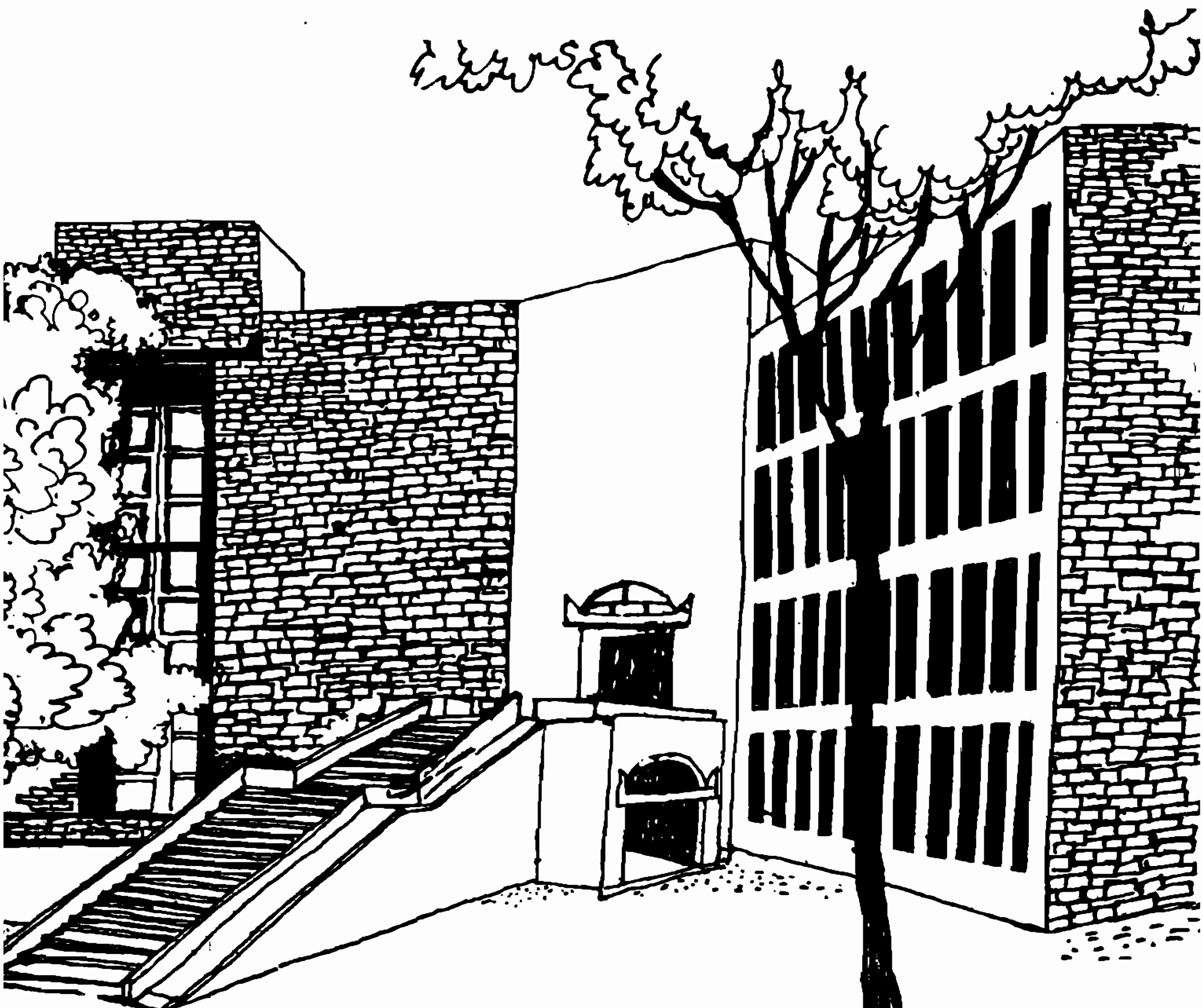
Working Paper



PRIMAL AND LAGRANGIAN HEURISTICS FOR MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM

By

V. Venkata Rao
R. Sridharan

W.P. No.1353
February 1997

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD - 380 015
INDIA

Primal and Lagrangian Heuristics for Minimum Weight Rooted Arborescence Problem

V. Venkata Rao and R.Sridharan
Indian Institute of Management
Ahmedabad 380015

Abstract

Consider a rooted acyclic graph G with weights on arcs. In this graph, a minimum weight rooted arborescence (MRA) can be defined as one whose sum of arc weights is less than or equal to that of any other rooted arborescence (RA) in that graph. We introduce a Lagrangian heuristic for this problem and present computational results.

First, we formulate the MRA problem as a zero-one integer program and discuss a heuristic H to construct an RA in a G. This heuristic generates an upper bound on the value of the objective function for the MRA problem. Next, we formulate a Lagrangian problem LMRA by relaxing one set of constraints of the zero-one program. In the process of relaxation, a set of multipliers U are required, one for each constraint to be relaxed. For a given set of U's, LMRA can be easily solved to optimality by separating it into several independent knapsack problems. Finally, for MRA, we propose a Lagrangian heuristic that iterates between the upper bound heuristic H and the knapsack solution for LMRA. Beginning with an upper bound generated by H, and an initial set of multipliers U's, we solve LMRA and obtain a lower bound for MRA, at the same time generating a partial solution which can be completed by H, thus getting a new upper bound for MRA. The iterations continue till either the best upper bound and best lower bound come close enough, or a suitable stopping condition is satisfied.

The Lagrangian heuristic was tested with fifty test problems, the number of nodes in the problems varying from ten to fifty-five. The following output was collected for each test problem: the value of the best upper bound, the value of the best lower bound, iteration numbers corresponding to the best upper and lower bounds, initial upper bound given by heuristic H, total number of iterations executed when the program stopped, number of times the value of the solution given by H was improved, and the total execution time in milli-seconds.

In a high percentage (86%) of the test cases, the Lagrangian heuristic yielded optimal solution. In forty percent of the cases, the initial solution obtained by the heuristic H itself turned out to be optimal. The execution time was less than 2 seconds for most of the test problems. Thus, the proposed heuristic seems promising enough to warrant further study.

Primal and Lagrangian Heuristics for Minimum Weight Rooted Arborescence Problem

1. Introduction

A connected acyclic graph, G, with one of its nodes (*root node*) not having any incoming arcs and having exactly one outgoing arc (*root arc*), consists of, in general, several rooted (not necessarily spanning) arborescences. 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 arborescence in the graph under consideration. Of the different types of rooted acyclic graphs, we are in particular interested in two: 1. rooted acyclic graph with weights on nodes, and 2. rooted acyclic graph with weights on arcs. In the first category, an optimal rooted arborescence can be defined as one whose sum of node weights is less than or equal to that of any other rooted arborescence in G; the problem of finding such an arborescence is called the *minimum weight rooted arborescence problem in an acyclic rooted graph with weights on nodes*. Similarly, in the second category, an optimal rooted arborescence can be defined as one whose sum of arc weights is less than or equal to that of any other rooted arborescence in G; the corresponding problem is called *the minimum weight rooted arborescence problem in a rooted acyclic graph with weights on arcs*.

The minimum weight rooted arborescence problem with weights on arcs was introduced in [5]. As we show later, this problem is related to the uncapacitated plant location problem. Further, the minimum weight rooted problem with weights on nodes, which turns out to be a special case of the problem with weights on arcs arises in an integer programming model of a multistage production system [3]. Hence, we have chosen to focus this paper on the minimum weight rooted arborescence problem with weights on arcs (*MRA problem*).

This paper is a sequel to [5], our earlier paper on the same subject. [5] outlined certain algorithms for the MRA problem, but, as the algorithms were not tested at that time, the paper did not report computational results. This paper fills that gap. Even though this paper is mainly concerned with testing of the algorithms already proposed in [5], we repeat here, for the sake of completeness, the definition of the problem and the statement of the algorithms, with improved notation and terminology.

In section 2 we introduce the basic definitions, notation, and terminology of the paper. We formulate in section 3 the MRA problem as a zero-one integer programming problem. In section 4 we discuss the relation between MRA problem and the uncapacitated plant location problem (UL). We present in section 5 a heuristic to construct a rooted arborescence RA in G. In section 6, we discuss the formulation of a Lagrangian Dual of MRA problem. We propose a Lagrangian heuristic for the MRA problem in section 7. In section 8, we report our computational experience with the heuristics. Finally, in section 9 we give concluding remarks.

2. Definitions, Terminology and Notation

Let G = {N,A} be a connected acyclic graph, with $N > 2$. The nodes are indexed with consecutive integers 1, 2, ..., 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. 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.

Additionally, G is called a *rooted acyclic graph* if it is an acyclic graph possessing exactly one node, called the *root node*, with no incoming arcs; this root node, node 1, is connected only to node 2 by arc (1,2), called the *root arc*. If each arc (i,j) of G carries a weight W(i,j), where W(i,j) is any real number, positive, negative or zero then G is called a *rooted acyclic graph with weights on arcs*. From now on, by default, we deal with only rooted acyclic graphs with weights on arcs.

A subgraph RA(G) of G is called a *rooted arborescence* of G if: 1. RA(G) contains the root arc as one of its arcs, 2. RA(G) is connected, and 3. no two arcs of RA(G) are directed towards the same node. The sum of weights of the arcs in RA(G) is called the weight of the rooted arborescence.

As mentioned earlier, a rooted arborescence of a graph G is called a minimum weight rooted arborescence (MRA) of G if its weight W[MRA] is less than or equal to the weight of every other rooted arborescence RA(G) of that graph. See Figure-1 for an illustration of MRA.

A *rooted path* for a node k, $k > 1$, in G is an alternating sequence of nodes and arcs, starting with root node 1 and ending with node k, which can be written as

[1, (1,2), 2, ..., (i,j), j, (j,k), k].

The sum of weights of the arcs in a rooted path of a node is called the weight of the rooted path; among all the rooted paths of node k, that which has the minimum weight is called the minimum rooted path of node k, denoted by MWRP(k), and its weight by w(MWRP(k)). Among all the nodes of G, let i be the node which has the lowest weight minimum rooted path. Then the minimum weight rooted path of i is called minimum weight rooted path in G, or simply the *min-weight rooted path*, denoted by MWRPG. The value of the weight of MWRPG is denoted by w(MWRPG). MWRPG can be easily found by a shortest path algorithm.

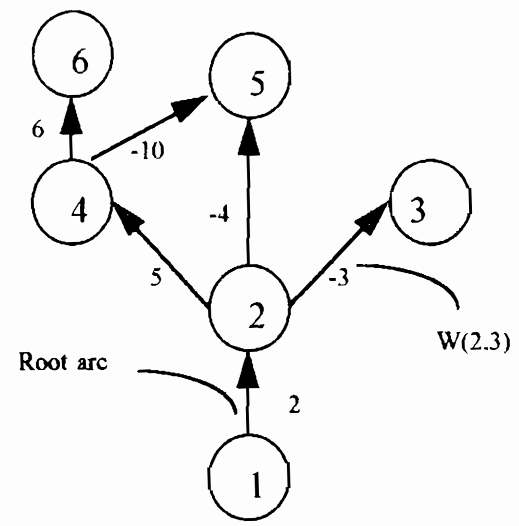


Figure 1a. A rooted acyclic graph with weights on arcs

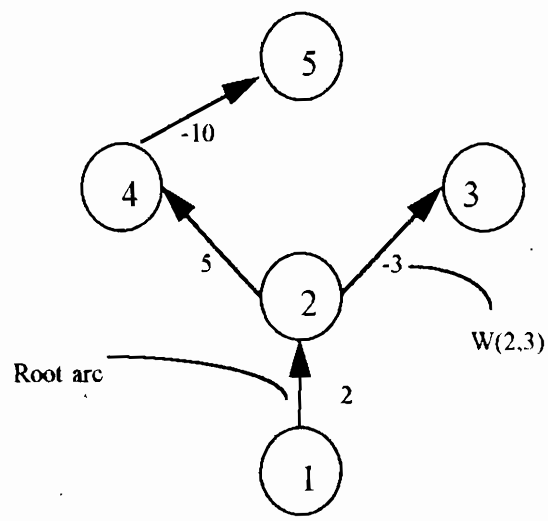


Figure 1b. MRA of the graph of Figure 1a. w(MRA) = -6

Figure 1. MRA: an illustration .

See appendix-1 for a summary of the important symbols used in this paper.

3. Relation between Uncapacitated Plant Location Problem (UL) and MRA Problem.

We show here that UL [1] can be formulated as an MRA problem. This reduction is significant, because UL is a widely studied NP-Hard problem.

Consider an UL with m plants and n demand points, with a fixed cost associated with each plant, and a variable cost associated with each plant-demand point pair. Reduce this UL to an MRA as follows. Formulate a directed bi-partite graph with the first set of nodes corresponding to the plants and the second set corresponding to demand points. Link each plant-node with each demand-node by an arc directed from the plant-node to the demand-node, and let the arc carry a weight equal to the corresponding variable cost. Create a directed root arc with a weight of zero and connect its end node to each of the plant nodes by arcs directed towards the plant-nodes, each such arc carrying a weight equal to the fixed cost of the corresponding plant. Finally, create a duplicate node for each demand-node, and link it by an arc directed away from the original demand-node and carrying a weight of minus infinity. It can be seen that a minimum weight arborescence of this graph corresponds to an optimal solution of UL. Figure 2 contains an illustration of a UL and its MRA.

4. Mathematical Formulation of MRA Problem.

The MRA problem can be stated as a zero-one program as below:

$$Z=Min \sum_{(i,\ j)\in A} W(i,\ j)Y(i,\ j) \qquad (1)$$

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

$$Y(i,j) \le \sum_{k\in P(i)} Y(k,i),\ for\ (i,\ j)\in A \setminus (1,2) \qquad (3)$$

$$Y(1,2)=1 \qquad (4)$$

$$Y(i,j)\in\{0,1\}\ for\ (i,\ j)\in A \setminus (1,2) \qquad (5)$$

In the above problem, each zero-one variable, Y(i,j), corresponds to an arc (i,j)∈ A. In the solution, Y (i,j) = 1 implies that arc (i,j) is present in MRA; Y(i,j) = 0 implies that arc (i,j) is absent. Constraints (2) ensure that, in the selected subgraph, not more than one arc is directed towards a selected node; constraints (3) ensure that an arc is not selected, unless at least one of its predecessor arcs is also selected. Constraint (4) ensures that the root arc is definitely present in the final solution. In future, we refer to the constraints (2) as *incidence constraints* and (3) as *connectivity constraints*.

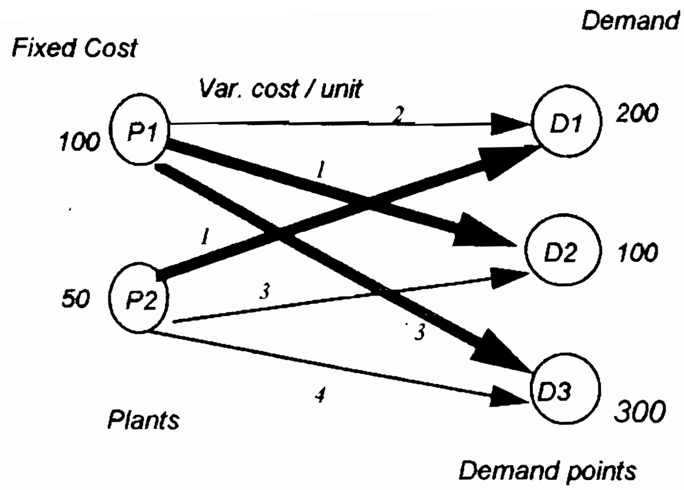


Figure 2a. UL
(Thick lines indicate an optimal solution)

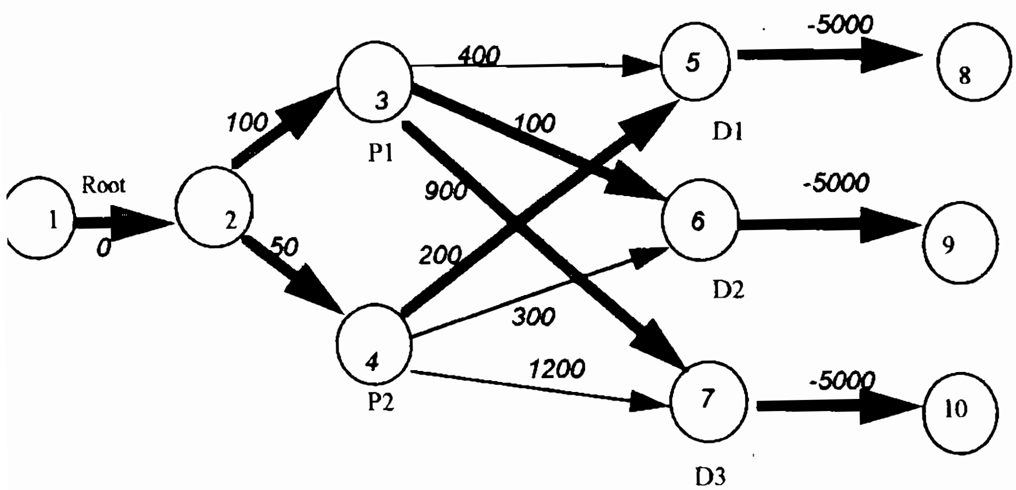


Figure 2b. Rooted acyclic graph
for the UL of Figure 2a
(Thick lines indicate an MRA)

Figure 2. UL and MRA Problem: an
Illustration

5. A Heuristic H for the MRA Problem

A heuristic solution for the MRA 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]. 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; these arcs are specified as members of set θ_{in}. Thus MRA_θ problem can be stated as below:

$$Min \sum_{(i,j)\in A} W(i,j)\,Y(i,j) \qquad (6)$$

$$s.t. \quad (2), \quad (3), and$$

$$Y(i,j)=1, \qquad (i,j)\in\theta_{in} \qquad (7)$$

$$Y(i,j)\in\{0,1\}, \qquad (i,j)\in A\backslash\theta_{in} \qquad (8)$$

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

We present below a four-phase heuristic H for selecting a rooted arborescence from a given G. This is similar to, but not the same as, the heuristic presented in Section V of [4].

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. Open two arc lists B1 and B2, and make them empty.

Phase 2. Find MWRPG in G. If w(MWRPG) is negative, then append the arcs of MWRPG to list B1. Update to zero the weights of all the selected arcs. Furthermore, update to infinity the weight of each unselected incident arc on each selected node. Once again, in the updated graph, find MWRPG, and if its weight is negative then append all its arcs to B1, except those already present in B1, while at the same time updating the weights of the arcs as described above. Continue this process until w(MWRPG) happens to have a zero or positive weight.

Phase 3. As long as there are negatively weighted arcs in the graph continue to select the MWRPG; and update its arc weights as described above. But, in this phase, append the selected arcs (excepting those already in B1) to a second list B2. Also, keep a running total of the original weights of the arcs entered in B2. When this total becomes negative, append all the arcs of B2 to B1, and make B2 empty.

End this phase only when no negative weight arcs are left in G.

Phase 4: For each unselected arc (i,j), if any, in θ_{in} find the minimum weight rooted path. Select from among these paths that with the lowest weight and append the corresponding arcs not already selected, to B1. Update the weights of its arcs as described in phase 2. Repeat this processes till arcs of θ_{in} are selected.

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

The above heuristic is defined in the form of a pseudocode in appendix-2.

Example to illustrate H. Figure 3a shows a G with 11 nodes and 14 arcs. $\theta_{in} = \{(1,2),(8,9)\}$. The above four phase heuristic is applied on this graph to generate an MRA of this graph. The status of the graph G at important stages in the application of H is shown in Figure 3a through 3d. 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 ∞.

Phase 2. This phase is comprised of three iterations. During each iteration, find the node which has the MWRPG. Figure 3b shows the weight of the minimum weight rooted path for each node during iteration 1. At the end of iterations 1, 2, and 3, the end nodes of MWRPG are 6,7, and 5 respectively. End this phase here, as none of the nodes has a minimum rooted path with a negative weight (Figure 3c). The arcs selected for the arborescence in this phase are: (1,2), (2,3), (3,6), (3,7), and (3,5). The computations of this phase are summarised in Table 1.

Table 1. Summary of computations in Phase 2 of the example for H

Iteration No.	Min-weight rooted path (MWRPG) in the graph		Arcs which are added to B1 and whose weights are updated to zero	Arcs whose weights are updated to infinity
	Weight of path	Node		
1	-115	6	(1,2), (2,3), (3,6)	(5,6)
2	-80	7	(3,7)	(6,7)
3	-5	5	(3,5)	----

Phase 3. The acyclic graph still consists of three negatively weighted arcs. Therefore, add their minimum weight rooted paths to B2. At this point, the sum of the original weights of the arcs in B2 becomes negative. Therefore, transfer to B1 all the arcs of B2, that is arcs (2,4),(4,8),(8,10),(8,11),(8,12). Stop this phase here, as none of the arcs in the acyclic graph has a negative weight (Figure 3d). The computations of this phase are summarized in Table 2.

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),(3,6),(3,7),(3,5),(2,4),(4,8),(8,10),(8,11),(8,12),(8,9).Theweight of this arborescence is -200.

Table 2. Summary of computations in Phase 3 of the example for H

Iteration No.	Min- weight rooted path in the graph		Arcs which are added to B2 and whose weights are updated to zero	Arcs whose weights are updated to infinity	Sum of original weights of arcs of B2
	Weight of path	Node			
1	1	10	(2,4), (4,8), (8,10)	---	1
2	-1	11	(8,11)	---	0
3	-1	12	(8,12)	----	-1

6. Lagrangian Relaxation of the MRA Problem

The motivation for studying a Lagrangian relaxation, LMRA, for MRA problem 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. 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 jth node. We can formulate our first Lagrangian problem as:

$$\underset{U\geq 0}{Max}\quad \underset{Y}{Min}\left[\sum_{(i,j)\in A} W(i,j)Y(i,j)+\sum_{j=2}^{N} U(j)\{(\sum_{i\in P(j)} Y(i,j))-1\}\right] \tag{9}$$

s.t.(3), (4) *and* (5)

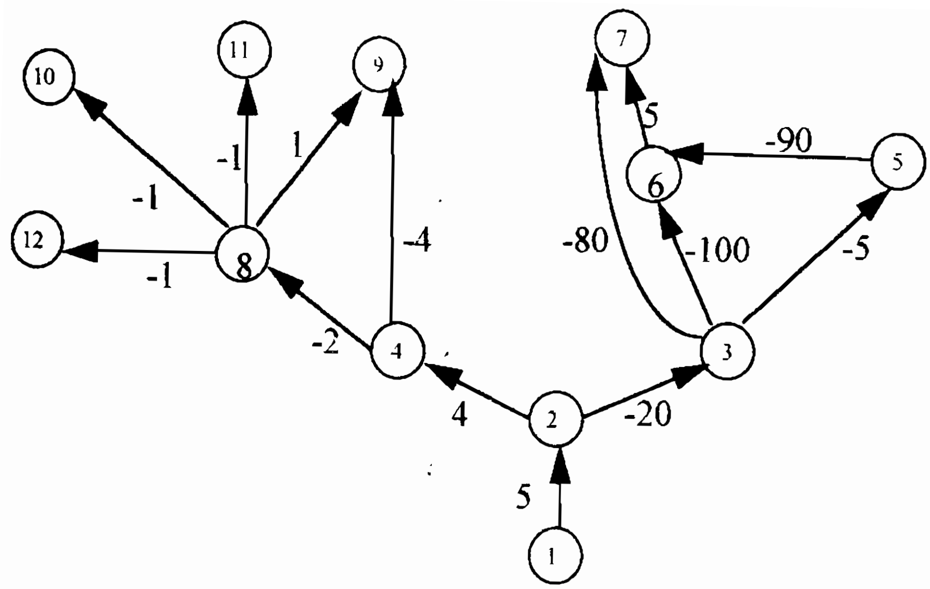


Figure 3a. G for the example of section 5

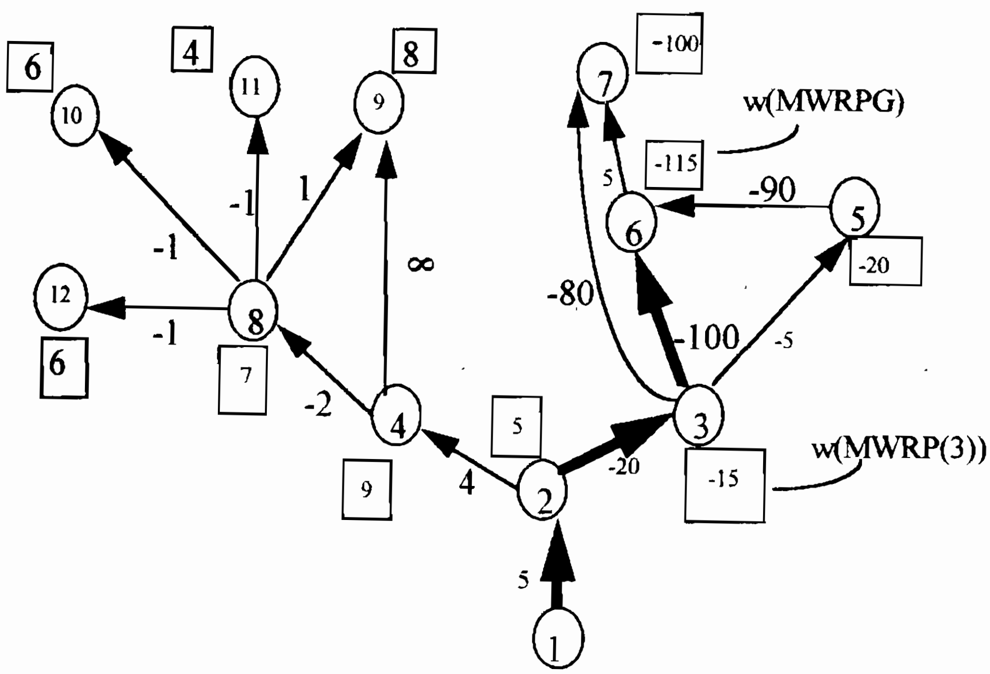


Figure 3b.MWRPG computation in iteration 1 of phase 2

(The number in square bracket near node k is w(MWRP(k)

The thick arrows indicate MWRPG)

Figure 3. Illustration of H

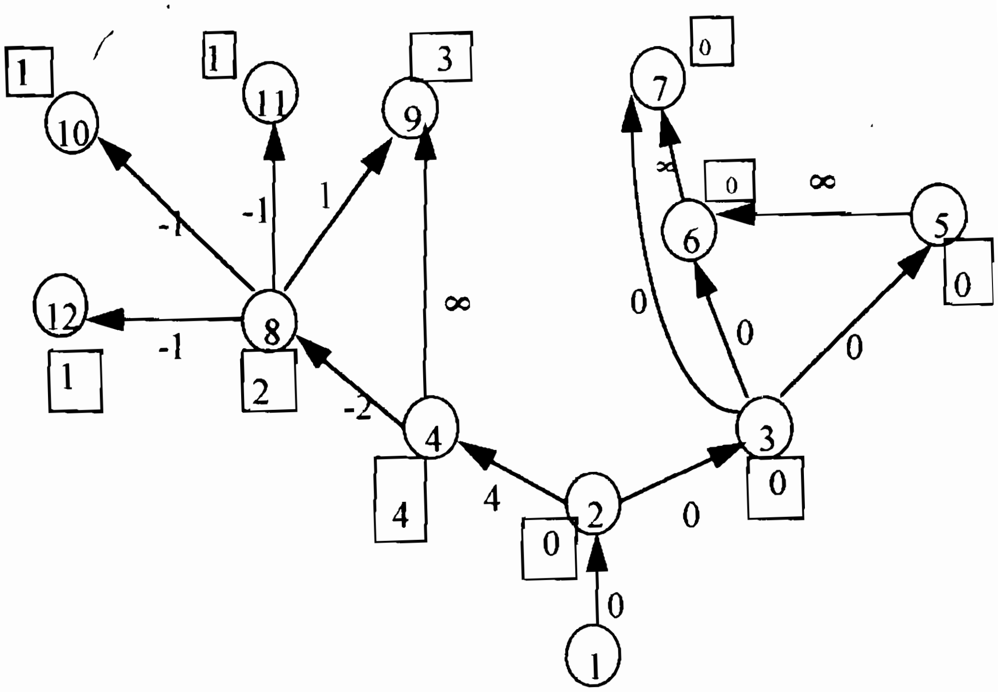


Figure 3c. G at the end of phase 2 .

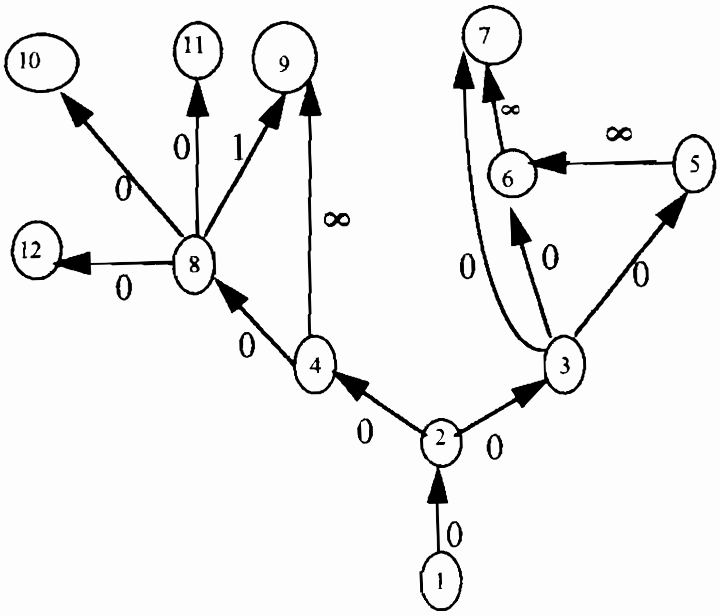


Figure 3d. G at the end of phase 3 .

Figure 3 (Contd.)

The constraints of the above problem resemble those of MRA problem with weights on nodes [4]. However, the objective function (13) does not exhibit any known form.

2. The constraint set chosen for relaxation here is the set of connectivity constraints (3). For this, consider the multipliers $U = \{U(i,j) \mid (i,j) \in A,\ i > 1\}$, each U(i,j) being associated with the connectivity constraint corresponding to arc (i,j). Therefore, the second Lagrangian problem (LMRA) can be written as

LMRA

$$\underset{U\geq 0}{Max}\ \underset{Y}{Min}\{\left(\sum_{(i,j)\in A} W(i,j)Y(i,j)\right)+\sum_{(i,j)\in A, i>1} U(i,j)\left(Y(i,j)-\sum_{k\in P(i)} Y(k,i)\right)\} \qquad (10)$$

$$s.t. (2),\ (4)\ and\ (5)$$

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 (2) are in the form of Knapsack constraints. Hence, for a given set of multipliers U, LMRA can be solved by splitting it into several independent knapsack problems, one for each j whose S(j) is not null, j > 1. It is useful to rewrite the objective function (10), by regrouping the terms, in the following form:

$$\underset{U\geq 0}{Max}\ \underset{Y}{Min}\left\{\sum_{(i,j)\in A} Y(i,j)\left[W(i,j)+U(i,j)-\sum_{k\in S(j)} U(j,k)\right]\right\} \qquad (11)$$

Example on the formulation of LMRA. To illustrate the separability of LMRA into several knapsack problems, let us consider an example using the graph shown in Figure 4. We give below the Lagrangian problem LMRA for the graph of Figure 4.

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

$$Z=\underset{U}{Max}\ Z(U)=\underset{U}{Max}\ \{Z_1(U)+Z_2(U)+Z_3(U)\} \qquad (12)$$

where:

$$Z_1(U) = Min\ \{Y(1,2)[W(1,2)+U(1,2)-U(2,3)-U(2,4)+ Y(2,3)[W(2,3)+U(2,3)-U(3,4)-U(3,5)]\} \qquad (13)$$

$$s.t.\ Y(1,2)=1,\ and\ Y(2,3) \in \{0,1\} \qquad (14)$$

$$Z_2(U) = \text{Min}\ \{Y(3,4)[W(3,4)+U(3,4)-U(4,5)]+ Y(2,4)[W(2,4)+U(2,4)-U(4,5)]\} \quad (15)$$

$$\text{s.t. } Y(3,4)+Y(2,4) \leq 1 \quad (16)$$

$$Y(3,4),\ Y(2,4) \in \{0,1\} \quad (17)$$

$$Z_3(U) = \text{Min}\ \{Y(3,5)[W(3,5)+U(3,5)]+Y(4,5)[W(4,5)+U(4,5)]\} \quad (18)$$

$$\text{s.t. } Y(3,5)+Y(4,5) \leq 1 \quad (19)$$

$$Y(3,5), Y(4,5) \in \{0,1\} \quad (20)$$

It can be seen that the problem (13)-(14) can be easily solved and that (15)-(17) and (18)-(20) can be solved by picking the variable with least negative coefficient in the objective function; no variable is picked if all coefficients are positive. By substituting the given U's and the values of Y(i,j)'s obtained as above in the objective function (11) we obtain a lower bound for the MRA problem.

7. Lagrangian Heuristic L for MRA Problem

We propose a heuristic solution for MRA problem based on LMRA discussed above. At each Lagrangian iteration, for a given set of U's, we obtain a lower bound for MRA problem 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 heuristic H of section 5, we can construct a feasible solution to MRA problem with the above Y(i,j)'s set at 1. Hence, at every Lagrangian iteration we generate a lower bound as well as an upper bound for MRA problem. This heuristic procedure is described below:

Step 1 (Initialize): With $\theta_{in} = \{(1,2)\}$ obtain an initial upper bound by using heuristic H described in section 5. Initialize the Lagrange multipliers U(i,j) = 0, for $(i,j) \in A$. Initialize the best lower bound Z_{LB} to $-\infty$

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

Step 3 (Find upper bound): Step 2 fixes some of the Y(i,j)'s at one. Let this set of Y(i,j)'s constitute θ_{in}. With this θ_{in} apply heuristic H to generate a feasible solution and an upper bound for w(MRA). Update the best upper bound, Z^{UB}, if necessary .

Step 4 (Apply stopping rule): If any of the following three conditions are satisfied, stop: 1. $Z^{UB} = Z_{LB} + \epsilon$; 2. The iteration count has exceeded a given limit; 3. The best lower bound converges to a given value. If none of the above conditions is satisfied, go to step 5.

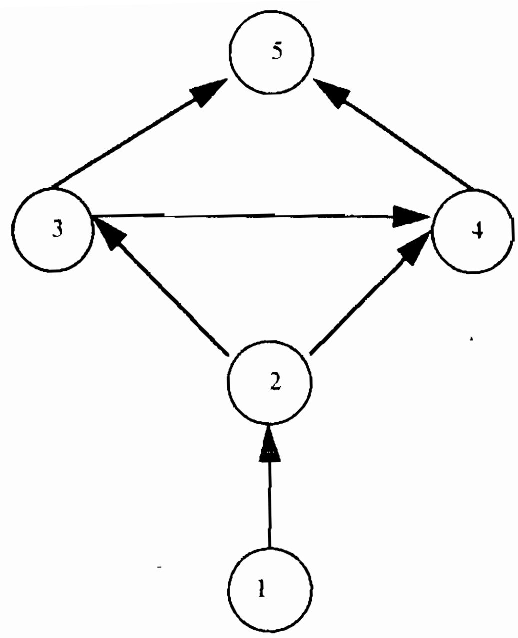


Figure 4. Graph to illustrate LMRA

Step 5 (Update 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 [2]. Let $Y^*(i,j)$ be the optimal solution to the Lagrangian problem LMRA. Let k be the iteration number. We then compute the subgradients NU(i,j) for U(i,j) as

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

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 equal to 1 and halve the value every 8 iterations if the lower bound does not improve.

8. Computational Results.

Heuristic L was tested on ten different graph structures, with the number of nodes ranging from ten to fifty-five, in intervals of five. From each structure, we generated five test problems by changing the arc weights, thus getting fifty test problems in all. The graph structures and arc weights were generated randomly, while ensuring that trivial solutions do not occur. Figure 5 shows one of the graphs used in the testing.

In each of the problems, the arc weights were integers. For the stopping rule, ϵ was given as 0.99, and the maximum number of iterations permitted was 500.

The programs were written in TURBO C and run under DOS on a Pentium PC with 100 MHZ clock, and 16 MB RAM.

The following statistics were collected for each problem while running L (Table 3): best upper bound value (BESTUB), best lower bound value (BESTLB), the iteration number when the BESTUB was found (ITERUB), the iteration number when the BESTLB was found (ITERLB), value of the feasible solution found by H initially (INITHEUR), iteration number when L stopped (TOTITER), number of times the upper bound was updated (NUPDT), execution time in milli seconds taken by L (TIMEMSEC).

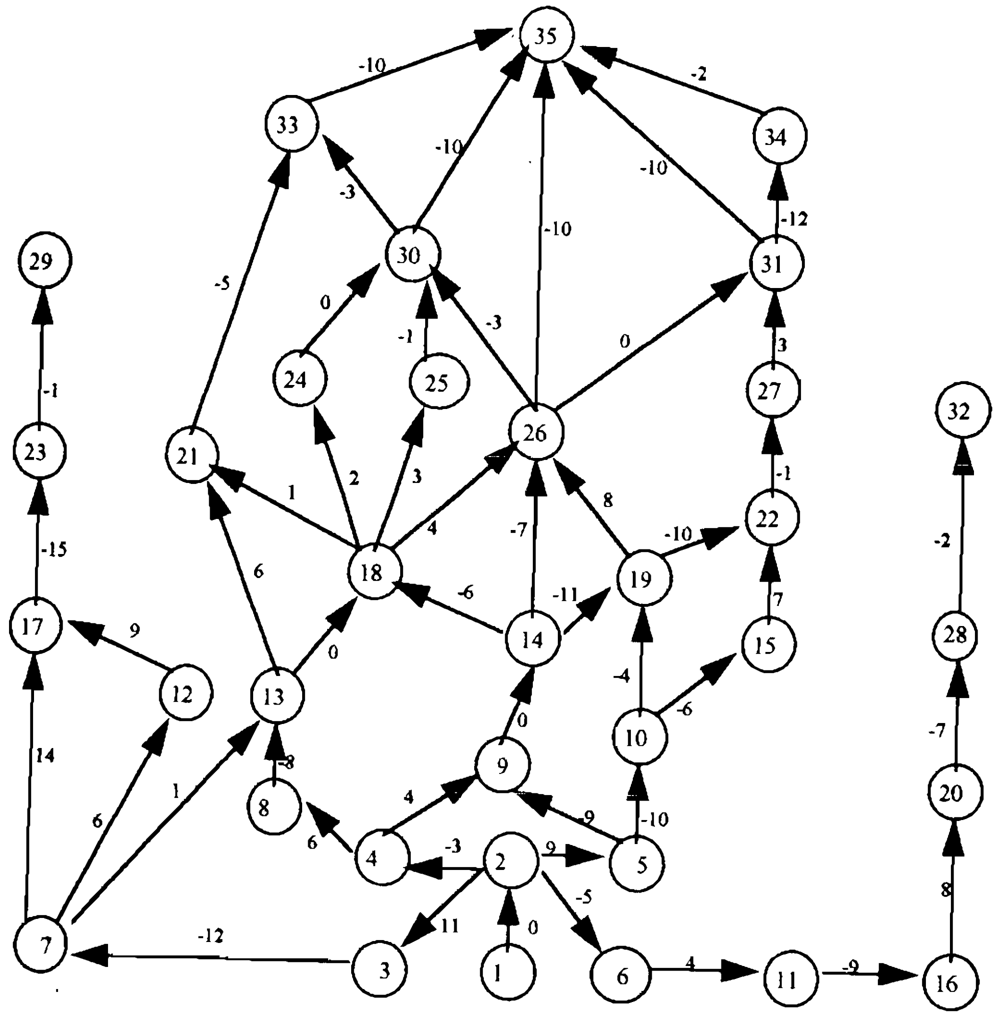


Figure 5. One of the test graphs, T35A

When judged by its ability to reach an optimal solution, the performance of L in the testing was quite satisfactory. In eighty-six percent of the test problems, L found the optimal solution.

As mentioned before, H could be used as a stand alone heuristic for MRA. The above testing revealed the efficacy of the solution given by H, when no arcs are forced to be in the solution. The value of this solution of H in L was recorded as INITHEUR. In forty percent of the test problems, INITHEUR itself turned out to be the BESTUB; out of these, in seventy five percent of the cases, the INITHEUR was confirmed to be optimum, as BESTLB and BESTUB were equal in those cases. Without testing L, in which H was an embedded step, the above insight could not have been achieved.

One of the purposes of L was to improve the initial solution given by H. In sixty four percent of the test problems, INITHEUR was improved at least once. The average percent improvement in INITHEUR as measured by 100*Abs(INITHEUR-BESTUB)/INITHEUR over all the test problems was 6.53. For the test problems with at least one improvement in INITHEUR, the above measure was 10.21.

During the tests, we also measured the duality gap, to find how close the BESTLB found by L was to the optimal solution. The average value of the duality gap, calculated as Abs(100*(BESTUB-BESTLB)/BESTUB), over all the test problems, was 2.74 percent. For the test problems with duality gap greater than zero percent, this average was 12.46 percent.

As expected, the time of execution per iteration tended to increase with the number of nodes, although the increase was not monotonic: it varied from a fraction of a milli second for problems upto 20 nodes, to six milli seconds for problems with 55 nodes. However, the total number of iterations taken by a problem did not necessarily depend on its size. The complexity of a problem, obviously, was governed not only by its size but also by the graph structure: in the 35 node problems, four out of five exceeded the iteration limit of 500, whereas all the fifty node problems could be solved to optimality within the same iteration limit.

9. Concluding Remarks

Our major contribution in this paper was to present a new, computationally hard problem and to propose heuristics to solve it. The computational results of these heuristics are promising enough to warrant further study of this problem and its solution methods. More over, the proposed heuristics add to the existing repertoire of solution methods for UL. Further, suitable enumeration techniques could be studied to close the non- zero duality gap, if any, resulting from an application of L on MRA problem.

Table 3. Summary of computational results.

Problem title	N	BESTUB	BESTLB	ITERUB	ITERLB	INITHEUR	TOTITER	NUPDT	TIME MSEC
T10A	10	-15	-15.6982	1	16	-15	16	0	0
T10B	10	-18	-18.9800	1	64	-18	64	0	60
T10C	10	-23	-23.8739	1	15	-23	15	0	0
T10D	10	-18	-18.9309	1	16	-18	16	0	11
T10E	10	-16	-16.9890	1	14	-16	14	0	0
T15A	15	-94	-97.5373	1	143	-94	500	0	220
T15B	15	-116	-122.0026	1	306	-116	500	0	270
T15C	15	-243	-247.0023	1	218	-243	500	0	270
T15D	15	-86	-114.0212	1	171	-86	500	0	280
T15E	15	-152	-152.9943	1	90	-152	90	0	50
T20A	20	-46	-46.9102	1	19	-42	19	1	0
T20B	20	-53	-53.9815	1	26	-53	26	0	60
T20C	20	-49	-49.9359	1	24	-49	24	0	0
T20D	20	-55	-55.9934	1	25	-55	25	2	0
T20E	20	-34	-34.9499	20	21	-32	21	1	0
T25A	25	-95	-95.8689	1	22	-95	22	0	0
T25B	25	-99	-99.7528	2	10	-90	10	1	60
T25C	25	-103	-103.9339	2	10	-101	10	1	0
T25D	25	-108	-108.8514	2	7	-96	7	1	0
T25E	25	-102	-102.9167	2	17	-101	17	1	0
T30A	30	-410	-412.5039	1	198	-410	500	0	770
T30B	30	-427	-427.8703	1	9	-427	9	0	50
T30C	30	-435	-435.8993	4	5	-432	5	2	60
T30D	30	-435	-435.9390	2	8	-435	8	1	60
T30E	30	-29	-41.5145	2	156	-27	500	1	820
T35a	35	-103	-103.9903	2	87	-101	87	1	220
T35B	35	-89	-89.9352	1	44	-89	44	0	110
T35C	35	-97	-97.9674	1	46	-97	46	0	110
T35D	35	-97	-97.9460	1	46	-97	46	0	110
T35E	35	-96	-96.0506	48	47	-95	48	1	110
T40A	40	-328	-328.9909	12	68	-292	68	4	220
T40B	40	-338	-338.9822	16	76	-294	76	3	270
T40C	40	-349	-349.9822	16	76	-301	76	3	330
T40D	40	-342	-342.9822	16	76	-291	76	3	280
T40E	40	-378	-378.9836	16	76	-326	76	3	270
T45A	45	-368	-368.9741	12	71	-332	71	4	330
T45B	45	-409	-409.9912	15	125	-363	125	4	500
T45C	45	-443	-443.8936	31	30	-387	31	3	110
T45D	45	-449	-448.9999	46	45	-440	46	3	160
T45E	45	-427	-427.0436	33	32	-398	33	4	160
T50A	50	-344	-344.9848	13	76	-306	76	5	380
T50B	50	-340	-340.9776	13	74	-301	74	6	380
T50C	50	-350	-350.9996	17	86	-299	86	3	4401
T50D	50	-356	-356.9926	17	82	-320	82	4	440
T50E	50	-345	-345.9964	17	82	-302	82	3	440
T55A	55	-392	-392.9908	12	75	-328	83	5	440
T55B	55	-356	-356.9581	14	81	-305	81	7	500
T55C	55	-364	-364.9807	18	82	-321	82	7	500
T55D	55	-366	-367.0064	56	264	-299	500	9	3020
T55E	55	-396	-396.9578	16	88	-359	88	5	550

References

[1] G. Cornuejols, M.L. Fisher and G.L. Nemhauser, On the uncapacitated plant location problem, *Annals of Discrete Mathematics*, Vol 1, 1977, pp 163-177.

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

[3] V. Venkata Rao, Optimal Lot Sizing in Acyclic Multi-Stage Production Systems, Ph.D. Dissertation, Georgia Institute of Technology, University Microfilms Order No. 8124 295, 1981.

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

[5] V. Venkata Rao, and R. Sridharan, *The Minimum Weight Rooted Arborescence Problem: Weights on Arcs Case*, Working Paper No. 1030, Indian Institute of Management, Ahmedabad, May 1992.

Appendix-1

Summary of Notation

Symbol	Meaning
G	Rooted acyclic graph with weights on arcs.
N	Number of nodes in G.
A	The set of arcs in G.
(i,j)	Arc starting at node i and ending at node j.
W(i,j)	The weight of arc (i,j).
Wu(i,j)	The updated weight of arc (i,j) in heuristic H.
MRA	Minimum weight rooted arborescence
UL	Uncapacitated plant location problem.
MWRP(j)	Minimum weight rooted path for node j.
MWRPG	The rooted path which has the smallest weight among the MWRP(i) for all i in G, i > 1.
w(MRA)	Weight of minimum weight rooted arborescence in G.
w(MWRP(j))	Weight of MWRP(j).
w(MWRPG)	Weight of MWRPG.
H	Upper bound heuristic for MRA problem.
LMRA	Lagrarian relaxation of the MRA problem.
U(i,j)	Lagrange multiplier used for the connectivity constraint corresponding to arc (i,j) in formulating LMRA.
Y(i,j)	0-1 variable for arc (i,j) in the 0-1 integer program for MRA problem.
Θ_{in}	The set of arcs required to be present in the rooted arborescence.
S(j)	The set of immediate successors of node j in G.
P(j)	The set of immediate predecessors of node j in G.

Appendix-2

Pseudo-code for heuristic H

```
Begin
        Initialize
        Apply Phase 1
        Apply Phase 2
        Apply Phase 3
        Apply Phase 4
        Output the list of arcs in B1 as the RA
End
```

Pseudocode for "Initialize"

```
Begin
        Input N and A of G
        Input W(i,j) for each (i,j)
        in A
        B1 ← φ
        B2 ← φ
        For each (i,j) in G
          Wu(i,j) ← W(i,j)
        EndFor
        w(MWRPG) ← +∞
        Θin ← {(1,2)}
End
```

Pseudocode for "Apply Phase 1"

```
Begin
    For each (i,j) ∈ Θ in
        For each (k,j) ∈ A, k ≠ i
            W(k,j) ← ∞
            Wu(k,j) ← ∞
        EndFor
    EndFor
End
```

Pseudocode for "Apply Phase 2"

```
Begin
Find MWRPG
While w(MWRPG) < 0
    For each (i,j) which is in MWRPG and not in B1
        Append (i,j) to B1
        Wu(i,j) ← 0
        For each (k,j) ∈ A , k ≠ i
            Wu(k,j) ← ∞
        EndFor
    EndFor
    Find  MWRPG
EndWhile
End
```

Pseudocode for "Apply Phase 3"

```
Begin
SUMB2 ← 0
While there exists an (i,j) in G such that Wu(i,j) < 0
    Find MWRPG
    For each (i,j) which is in MWRPG and not in B1 nor in B2
        SUMB2 ← SUMB2 + W(i,j)
        Wu(i,j) ← 0
        For each (k,j) ∈ A and k ≠ i
            Wu(k,j) ← ∞
        EndFor
        Append (i,j) to B2
    EndFor
    If SUMB2 < 0 Then
        Transfer all arcs of B2 into B1
        B2 ← φ
        SUMB2 ← 0
    EndIf

EndWhile
End
```

Pseudocode for "Apply Phase 4"

```
Begin
While there exists an (i,j) such that (i,j) is in Θin and not in B1
    minnode ← 1
    minwt ← ∞
    For each (i,j) ∈ Θin and not((i,j) ∈ B1)
        Find MWRP(j)
        If w(MWRP(j)) <  minwt Then
            minnode ← j
            minwt ← w(MWRP(j))
        EndIf
    For each (i,j) which is in MWRP(minnode) and not in B1
        Append (i,j) to B1
        Wu(i,j) ← 0
        For each (k,j) ∈ A and k ≠ i
            Wu(k,j) ← ∞
        EndFor
    EndFor
EndWhile
End
```

Appendix-2 (contd...)

Pseudocode for "Find MWRPG"

```
Begin
w(MWRPG) ← ∞
IMWRPG ← 1
w(MWRP(1)) ← 0
For j = 2 to N
    w(MWRP(j)) ← ∞
    For each i ∈ P(j)
        If w(MWRP(i)) + W(i,j) < w(MWRP(j)) Then
            w(MWRP(j)) ← w(MWRP(i)) + W(i,j)
        EndIf
    EndFor
    If w(MWRP(j)) < w(MWRPG) Then
        w(MWRPG) ← w(MWRP(j))
        IMWRPG ← j
    EndIf
EndFor
End
```