arcs in T_{i+1} and T_i, respectively.*

Proof: Let m_i be the number of nodes in the cycle C_i of G_i. If T_{i+1} has r arcs, then T_i has $m_i + r - 1$ arcs. Let us suppose that T_i is not a minimal $(m_i + r - 1)$-branching, but that there exists a minimal $(m_i + r - 1)$-branching U_i in G_i such that $d(U_i) < d(T_i)$. We denote by T_i' and U_i' parts of branchings T_i and U_i, respectively, on the subgraph H_i, and with T_i'' and U_i'' parts of these branchings on the subgraph G_i^* of G_i.

Two cases need to be considered.

The first case. T_i' has m_i arcs, i.e. T_i'' has $r-1$ arcs. It is obvious that

$$d(T_i') = d(C_i) + d(x,y) - d(z,y), \tag{2}$$

where (x,y) is the entering arc of C_i with minimal $p_i(x,y)$ and (z,y) the corresponding arc in C_i.

According to the modification (M), G_i^* corresponding to a subgraph of G_{i+1}, induced by all arcs not entering the supernode formed by contracting C_i. Therefore, we have

$$d(T_{i+1}) = d(T_i'') + p_i(x,y).$$

a) U_i' has m_i arcs, i.e. U_i'' has $r-1$ arcs.

It is obvious that the internal nodes of U_i' must belong to C_i. As U_i' is a part of the minimal branching U_i, and C_i contains only minimal arcs, then U_i' consists of $m_i - 1$ arcs of C_i and an entering arc (x_1, y_1) of C_i. Therefore

$$d(U_i') = d(C_i) + d(x_1,y_1) - d(z_1,y_1), \tag{3}$$

where (z_1, y_1) is the corresponding arc in C_i which does not belong to U_i'.

As T_{i+1} is minimal r-branching in G_{i+1}, then

$$d(T_{i+1}) = d(T_i'') + p_i(x,y) \le d(U_i'') + p_i(x_1,y_1). \tag{4}$$

From (2)-(4) it follows that $d(U_i) \ge d(T_i)$, which leads to contradiction.

b) U_i' has $m_i - l$ arcs, $1 \le l \le m_i$, i.e. U_i'' has $r - 1 + l$ arcs.

It is obvious that

$$d(U_i') = d(C_i) - \sum_{i=1}^{l} t_i \,, \tag{5}$$

where $t_1,...,t_l$, are the weights of l arcs of C_i having the greatest weights.

G_i^* contains $r-1$ internal nodes of T_i. Therefore, according to Lemma 2, in U_i'' there exist at least l arcs with weights not smaller than weights $t_1,...,t_l$. Among these arcs we choose arbitrary $l-1$ arcs and denote their weights with $t_1', t_2',...,t_{l-1}'$.

Let U_i^* be a part of U_i'' not containing arcs with weights $t_1', t_2',...,t_{l-1}'$, i.e.

$$d(U_i'') = d(U_i^*) + \sum_{i=1}^{l-1} t_i' \,. \tag{6}$$

As U_i^* has r arcs, then

$$d(U_i^*) \geq d(T_{i+1}) = d(T_i'') + p_i(x,y) \,. \tag{7}$$

Now, from (2), (5)-(7) it follows that $d(U_i) \geq d(T_i)$, which leads to contradiction.

The second case. T_i' has $m_i - 1$ arcs, i.e. T_i'' has r arcs. According to Phase 2 of the algorithm,

$$d(T_i') = d(C_i) - d(u,w) \,, \tag{8}$$

where (u,w) is an arc in C_i with the maximal weight, and $d(T_i'') = d(T_{i+1})$.

a) U_i' has $m_i - 1$ arcs, i.e. U_i'' has r arcs. The internal nodes of U_i' must belong to C_i. As U_i' is a part of the minimal branching U_i and C_i consists of minimal arcs, then obviously U_i' contains $m_i - 1$ arcs of C_i. Therefore $d(U_i') \geq d(T_i')$.

As U_i'' has r arcs, then $d(U_i'') \geq d(T_{i+1}) = d(T_i'')$. Consequently, $d(U_i) \geq d(T_i)$, which leads to contradiction.

b) U_i' has $m_i - l$ arcs, $1 < l \leq m_i$, i.e. U_i'' has $r - 1 + l$ arcs.

In the same way as for the first case b), it can be proved that (5), (6) hold. Also,

$$d(U_i^*) \geq d(T_{i+1}) = d(T_i'') \,. \tag{9}$$

From (5), (6), (8), (9) it follows that $d(U_i) \geq d(T_i)$, which leads to contradiction.

c) U_i' has m_i arcs, i.e. U_i'' has $r-1$ arcs.

Now, according to the considerations in the first case a),

$$d(U_i') = d(C_i) + d(x,y) - d(z,y) \,, \tag{10}$$

where (x,y) is an entering arc of C_i and (z,y) the corresponding arc in C_i. Also we have

$$d(U_i'') + p_i(x,y) \geq d(T_{i+1}) = d(T_i'') . \tag{11}$$

From (8), (10), (11) it follows that $d(U_i) \geq d(T_i)$, which leads to contradiction.

Since in all considered cases we come to a contradiction, T_i is a minimal $(m_i + r - 1)$ branching in G_i. ♦

Lemma 4. *The branching T_q in G_q is a minimal branching with respect to the number of arcs in T_q.*

Proof: Let r be the total number of arcs in T_q and T_q^* be the set of all arcs from T_q, chosen in previous steps of Phase 1 (before forming G_q). If $T_q^* = \varnothing$ it follows, directly from Lemma 1, that T_q is a minimal r-branching.

If $T_q^* \neq \varnothing$, we prove that the weight $d(x,v)$ of a minimal arc (x,v) in G_q such that $(x,v) \notin T_q^*$, is not smaller than weights of arcs from T_q^*.

If v is not a supernode formed by contracting the cycle C_{q-1} in G_{q-1}, then $d(x,v)$ cannot be smaller than weights of arcs in T_q^* (Otherwise, (x,v) would be chosen in one of the previous steps of Phase 1, i.e. (x,v) would belong to T_q^*).

If v is a supernode formed by contracting the cycle C_{q-1} in G_{q-1}, then (x,v) corresponds to an entering arc (x,y) of C_{q-1}. Therefore,

$$d(x,v) = p_{q-1}(x,y) \geq d(u,w) ,$$

where (u,w) is an arc of C_{q-1} with the maximal weight, while $d(u,w)$ is its weight in G_{q-1}. As (u,w) is a minimal arc last chosen in G_{q-1} (Step 3 of Phase 1), then $d(u,w)$ is not smaller than weights of arcs from T_q^* and, consequently, the same holds for $d(x,v)$.

According to the above considerations and Lemma 1, all arcs of G_q chosen in Step 3 of Phase 1, together with arcs from T_q^*, represent a minimal r-branching in G_q. ♦

Now the correctness of the algorithm can be proved in the following way:

Theorem. *The branching $T \equiv T_0$, generated by the algorithm, is a minimal s-branching in the digraph G.*

Proof: According to Lemma 4, T_q is a minimal branching of G_q with respect to the corresponding number of arcs. Therefore, from Lemma 3 (iteratively applied to T_i for $i = q-1,\ldots,0$) it follows that T_0 is a minimal s-branching in $G_0 \equiv G$.

REFERENCES

[1] Bellmore, M., and Hong, S., "Transformation of the multisalesman problem to the standard traveling salesman problem", *J. Assoc. Comp. Mach.*, 21 (1974) 501-504.

[2] Bock, F.C., "An algorithm to construct a minimum directed spanning tree in a directed network", in: *Developments in Operations Research*, B. Avi-Itzak (ed.), Gordon and Breach, New York, 1971, 29-44.

[3] Camerini, M.P., Fratta, L., and Maffioli, F., "A note on finding optimum branching", *Networks*, 9 (1979) 309-312.

[4] Chu, Y.I., and Lin, T.H., "On the shortest arborescence of a directed graph", *Scientia Sinica*, 4 (1965) 1396-1400.

[5] Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., and Schrijver, A., *Combinatorial Optimization*, John Wiley & Sons, Inc., New York, 1998.

[6] Cvetković, D., "Finding a shortest rooted forest", unpublished report, Faculty of Electrical Engineering, Belgrade, 1987. (in Serbian)

[7] Cvetković, D., and Čangalović, M., "An algorithm for finding minimal branchings", *Proceedings of XXIV Yugoslav Symposium in Operations Research*, 1997, 183-186.

[8] Cvetković, D., Dimitrijević, V., and Milosavljević, M., *Variations on the Traveling Salesman Theme*, Libra Produkt, Belgrade, 1996.

[9] Dantzing, G.B., and Ramser, J.H., "The truck dispaching problem", *Management Sci*., 6 (1960) 80-91.

[10] Edmonds, J., "Optimum branchings", *J. Res. Nat. Bur. Standards*, B, 71 (1967) 233-240; reprinted in: *Mathematics of Decision Sciences, Lectures in Appl. Math*., G. Dantzing, A., Vienott (eds.), 1968, 346-351.

[11] Fischetti, M., Hamacher, H., Jörnsten, K., and Maffioli, F., "Weighted k-cardinality trees: complexity and polyhedral structure", *Networks*, 24 (1994) 11-21.

[12] Gabow, H.N., Galil, Z., Spencer, T., and Tarjan, R.E., "Efficient algorithms for finding minimum spanning trees in nondirected and directed graphs", *Combinatorica*, 6 (2) (1986) 109-122.

[13] Karp, R.M., "A simple derivation of Edmonds' algorithm for optimum branchings", *Networks*, 1 (1972) 265-272.

[14] Lawler, E.L., "Matroid intersection algorithms", *Math. Programming*, 9 (1975) 31-56.

[15] Minieka, E., *Optimization Algorithms for Networks and Graphs*, Marcel Dekker Inc,. New York-Basel, 1978.

[16] Tarjan, R.E., "Finding optimum branchings", *Networks*, 7 (1977) 25-35.

[17] Thulasiraman, K., and Swamy, M.N.S., *Graphs: Theory and Algorithms*, John Wiley & Sons, New York, 1992.