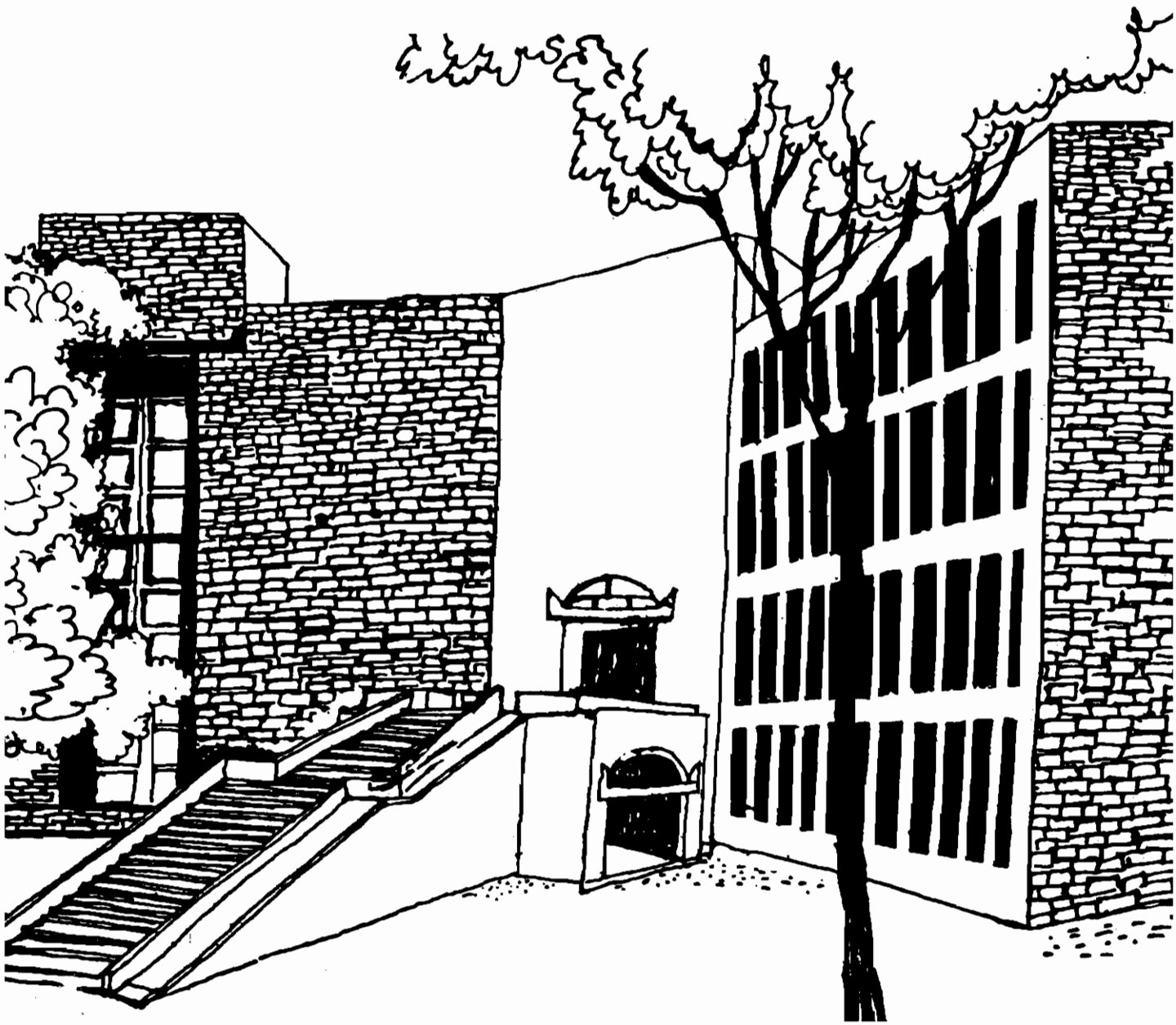
विद्याविनियोगाद्विकासः
IIM
AHMEDABAD

Working Paper



THE MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM: WEIGHTS ON ARCS CASE

By

V. Venkata Rao
&
R. Sridharan

W P No. 1030
May 1992

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

THE MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM: WEIGHTS ON ARCS CASE

V.Venkata Rao and R. Sridharan
Indian Institute of Management
Ahmedabad 380 015

ABSTRACT

In a rooted acyclic graph, G, there exist, in general, several rooted (not necessarily spanning) arborscences. Depending on whether the graph has weights on nodes, on arcs, or on both, it is possible to define, with different objective functions, several different problems, each concerned with finding an optimal rooted arborscence in the graph under consideration. Of the different types of rooted acyclic graphs, we are in particular interested in two: 1. rooted acyclic graph G_n with weights on nodes, and 2. rooted acyclic graph G_a with weights on arcs. In the first category, an optimal rooted arborsence can be defined as one whose sum of node weights is less than or equal to that of any other rooted arborscence in G_n; the problem of finding such an arborscence is called the *minimum rooted arborscence (MRA(G_n)) problem in an acyclic rooted graph with weights on nodes.* Similarly, in the second category, an optimal rooted arborscence can be defined as one whose sum of arc weights is less than or equal to that of any other rooted arborscence in G_a; the corresponding problem is called *the minimum rooted arborscence (MRA(G_a)) problem in a rooted acyclic graph with weights on arcs.*

The MRA(G_n) has already been studied. The objective of this paper is to explore the relation between MRA(G_a) and MRA(G_n) problems, and to propose approximate and exact methods for solving MRA(G_a) problem. However, the paper presents no computational results, as the programming of the proposed algorithms is still in progress.

After discussing the relation between the MRA(G_n) and MRA(G_a) problems, we formulate the MRA(G_a) problem as a zero-one programming problem, and discuss a a heuristic to construct a rooted arborscence RA in any given G_a. This heuristic can be used to generate an upper bound on the value of the objective function for MRA(G_a). We also discuss the formulation of a Lagrangian Dual of MRA(G_a) problem and present a linear relaxation of MRA(G_a). Finally, we present a branch and bound scheme for the MRA(G_a) problem.

THE MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM: WEIGHTS ON ARCS CASE

V.Venkata Rao and R. Sridharan
Indian Institute of Management
Ahmedabad 380 015

1. Introduction

In a rooted acyclic graph, G, there exist, in general, several rooted (not necessarily spanning) arborscences. Depending on whether the graph has weights on nodes, on arcs, or on both, it is possible to define, with different objective functions, several different problems, each concerned with finding an optimal rooted arborscence in the graph under consideration. Of the different types of rooted acyclic graphs, we are in particular interested in two: 1. rooted acyclic graph G_n with weights on nodes, and 2. rooted acyclic graph G_a with weights on arcs. In the first category, an optimal rooted arborsence can be defined as one whose sum of node weights is less than or equal to that of any other rooted arborscence in G_n; the problem of finding such an arborscence is called the *minimum rooted arborscence (MRA(G_n)) problem in an acyclic rooted graph with weights on nodes.* Similarly, in the second category, an optimal rooted arborscence can be defined as one whose sum of arc weights is less than or equal to that of any other rooted arborscence in G_a; the corresponding problem is called *the minimum rooted arborscence (MRA(G_a)) problem in a rooted acyclic graph with weights on arcs.*

The MRA(G_n) problem arises in a certain integer programming model of a multistage production system. Further, it is related to the uncapacitated warehouse location problem. MRA(G_n) was already studied in [Rao, and McGinnis, 84]. The focus of this paper is the second type of problem mentioned above, namely the MRA(G_a) problem. So far as we know, unlike the MRA(G_n) problem, the MRA(G_a) problem has not been identified to arise in any practical context. We have chosen to study this problem mainly because: 1. It bears a strong resemblance to the MRA(G_n) problem, which has, as mentioned above, some practical application, 2. A more general optimal arborescence problem in which the graph has weights on both nodes and arcs can be reduced to the MRA(G_a) problem.

The objective of this paper is to explore the relation between MRA(G_a) and MRA(G_n) problems, and to propose approximate and exact methods for solving MRA(G_a) problem. However, the paper presents no computational results, as the programming of the proposed algorithms is still in progress.

In the next section, section 2, we introduce the basic definitions, notation, and terminology of the paper. Section 3 states the MRA(G_a) problem as a zero-one programming problem. Section 4 discusses the relation between the two problems, MRA(G_a) and MRA(G_n). Section 5 presents a heuristic to construct a rooted arborscence RA in any given G_a. In section 6, we discuss the formulation of a Lagrangian Dual of MRA(G_a) problem; and then in section 7 we present a linear relaxation of MRA(G_a). Section 8 presents a branch and bound scheme for the MRA(G_a) problem. This scheme uses the heuristic of section 5 to generate upper bounds, and the Lagrangian Dual of section 6 to generate lower bounds. Finally, section 9 consists of concluding remarks.

2. Definitions, Terminology and Notation

Rooted acyclic graph, G. A graph G = {V,E} where V is the set of indices and E the set of edges (or arcs) is called a rooted acyclic graph if it posseses the following properties: 1.Each arc of the graph is directed, 2.There are no directed cycles embedded in the graph, 3. The graph is connected. One of the vertices which does not have any incoming arcs, and which has a connected path to every other node in the graph is specified as the root of the graph.

G consists of N nodes. To avoid trivial situations, we assume that $N > 0$ and, further, that G has at least one arc. The nodes are indexed with consecutive integers 1, ..., N in the topologically sorted order; that is, the nodes are indexed such that an arc is always directed from a lower index node to a higher index node. If an arc is directed from node i to node j, i is said to be an immediate predecessor of j, and j an immediate successor of i. As the root does not have any immediate predecessors, it bears the index 1. The set P(j) consists of the indices of the immediate predecessors of j and S(j) the indices of the immediate successors of j. F and L are sets consisting of vertex indices such that $j \in F \rightarrow P(j) = \emptyset$, and $j \in L \rightarrow S(j) = \emptyset$.

Rooted acyclic graph with weights on nodes (G_n). If a rooted acyclic graph has a weight $\alpha(j)$ associated with each of its nodes j, then the graph is called a rooted acyclic graph with weights on nodes. The weight $\alpha(j)$ of a node j can be any real number, negative, positive, or zero.

A subgraph $RA(G_n)$ of G_n is called a *rooted arborscence* of G_n if (1) $RA(G_n)$ contains the root as one of its vertices, (2) $RA(G_n)$ is connected, and (3) no two arcs of $RA(G_n)$ are directed towads the same vertex. The sum of weights of the vertices in $RA(G_n)$, $W[RA(G_n)]$, is called the weight of this arborscence.

Rooted acyclic graph with weights on arcs (G_a). If a rooted acyclic graph is such that: (1) associated with each arc (i,j) in the graph there is a weight W(i,j), where W(i,j) is any real number, positive, negative, or zero, (2) $N > 2$, and node 1 is connected to node 2 by arc (1,2), and it is the only arc emanating from node 1, then the graph is called a rooted acyclic graph with weights on arcs. Arc (1,2) is called the root arc of G_a.

A subgraph $RA(G_a)$ of G_a is called a rooted arborescence of G_a if 1. $RA(G_a)$ contains the root arc as one of its arcs, 2. $RA(G_a)$ is connected, and 3. no two arcs of $RA(G_a)$ are directed towards the same vertex. The sum of weights of the arcs in $RA(G_a)$, $W[RA(G_a)]$, is called the weight of the arborescence.

As mentioned earlier, a rooted arborescence of a graph G is called a minimum weight rooted arborescence MRA(G) of that graph if its weight W[MRA(G)] is less than or equal to that of any other rooted arborescence RA(G) of that graph.

Rooted path. A rooted path for a node i in G is an alternating sequence of nodes and arcs, starting with root node 1 and ending with node i, which can be written as

[1,(1,j), j, (j,k), k,...., (l,i), i].

Sometimes, while using the above notation for the rooted arborescence or minimum rooted arborescence, for the sake of simplicity, we will not mention explicitly the type of graph being referred to; that is whether it is G_a or G_n. In all such cases, it is to be understood that we are referring to a graph with weights on arcs G_a. See Figure-1 for an illustration of MRA(G_n) and MRA(G_a).

3. Mathematical Formulations of MRA(G_n) and MRA(G_a)

Both the problems MRA(G_n) and MRA(G_a) can be stated as zero-one linear programs. The zero-one program for MRA(G_n), already presented in [Rao and McGinnis 84], is:

MRA(G_n).

$$\underset{Y(j)\in\{0,1\}}{Min}\left[\sum_{j=1}^{N}\alpha(j)Y(j)\;\middle|\;Y(j)\le\sum_{i\in P(j)}Y(i),\ j=2,3,\ldots,N,\ Y(1)=1\right] \quad (1)$$

In the above formulation, Y(j)'s are zero-one variables which indicate the presence (Y(j) = 1) or absence (Y(j) = 0) of each node j in the optimal arborescence. The above formulation is concerned only with nodes. This is because the arcs do not have any weights; therefore, once we know which vertices to include in the arborescence, selection of arcs is simple.

Similarly, the MRA(G_a) problem also can be stated as a zero-one program as below:

MRA(G_a).

$$Z=Min\sum_{(i,\,j)\in E}W(i,\,j)Y(i,\,j) \quad (2)$$

$$s.t.\sum_{i\in P(j)}Y(i,j)\le 1,\quad j=2,3,\ldots,N \quad (3)$$

$$Y(i,j)\le\sum_{k\in P(i)}Y(k,i),\ for(i,\,j)\in E\setminus(1,2) \quad (4)$$

$$Y(1,2)=1 \quad (5)$$

$$Y(i,j)\in\{0,1\}\ for\ (i,\,j)\in E,\ (i,\,j)\neq(1,2) \quad (6)$$

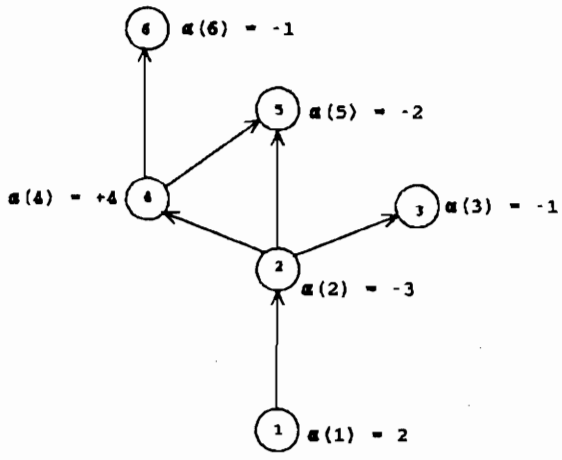


Figure-1a. A rooted acyclic graph with weights on nodes(G_n)

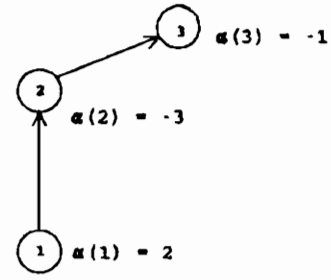


Figure-1b. MRA(G_n) of the G_n in Figure-1a. W(MRA(G_n) = -2

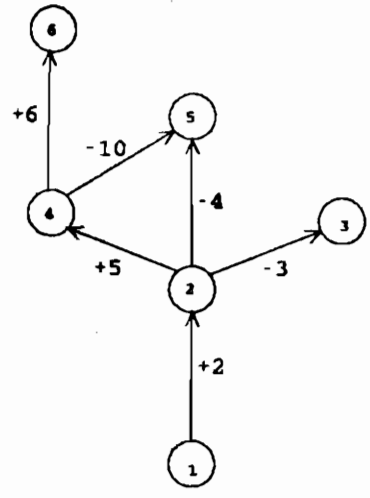


Figure-1c. A rooted acyclic graph with weights on arcs, G_a

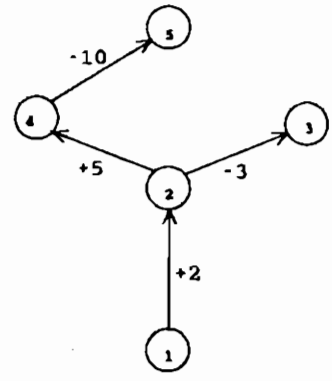


Figure-1d. MRA(G_a) of the G_a in Figure-1c. W(MRA(G_a) = -6

Figure-1. MRA in G_n and G_a : Illustrations

In the MRA(G_a) problem, unlike in MRA(G_n) problem, the zero-one variables are associated with arcs and not with nodes: each Y(i,j) corresponds to an arc (i,j)$\in$ E. In the solution, Y (i,j) = 1 implies that arc (i,j) is present in MRA(G_a); Y(i,j) = 0 implies that arc (i,j) is absent. Constraints (3) ensure that, in the selected subgraph, not more than one arc is directed towards a selected node; constraints (4) ensure that an arc is not selected, unless at least one of its predecessor arcs is also selected. Constraint (5) ensures that the root arc is definitely present in the final solution. In future, we refer to the constraints (3) as *incidence constraints* and (4) as *connectivity constraints*.

4. Relation between MRA(G_a) and MRA(G_n) problems

Algorithms are already available [Rao and McGinnis, 84] to construct an MRA in a given G_n. Suppose there exist rules to transform a given G_a into a corresponding G_n. Further, suppose there exist rules to find for an MRA of this G_n a corresponding RA in G_a; suppose also that the RA(G_a) obtained through these rules is ensured to be an MRA(G_a). Then, obviously, these rules can be used to solve the MRA(G_a) problem as follows: first, for the given G_a, construct the corresponding G_n; then, using the algorithm of [Rao and McGinnis 84] construct an MRA for this G_n; for the MRA(G_n) so constructed, find the corresponding RA(G_a). The RA(G_a) so obtained is an MRA(G_a) that we need.

However, unfortunately, for a general G_a, it turns out that it is difficult (currently impossible) to find rules that transform the G_a into a corresponding G_n.

While a general G_a is difficult to handle through the above approach, the special case when G_a is a tree is amenable to solution, because, in such a case, appropriate transformation rules between G_a and G_n exist. For this special case, let us state these conversion rules below.

First, note that when G_a is a tree, no two arcs in G_a are directed towards the same node.

Given a G_a construct the corresponding G_n as below:

For each node i of G_a, set a corresponding node i′in G_n. For each arc (i,j) of G_a introduce an additional node k′ in G_n between i′ and j′. The node k′ is called an intermediate node. Establish an arc from i′ to k′ and another from k′ to j′. Make the weight of the intermediate node k′ in G_n equal to the weight of its corresponding arc (i,j) in G_a. Set the weights of nodes i′ and j′ to zero. That is, $\alpha(k') = W(i,j)$, $\alpha(i') = \alpha(j') = 0$. In other words, G_n is the same as G_a except that each arc of G_a gets split into two in G_n, the two segments being connected by an intermediate node, which bears the same weight as the weight of its original arc.

Given an MRA in G_n, where G_n is a weights on node graph constructed using the above rules, construct the corresponding RA(G_a) as below:

For each k′$\in$ MRA(G_n), where k′ is an intermediate node obtained using the conversion method mentioned above, select the corresponding arc (i,j) of G_a to be present in RA(G_a).

In the above transformation, it is easy to show that:

1. G_a is a tree implies that G_n is a tree

2. MRA(G_n) is a rooted tree (or arborescence) implies that RA(G_a) is a rooted tree (or arborescence).

3. The RA(G_a) is an MRA(G_a).

All the above three properties can be proved by the method of contradiction.

Figure-2 illustrates the above ideas by showing a G_a, the corresponding G_n, MRA(G_n) and the corresponding MRA(G_a).

Even when G_a is not a tree, the above conversion rules can still be applied; but, in that case, there is no guarantee that the weight of MRA(G_n) will be equal to the weight of MRA(G_a), or that the arcs selected in G_a will form an arborescence. This is because the incidence constraints which are a part of MRA(G_a) problem are not a part of MRA(G_n) problem. Figure-3 illustrates this point through an example. This example shows that MRA(G_a) cannot have both the arcs (2,4) and (3,4) in the solution whereas the corresponding "transformed" MRA(G_n) problem will choose both the corresponding arcs in the solution. However, we see that the "equivalent" MRA(G_n) provides a lower bound for MRA(G_a).

5. A heuristic for the MRA(G_a) problem

A heuristic solution for the MRA(G_a) problem has two uses: first, it can be used as a solution algorithm; second, the weight of the heuristic solution serves as an upperbound for W[MRA(G_a)]. Before presenting a heuristic, let us first introduce a restricted version of the MRA problem, which will be referred to as MRA_θ problem. The restriction is in the form of necessarily requiring some arcs to be present in the solution and some to be absent. To reflect the given restrictions, we define three sets θ_{in}, θ_{out}, and θ_{free}. θ_{in} consists of all the arcs which need to be present in the solution, θ_{out} consists of all those which need to be absent, and θ_{free} all the remaining. Thus MRA_θ problem can be stated as below:

MRA_θ (G_a).

$$Min \sum_{(i,j)\in \theta_{in} \cup \theta_{free}} W(i,j)Y(i,j) \qquad (7)$$

s.t. (3), (4),*and*

$$Y(i,j)=0, \quad (i,j)\in \theta_{out} \qquad (8)$$

$$Y(i,j)=1, \quad (i,j)\in \theta_{in} \qquad (9)$$

$$Y(i,j)\in \{0,1\}, \quad (i,j)\in \theta_{free} \qquad (10)$$

When $\theta_{in} = \{(1,2)\}$, and $\theta_{out} = \emptyset$, the resulting MRA(G_a) problem is the same as the unrestricted MRA(G_a) problem that we have considered in the previous sections. As MRA(G_a) problem is a special case of MRA_θ(G_a) problem, an algorithm for the latter problem can be automatically used for the former.

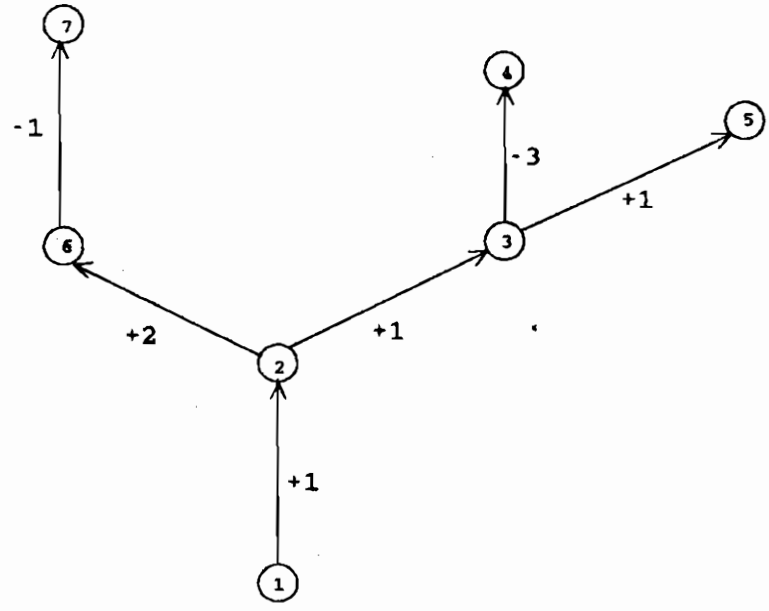


Figure-2a. G_a. MRA(G_a) consists of arcs (1,2),(2,3),(3,4).
W(MRA(G_a) = 1+1-3 = -2

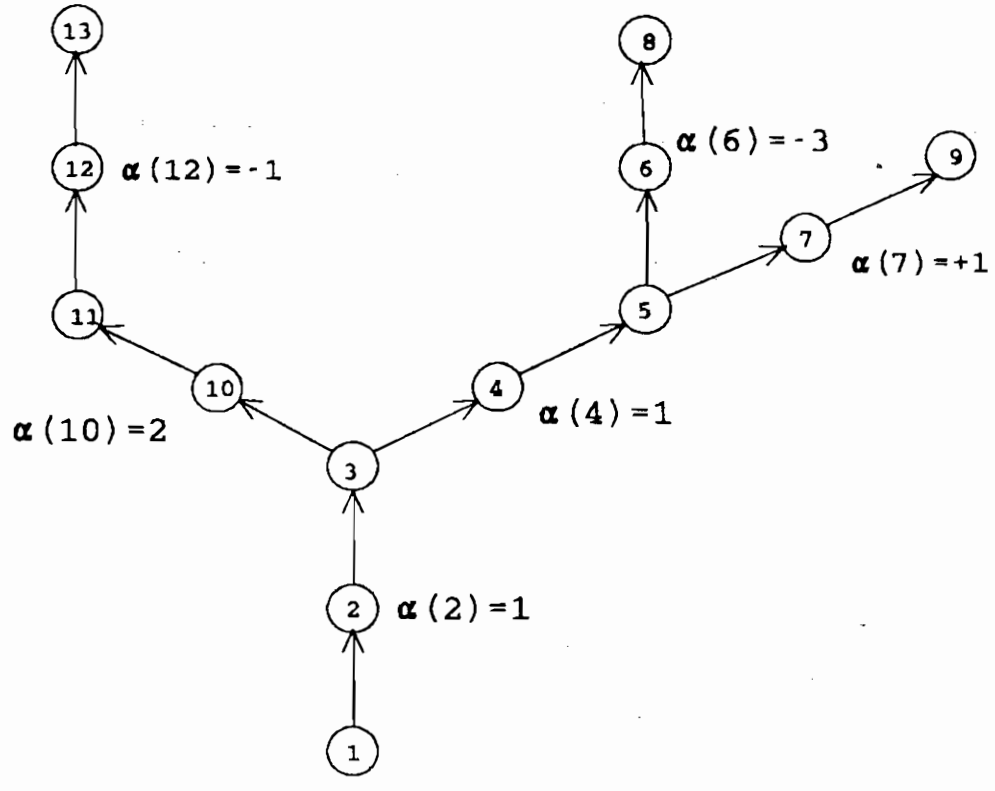
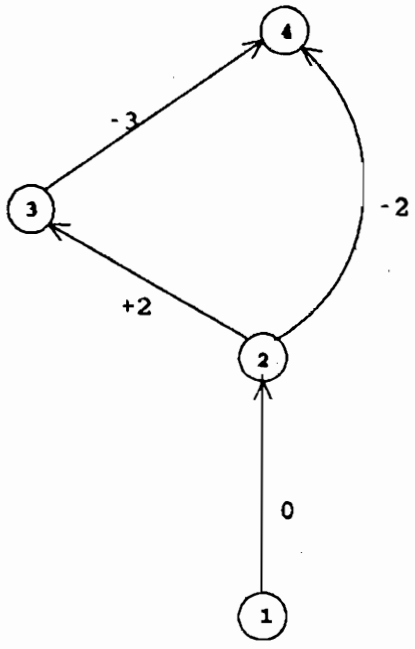


Figure-2b. G_n. MRA(G_n) consists of nodes 1,2,3,4,5,6
W(MRA(G_n))= 1+1-3 = -2

Figure-2. Correspondence between MRA(Ga) and MRA(Gn) when Ga is a tree: an Illustration



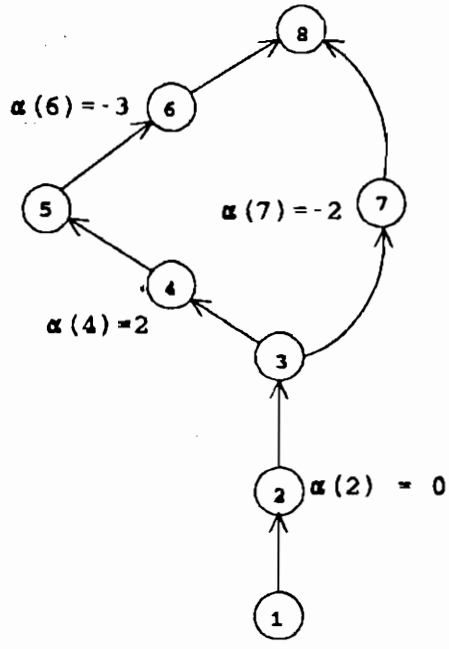


Figure-3a. G_a :W(MRA(G_a)) = -2

Figure-3b. G_n: W(MRA(G_n)) = -3

Figure-3. Correspondence between MRA(Ga) and MRA(Gn) when Ga is not a tree: an Illustration.

We present below a four-phase heuristic for selecting a rooted arborescence from a given G_a. This is similar to, but not the same as, the heuristic presented in section V of [Rao and McGinnis 84] to construct an RA(G_n).

Phase 1. For each $(i,j) \in \theta_{in}$, if there are any other arcs directed towards j, make the weights of those arcs ∞, to prevent them from entering the solution.

Phase 2. For each node in G_a find the minimum weight rooted path. Among all the nodes, if the minimum of the minimum weight paths found above has a negative weight, then we place the arcs (and corresponding nodes) in that path in bucket B1. Then the weights of all selcted arcs are updated to zero. Furthermore the weight of each unselected incident arc on each selected node is updated to ∞. Once again, in the updated graph the rooted path with minimum weight is found, and if its weight is negative then all its arcs (and nodes), except those already present in B1, are entered in B1, while at the same time updating the weights of the arcs as described above. We continue this process until the minimum weight rooted path in the graph happens to have a zero or positive weight.

Phase-3. As long as there are negatively weighted arcs in the graph we continue to select the minimum weight rooted path and update the weights. But, this time, we place the selected arcs and nodes in a second bucket B2. Also, we keep a running total of the original weights of the new arcs entered in B2, that is, those not already present in B1. When this total becomes negative, all the indices in B2 are transferred to B1.

This phase would end only when no negative weight arcs are left in the graph.

Phase-4: In this phase, we force in the arcs which are in θ_{in} and are not selected in the above phases. For each such arc we find the minimum weight rooted path and enter the corresponding arcs and nodes into B1.

At the end of phase-4 all the arcs in B1 form the arcs of the desired RA.

Example. Figure-4a shows a G_a with 11 nodes and 14 arcs. $\theta_{out} = \emptyset$, and $\theta_{in} = \{(8,9)\}$.The above four phase heuristic is applied on this graph to generate an MRA of this graph. The computations are shown through Figure-4a through 4f. The major steps are described below:

Phase 1. On node 9, two arcs, (4,9) and (8,9) are incident. Make W(4,9) as ∞, to obtain the graph of Figure-4b.

Phase 2. Enter the arcs on the path from 1 to 5 in B1 (see Figure-4c). At this stage the arcs in B1 are: (1,2),(2,3),(3,5). Update the weights of (1,2),(2,3) and (3,5) to zero. At node 5, (6,5) is the unselected incident arc. Hence make its weight ∞ (Figure-4d).

In Figure-4d, node 7 has the minimum weight rooted path with negative weight. Hence enter (3,7) in B1. Update W(3,7) to zero, and W(5,7) to ∞ (see Figure-4e). At this stage, the arcs in B1 are: (1,2),(2,3),(3,5),(3,7).

In Figure-4e, node 6 has the minimum weight rooted path with negative weight. Hence enter (3,6) in B1, and update W(3,6) to zero. The arcs in B1 at this stage are: (1,2),(2,3),(3,5),(3,7),(3,6). At this stage no rooted path has a negative weight. Hence phase 2 ends.

Phase 3. There are still three negatively weighted arcs. Therefore, we have to add their rooted paths one after the other to B2, till the running total of the arc weights becomes negative. The arcs are added to B2 in the follwing sequence:

i. Add the arcs (2,4),(4,8),(8,11); running total of weights in B2 becomes 1.
ii. Add the arc (8,10); running total of weights becomes 0.
iii. Add the arc (8,12); running total of weights becomes -1.

Now, transfer the arcs (2,4),(4,8),(8,11),(8,10),(8,12) to B1.
This is the end of Phase 3 with B1 having the arcs:
(1,2),(2,3),(3,5),(3,7),(3,6),(2,4),(4,8),(8,11),(8,12).

Phase 4. (8,9) is in θ_{in} and is not in B1. Hence add (8,9) to B1. Thus , finally the arborescence selected by the heuristic consists of the following arcs:

(1,2),(2,3),(3,5),(3,7),(3,6),(2,4),(4,8),(8,11),(8,12),(8,9).

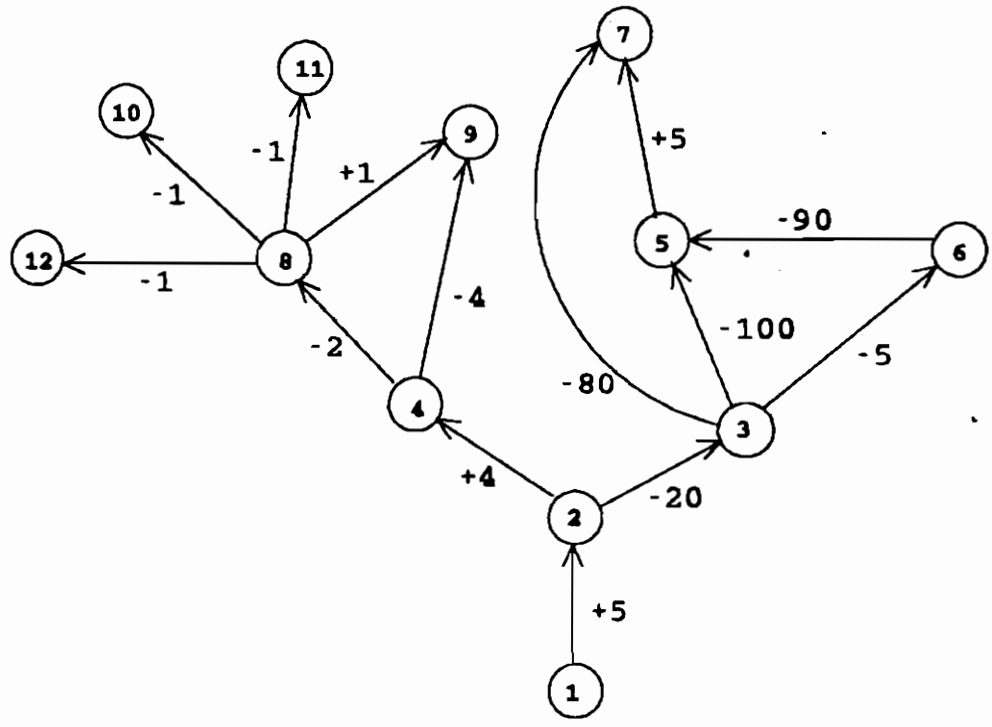


Figure-4a.

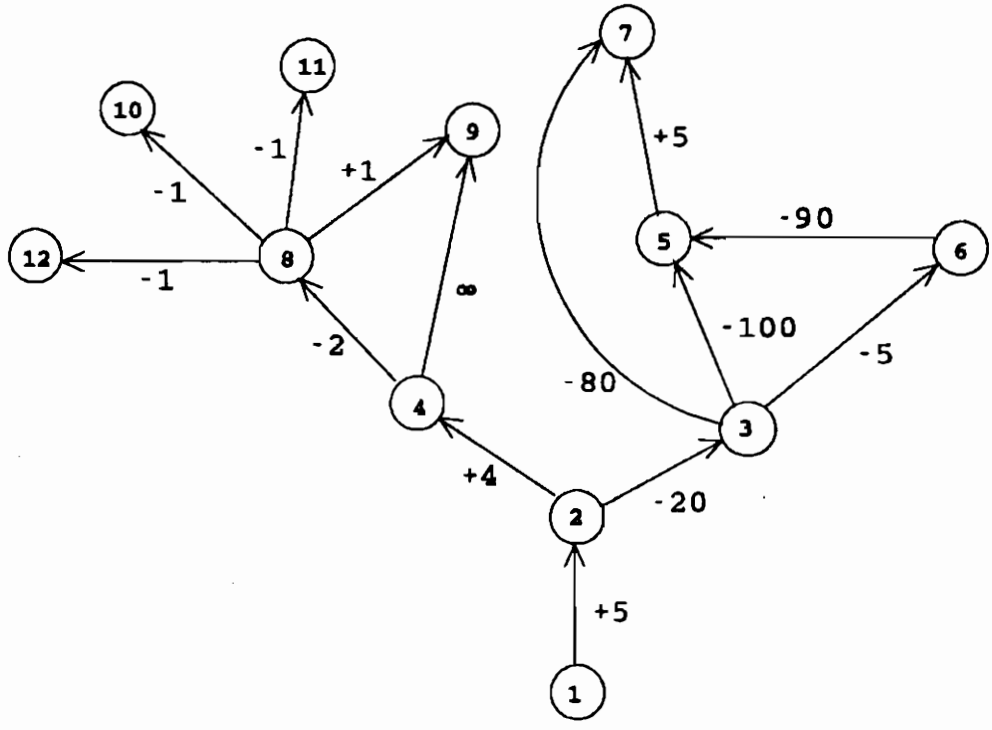


Figure-4b.

Figure-4. Example on constructing an RA(G_a) (Contd.)

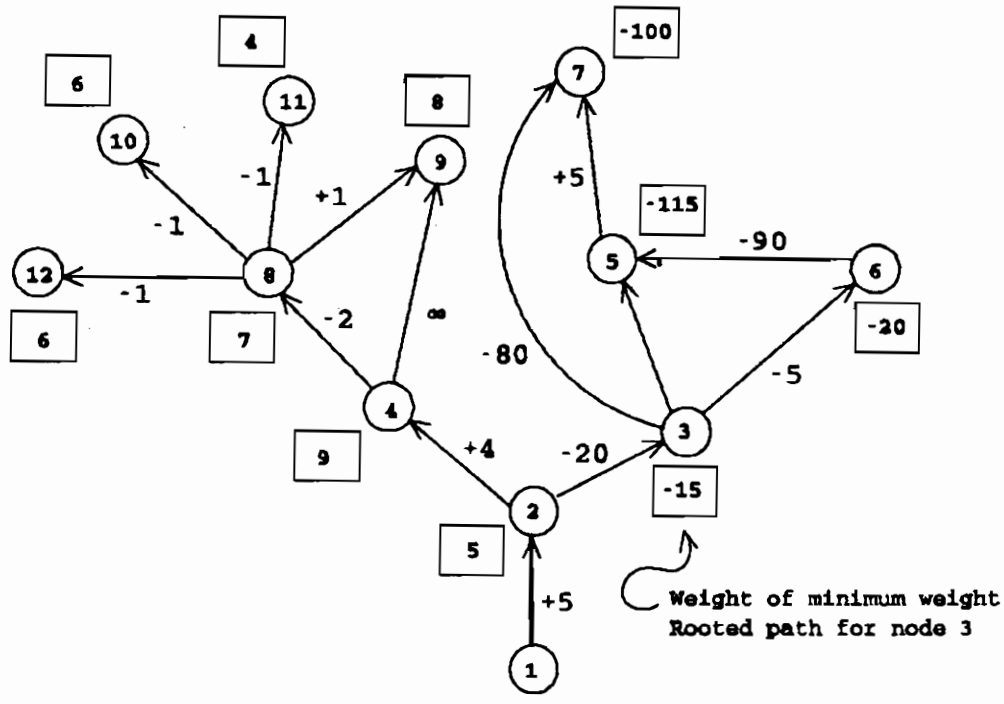


Figure-4c.

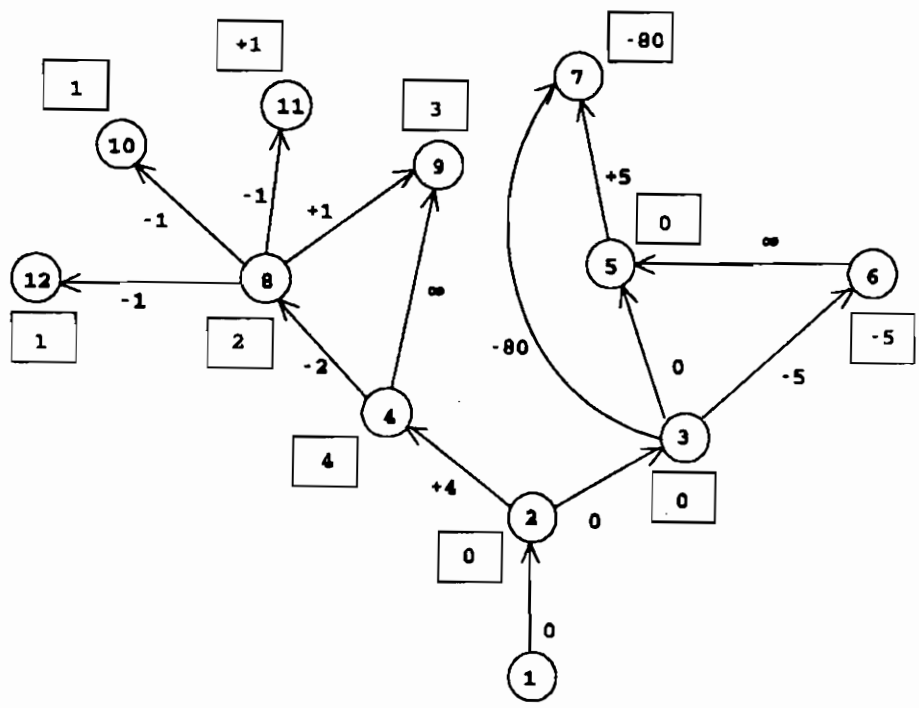


Figure-4d

Figure-4 (Contd.)

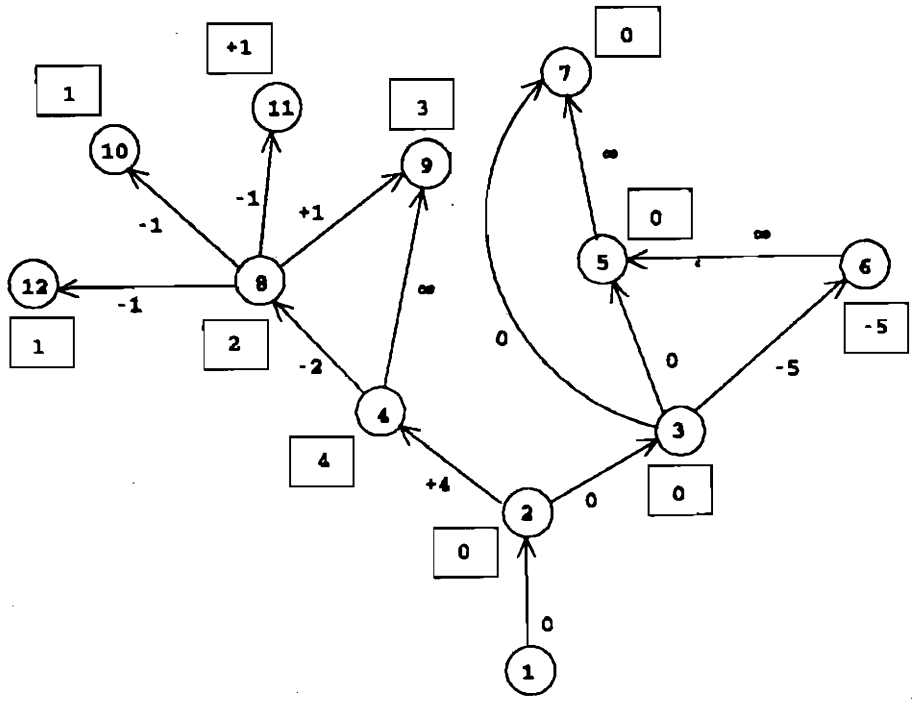


Figure-4e

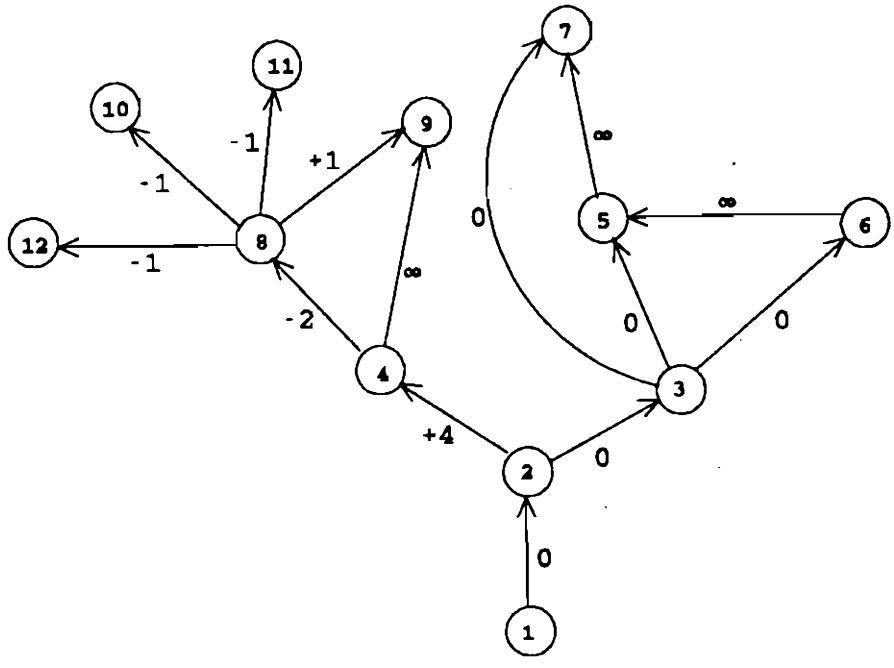


Figure-4f

Figure-4.

6. Lagrangian Relaxation of the MRA(G_a) problem

The motivation for developing a Lagrangian relaxation LMRA(G_a), for MRA is two fold: 1. using LMRA, it will be possible to obtain a lower bound on the optimal objective value of the MRA problem, 2. a powerful heuristic for the MRA problem can be developed using LMRA.

The effectiveness of this method depends largely on which constraint set from the original problem we choose for relaxation; the resulting Lagrangian problem should be far easier to solve than the original problem.

We discuss below two alternative ways of formulating the Lagrangian problem.

1. The constraint set chosen for relaxation is the set of the incidence constraints (3). For this, consider the multipliers U = {U(j), j=2, ...,N}, each U(j) associated with the constraint corresponding to the j^{th} node. We can formulate our first Lagrangian problem $LMRA^1(D_a)$ as:

$LMRA^1(D_a)$.

$$\underset{U \geq 0}{Max} \quad \underset{Y}{Min} \left[\sum_{(i,j) \in E} W(i,j)Y(i,j) + \sum_{j=2}^{N} U(j)\{(\sum_{i \in P(j)} Y(i,j)) - 1\} \right] \qquad (11)$$

s.t.(4) *and* (5) *and* (6)

The constraints of the above problem resemble those of MRA(G_n), (1). However, the objective function (13) does not exhibit any known form. Hence, we do not have any readily available, easy way of solving it, and hence do not consider it further in this paper. Even if the objective function were to have the same form as MRA(G_n), since MRA(G_n) is also NP-hard, an optimal solution to MRA(G_n) is not easy to obtain.

2. The constraint set chosen for relaxation here is the set of connectivity constraints (4). For this, consider the multipliers U = {U(i,j)| (i,j)$\in$ E, i >1}, each U(i,j) being associated with the connectivity constraint corresponding to arc (i,j). Therefore, the second Lagrangian problem $LMRA^2(D_a)$ can be written as

$LMRA^2(D_a)$.

$$\underset{U \geq 0}{Max} \quad \underset{Y}{Min} \left\{ \left(\sum_{(i,j) \in E} W(i,j)Y(i,j) \right) + \sum_{(i,j) \in E, i>1} U(i,j) \left(Y(i,j) - \sum_{k \in P(i)} Y(k,i) \right) \right\} \qquad (12)$$

s.t.(3) *and* (5) *and* (6)

As a given arc (i,j) is directed towards only one node j, the different constraints in (3) are separable for each j. Further, the constraints (3) are in the form of the constraints in a knapsack problem. Hence, for a given set of multipliers U, LMRA^2 can be solved by splitting it into several independent knapsack problems, one for each j, j = 2,...., N-1. It is useful to rewrite the objective function (13), by regrouping the terms, in the following form:

$$\underset{U\geq 0}{Max}\ \underset{Y}{Min}\left\{ \sum_{(i,j)\in E} Y(i,j)\left[W(i,j)+U(i,j)-\sum_{k\in S(j)} U(j,k)\right]\right\} \qquad (13)$$

Example. To illustrate the separability of LMRA^2 into several knapsack problems, let us consider an example. Let G_a be the graph shown in Figure-5. We give below the Lagrangian problem LMRA^2 for the graph of Figure-5.

This problem can be separated into three independent problems whose objective functions are $Z_1(U)$, $Z_2(U)$, and $Z_3(U)$. Then overall objective function of the problem can be written as

$$Z=\underset{U}{Max}\ Z(U)=\underset{U}{Max}\ \{Z_1(U)+Z_2(U)+Z_3(U)\} \qquad (14)$$

where:

$Z_1(U)$ = Min {Y(1,2)[W(1,2)+U(1,2)-U(2,3)-U(2,4)+
Y(2,3)[W(2,3)+U(2,3)-U(3,4)-U(3,5)]} (15)
s.t. Y(1,2)=1, and Y(2,3) $\in$ {0,1} (16)

$Z_2(U)$ = Min {Y(3,4)[W(3,4)+U(3,4)-U(4,5)]+
Y(2,4)[W(2,4)+U(2,4)-U(4,5)]} (17)
s.t. Y(3,4)+Y(2,4) $\leq$ 1 (18)
Y(3,4), Y(2,4) $\in$ {0,1} (19)

$Z_3(U)$ = Min {Y(3,5)[W(3,5)+U(3,5)]+Y(4,5)[W(4,5)+U(4,5)]} (20)
s.t. Y(3,5)+Y(4,5) $\leq$ 1 (21)
Y(3,5),Y(4,5) $\in$ {0,1} (22)

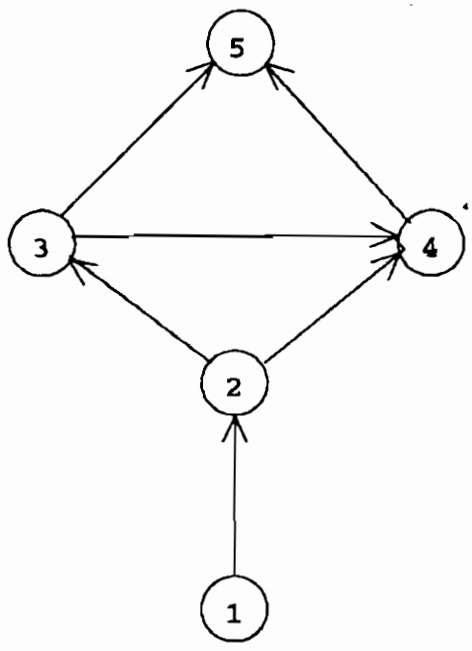


Figure-5. Graph to illustrate LMRA^2.

It can be seen that the problem (15)-(16) can be easily solved and that (17)-(19) and (20)-(22) can be solved by picking the variable with least negative coefficient in the objective function; no variable is picked if all coefficients are positive.

Lagrangian Heuristic for MRA(G_a)

We propose a heurstic solution for MRA(G_a) based on LMRA^2 discussed above. At each Lagrangian iteration, for a given U, we obtain a lower bound for problem MRA(G_a) by solving (13) subject to (3), (5) and (6). This will return a set of Y(i,j)'s with value one, which may not satisfy (4). Through the algorithm described later, we can construct a feasible solution to MRA(G_a) with the above Y(i,j)'s set at 1. Hence, at every Lagrangian iteration we will generatge a lower bound as well as an upper bound for MRA(G_a). This heuristic procedure is described below:

Step 1: Identify an initial upper bound by using the heuristic described in section 5. Initialize the Lagrange multipliers U(i,j) = 0, for (i,j)$\in$ E.

Step 2: For a given U, solve (13) subject to (3), (5) and (6). This provides a lower bound Z_{LB} for MRA(G_a). Update the lower bound if necessary.

Step 3: Step 2 will fix some of the Y(i,j)'s at one. Use this set of Y(i,j)'s to generate a feasible solution for MRA(G_a). The procedure for generating such a feasible solution is described in the next section. Update the upper bound, Z^{UB},if necessary .

Step 4 (Stopping rule): If $Z^{UB} = Z_{LB} + \varepsilon$, stop; we have an optimal solution. The value of epsilon is taken to be .99, since all our data are in integers. If the iteration count has exceeded, stop. If the lower bound converges to a particular value, stop. If none of the above conditions are encountered, go to step 5.

Step 5 (Updating U(i,j)'s). Update the Lagrange multipliers using the subgradient procedure described below. If the subgradients are all zero, stop. Else go to step 2.

Subgradient procedure. The subgradient procedure used in our algorithm is described now. For a detailed analysis of subgradient optimization see [Held, Wolfe, and Crowder,1974]. Let $Y^*(i,j)$ be the optimal solution to the Lagrangian problem $LMRA^2$. Let k be the iteration number. We then compute the subgradients NU(i,j) for U(i,j) as

$$NU(i,j) = Y^*(i,j) - \sum_{(h,i)\in E} Y^*(h,i)$$

The Lagrange multipliers are then updated as follows:

$$u(i,j)^{k+1} = \max\{u(i,j)^k + t_k NU(i,j), 0\}$$

where

$$t_k = \lambda(Z^{UB} - Z_{LB}) / \sum_{(i,j)} [NU(i,j)]^2$$

We start with an initial value of λ and halve the value every 8 iterations if the lower bound does not improve.

Generation of an upper bound in the Lagrangian heuristic. As mentioned earlier an upper bound is computed in every Lagrangian iteration which returns a set of Y(i,j)'s fixed at one. The procedure for computing the upper bound will be described now.

Let Y(i,j) be a given solution such that

$$\{Y(i,j) \mid \sum_{(i,j)\in E} Y(i,j) \le 1 \; for \; j=2,3,\ldots N,\; Y(i,j)\in\{0,1\} \; for \; (i,j)\in E,\; Y(1,2)=1\}$$

From the above solution, we want to generate a feasible solution to the $MRA(G_a)$ problem.

Let s be a subgraph of G_a such that it consists of nodes i and j and arc (i,j) for each Y(i,j)=1 in the above solution.

Note that the set of arcs in s satisfy the incidence constraints, but not necessarily the connectivity constraints. It is possible that some nodes and arcs of the subgraph s are not connected to the root node 1. The following heuristic considers such nodes and establish proper connections such that the resulting graph is connected. Finally, after obtaining a connected graph, the heuristic attempts to improve the weight of the graph by pruning some

wasteful ends.

The heuristic consists of three phases. In phase I, we note whether a node of G_a has at least one of its direct and/or indirect successors is present in s. In phase II, we obtain a solution by including some more arcs of G_a in s , if necessary, so that the connectivity constraints are satisfied by the arcs of s. In phase III, some of the arcs in s are removed to improve the weight of the arborescence. These phases are elaborated below.

Phase I. This is essentially a procedure for developing a label m(i) for each node i, by considering the nodes of G_a in decreasing order of node index, that is in the order N,N-1,...,2. The rules for labelling are designed such that a zero label for a node indicates that none of its successors, direct or indirect, are present in s, and a positive label indicates that atleast one of its direct or indirect successors is present. When a node i is considered for labelling, first we check whether it is a terminal node. If it is, then its label is set to zero, that is m(i) <-- 0; otherwise, its label is set equal to :

$$\sum_{k,\ (i,k)\in s} \{m(k)+1\} + \sum_{k,\ (i,k)\notin s} m(k)$$

Phase II. Let $\beta(1)=0$, and $\beta(2)= W(1,2)$. Here we scan the nodes i of G_a from 2 to N in increasing order of node index. If m(i)=0 then we need not worry about connecting it to the root, because none of its successors is present in s. If m(i) > 0, then we check whether any of its immediate predecessors k and the corresponding arc (k,i) are present in s. If so, there already exists in s a connected path from 1 to i; we compute $\beta(i)$ as equal to $\beta(k) + W(k,i)$. Else, one of the arcs (k,i), $k \in P(i)$ has to be included in s. If i has more than one predecessor k, then we choose k such that $\beta(i) = \beta(k) + W(k,i)$ is a minimum, among all $k \in P(i)$.

Phase III. At the end of the above phase, arcs in s form a rooted arborescence. Each node i in this arborescence can be looked upon as the root for a sub arborescence in s. Let the sub arborescence for which i is the root be referred to as $s^1(i)$. In phase 3, we compute for each $i \in s$, $W(s^1(i))$. Then we scan the nodes of i from 1 to N and when $W(s^1(i))$ is greater than or equal to zero, then we remove $s^1(i)$ from s. This improves the weight of s.

Example. Consider the graph G_a of Figure-6. Suppose the solution given by the sub gradient algorithm in one iteration is :

Y(1,2)=Y(7,9)=Y(13,14)=Y(8,10)=1; other Y(i,j)'s are zero.

The sub graph s corresponding to the above solution is shown in the figure in thick lines; others are shown in thin lines.

Application of phase I yields the labels shown in square blocks alongside each node in Figure-6a.

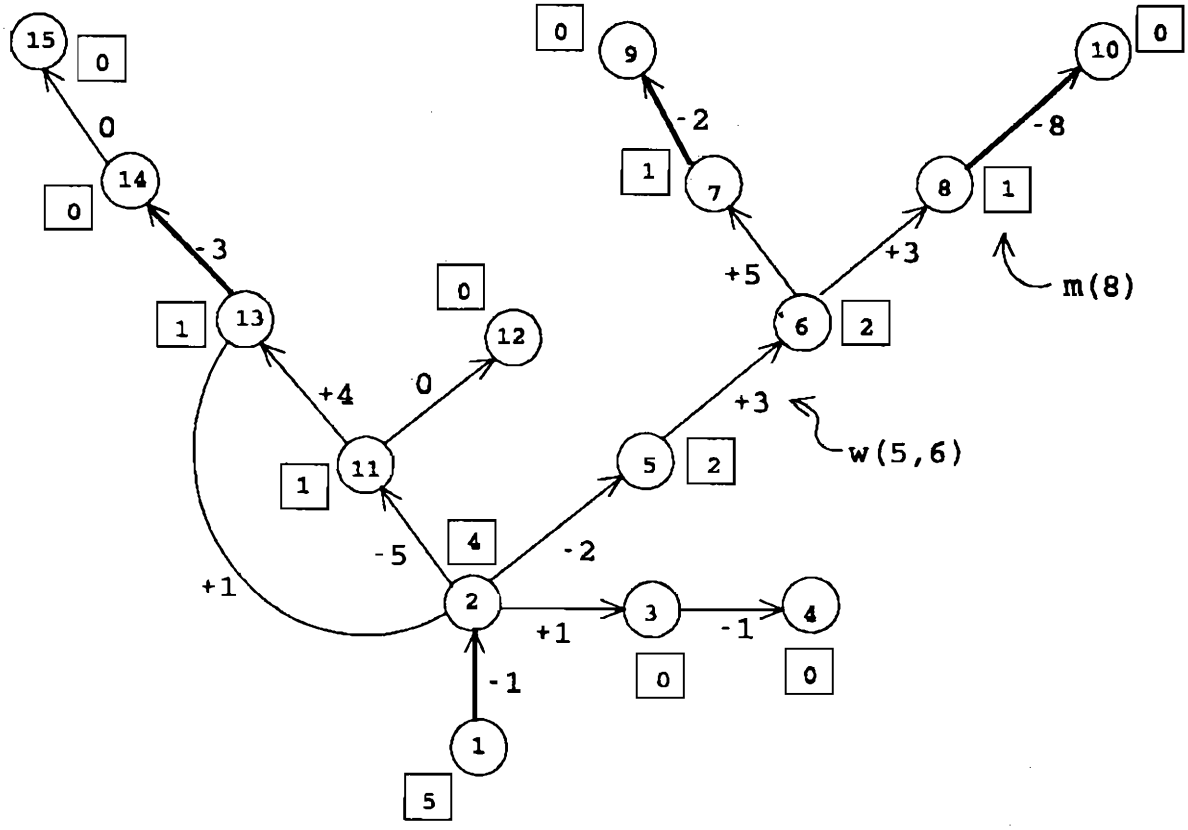


Figure-6a.

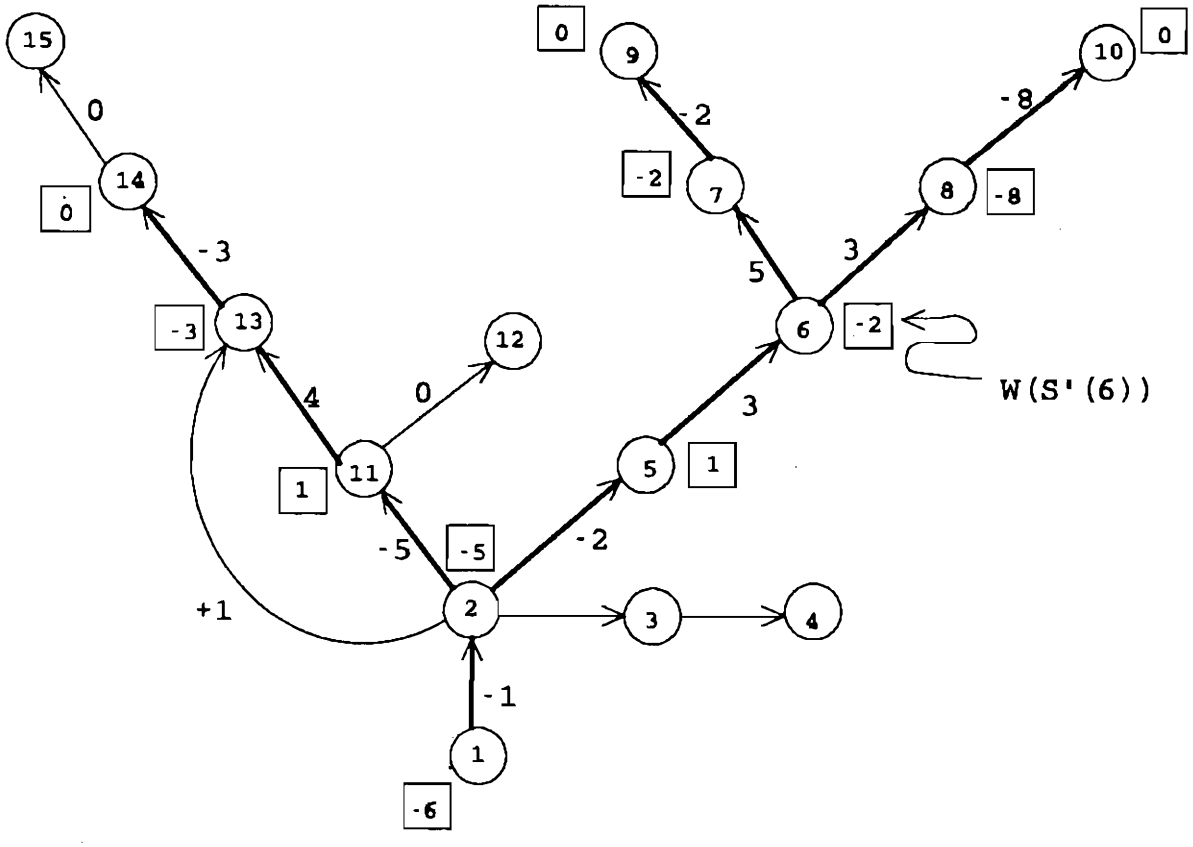


Figure-6b.

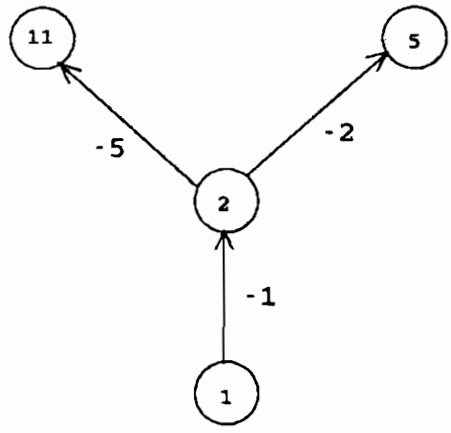


Figure-6c.

Figure-6. Example on generating upper bound in Lagrangian heuristic.

In phase II, we examine the nodes of G_a in the order 2,3,...,15. Node 2 is in s and is connected to the root node 1. Nodes 3 and 4 are currently not in s and need not be considered for inclusion in s, because their labels are zero. Node 5 is not in s , but m(5) is greater than zero. Hence node 5 has to be included in s along with the only possible connecting arc (2,5). Similarly, node 6 also has to be included in s along with arc (5,6), node 7 along with arc (6,7), node 8 along with (6,8), and node 11 along with (2,11). Node 12 need not be included in s because s(12) = 0. Node 13 is in s and is not connected to the root. Hence select (11,13) or (2,13), whichever results in the lesser weight path from the root to node 13; accordingly, select (11,13). Node 15 need not be considered for inclusion in s because m(15) = 0. The arborescence obtained at the end of phase II is shown in Figure-6b.

The values of $W(s^1(i))$ for each node i are shown in the form of labels by the side of each node in Figure-6b. From this figure it is clear that the sub tree with 5 as root and that with 11 as root have to be removed. The resulting arborescence is shown in Figure-6c.

7. Linear Relaxation of $MRA_\theta(G_a)$, $LPMRA_\theta(G_a)$

This problem is the same as $MRA_0(G_a)$ problem (7)-(10) except that the the (0,1) constraints (10) are replaced by their linear relaxation

$$Y(i,j) \leq 1, \ (i,j) \in \theta_{free}$$

It is important to note that in general the solution of $LPMRA_\theta(G_a)$ cannot be expected to be a zero-one solution, and hence the optimal value of the objective functoion of $LPMRA_\theta(G_a)$ can serve only as a lower bound on the weight of $MRA_\theta(G_a)$. To illustrate this point, consider the graph G_a shown in Figure 7. An optimal solution for the $MRA(G_a)$

$LPMRA_\theta(G_a)$ can serve only as a lower bound on the weight of $MRA_\theta(G_a)$. To illustrate this point, consider the graph G_a shown in Figure 7. An optimal solution for the $MRA(G_a)$ problem for this graph is

Y(1,2)=Y(2,3)=Y(3,4)=Y(4,6)=Y(6,7)=1, and other Y(i,j)'s are zero. Thus W [$MRA(G_a)$] = 0+1+0+1-100=2-100 = -98.

An optimal solution of $LPMRA(G_a)$ is

Y(1,2)= Y(6,7)=1; Y(2,3)=Y(3,4)=Y(3,5)=Y(4,6)=Y(5,6)=0.5; Y(3,6)=0,

making the value of the objective function equal to -98.5, which is less than W($MRA(G_a)$.

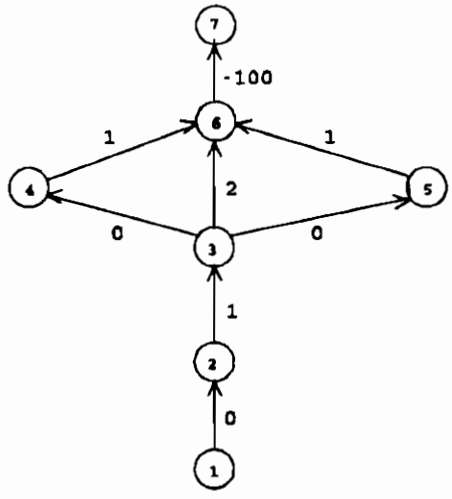


Figure-7

8. A branch and bound method to solve $MRA(G_a)$

We propose here a branch and bound method for the $MRA_\theta(G_a)$ problem. In the proposed method, the branch and bound tree is a binary tree. The branching variables are Y(i,j)s. At each node in the tree some Y(i,j)s are fixed at 1, some at zero, and others are free. The two descendents from a node are obtained by fixing one of the free Y(i,j)s at 1 in one

descendent, and at zero in the other. The very first candidate problem in the tree corresponds to the $MRA(G_a)$ problem, whereas at the other nodes the problems are of the type $MRA_\theta(G_a)$, (7) -(12). At each node an upper bound on $W[MRA_\theta(G_a)]$ is obtained by using the heuristic of section 5. A lower bound on $W[MRA_\theta(G_a)]$ can be generated either by solving the Lagrangian problem $LMRA_\theta(G_a)$ of section 6, or by solving the linear relaxation $LPMRA_\theta(G_a)$, of section 7. The branch and bound scheme is outlined in Figure-8.

Branching

In choosing a Y(i,j) for branching from the candidate problem we make use of the weight of the minimum weight path from root to j, the ending node of arc (i,j). For each arc (i,j) $\in \theta_{free}$ the minimum weight rooted path is found. Suppose the arc (i^*, j^*) has the minimum of minimum weight paths found above. Then, $Y(i^*,j^*)$ is the variable selected for branching. $Y(i^*,j^*)$ is forced to one in the left branch and to zero in the right. The candidate problem for the right branch is entered in the candidate queue first and then the left one. The order of retrieval of candidates from the branch and bound tree is last-in-first-out.

Figure-8
Branch and Bound Algorithm: Outline

```
BEGIN
    The problem at the very first node is MRA_θ(G_a) where
    θ_in = {(1,2)}, θ_out = Ø, θ_free = E\(1,2)
    Enter the first node in candidte queue
    Value of incumbent <-- ∞
    REPEAT
        Remove the last candidate from the candidate queue and make it the current node.
        Attempt to construct a feasible solution to MRA_θ(G_a) where θ_in, θ_out, and θ_free
        correspond to the current node (Algorithm of section 5).
        IF there is a feasible solution to MRA_θ(G_a) THEN
            IF W[MRA_θ(G_a)] < value of incumbent solution THEN
                value of incumbent solution <-- W[MRA_θ(G_a)]
                incumbent solution <-- solution obtained for MRA_θ(G_a)

            ENDIF
            Generate lower bound LB on W[MRA_θ(G_a) by solving problem
            LMRA_θ(G_a) or LPMRA_θ(G_a)
            IF LB > value of incumbent solution THEN
            Fathom the current node
            ELSE
                IF θ_free is not empty THEN
                Branch from current node.
                ELSE
                Fathom the current node
                ENDIF
            ENDIF
        ELSE
        Fathom the current node
        ENDIF
    UNTIL the candidate queue is empty.
    Optimal solution <-- incumbent solution
END
```

9. Concluding remarks

The essential contribution of this paper lies in formulating the MRA(G_a) problem and in proposing two solution methods, the first a heuristic and the second an optimization algorithm, for it. Even though the performance of the proposed algorithms is still to be studied, the properties of the problem discussed here, in particular, the correspondence between the weights on arc and weights on node problems, are believed to be of interest in themselves. We have posed the MRA(G_a) problem as an interesting problem in itself, and we hope that future research in this area will reveal some practical applications for it.

References

1. Rao, V.Venkata, McGinnis, L., *The Minimum Weight Rooted Arborescence Problem: A Branch and Bound Solution*, Working Paper No. 514, Indian Institute of Management, Ahmedabad, June 1984.

2. Held,M., Wolfe,P., and Crowder, H.D., *Validation of Subgradient Optimization*, Mathematical Programming, Vol 6, 1974, pp 62-88.
