Solving the Assignment Problem by Relaxation

MING S. HUNG
Kent State University, Kent, Ohio

WALTER O. ROM
Cleveland State University, Cleveland, Ohio

(Received June 1978; accepted February 1980)

This paper presents a new algorithm for solving the assignment problem. The algorithm is based on a scheme of relaxing the given problem into a series of simple network flow (transportation) problems for each of which an optimal solution can be easily obtained. The algorithm is thus seen to be able to take advantage of the nice properties in both the primal and the dual approaches for the assignment problem. The computational bound for the algorithm is shown to be $O(n^3)$ and the average computation time is better than most of the specialized assignment algorithms.

DEGENERACY is a frequently occurring phenomenon in network flow problems. Cunningham [6] showed that the simplex method will not cycle at a degenerate extreme point if only "strongly feasible bases" are constructed. For the assignment problem a strongly feasible basis is equivalent to the "alternating path basis" used by Barr, Glover and Klingman [3].

We show that the alternating path basis can also be used to develop an efficient dual algorithm for the assignment problem. The assignment problem is well known:

(AP) Min $\sum_i \sum_j c_{ij} x_{ij}$

subject to

$$\sum_i x_{ij} = 1, j \in J \tag{1}$$

$$\sum_j x_{ij} = 1, i \in I \tag{2}$$

$$x_{ij} \geq 0, i \in I \text{ and } j \in J \tag{3}$$

where $I = J = \{1, 2, \cdots, n\}$.

The network for the assignment problem consists of a set of origin nodes (denoted as 0_i, $i \in I$), a set of destination nodes ($D_j, j \in J$) and arcs (i, j) directed from an origin to a destination. This bipartite structure and the $0 - 1$ solution at an extreme point of the assignment polytope makes (AP) easier to solve than most of the network flow problems.

Operations Research
Vol. 28, No. 4, July–August 1980

0030-364X/80/2804-0969 $01.25

Most of the specialized algorithms for the assignment problem are of the dual nature, in that at every intermediate step the algorithm produces a solution satisfying only some of the constraints in (1) and (2). Kuhn's Hungarian method [12] is perhaps the best known in this category. Other algorithms include that of Desler and Hakimi [7], Dinic and Kronrod [9] and, as Lawler [13] indicates, Edmonds and Karp [10].

Since every basis at an extreme point of the assignment polytope consists of $n - 1$ degenerate variables, primal algorithms like the simplex method usually require many degenerate pivots. Balinski and Gomory [1] presented a modified simplex algorithm to reduce the number of degenerate pivots. Barr, Glover and Klingman [3], achieved a greater decrease in degenerate pivots by using alternating path bases. However, the computational experience in [3] shows that, still, more than 90% of the pivots in the problems tested are degenerate.

Our algorithm, like the dual algorithm, solves AP by relaxing some of the constraints in (1) and (2). But the method for enforcing violated constraints uses the alternating path bases in the relaxed problem. Furthermore, an optimal solution is always available for the relaxed problem and the optimal alternating path basis is constructed not by simplex pivoting but by shortest path algorithms.

I. BASIS STRUCTURE OF THE ASSIGNMENT PROBLEM

It is known that a basis in a network flow problem can be represented by a spanning tree. An extreme point of the assignment polytope consists of n positive variables, each having a value of 1. The graphic representation of an extreme point has n disjoint pairs of nodes, one an origin and one a destination, which are connected by the positive variable. A basis at the extreme point is then formed by adding $n - 1$ degenerate arcs such that the n pairs are linked into a tree. (From which one can show that there are $2^{n-1}n^{n-2}$ bases at an extreme point. See Balinski and Russakoff [2].)

Denote a basis tree by a generic symbol T. A destination node in T is designated as the "root." As in [6], let $R_{ij}(T)$ denote the node set of the tree still connected to the root if arc (i, j) were removed from T. An arc (i, j) is called "directed toward the root" if $j \in R_{ij}(T)$ and "directed away from the root" otherwise. An alternating path (short as A-P) basis specifies that every degenerate arc on T be directed away from the root. See Figure 1 for an example of an A-P basis, in which D_2 is the root.

If the variables x_{ij} are put in an $n \times n$ matrix, an extreme point of the assignment polytope is a $0 - 1$ matrix with exactly one 1 on every row and every column. An A-P basis then requires one degenerate variable in every column, except for the root column.

The name "alternating path basis" arises from the fact that a path

from a node to the root of the tree consists of arcs which are alternatively positive and degenerate, and are alternatively directed toward and away from the root.

The dual of problem AP is

$$\text{(D)} \qquad \max \quad \sum_i u_i + \sum_j v_j$$

subject to

$$u_i + v_j \leq c_{ij},\ i \in I, j \in J$$

u_i, v_j unrestricted in sign.

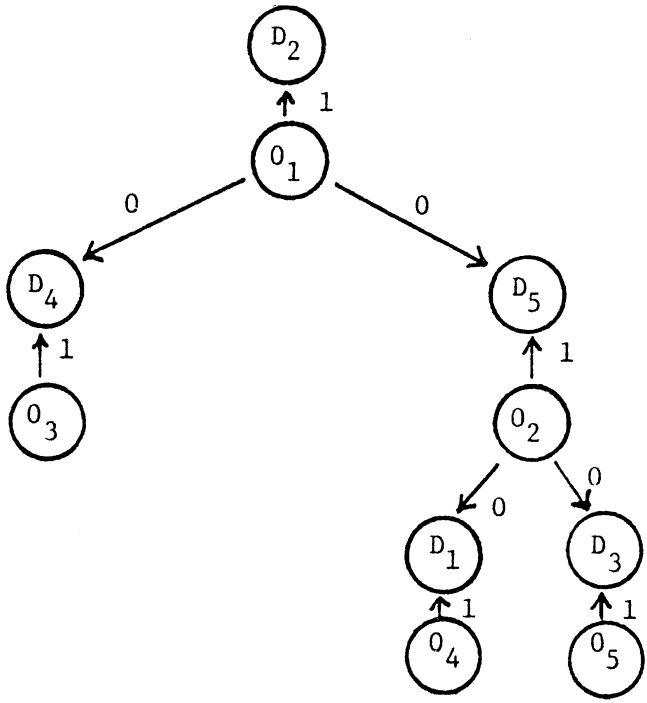


Figure 1. An alternating path basis tree.

A dual solution satisfying the complementary slackness condition will satisfy $u_i + v_j = c_{ij}$ if x_{ij} is a basic variable. From now on, we will always assume $v_r = 0$ where r is the root. Define the "distance" of an A-P to be the total cost of degenerate arcs minus the total cost of the positive arcs. Then it is straightforward to check that v_j is the distance of the A-P from destination j to the root, whereas u_i is the negative of the distance of the A-P from origin i.

REMARK 1 [3]. *Let T_1, T_2 be A-P bases at the same extreme point and $T_2 = T_1 \cup\{(i^*, j^*)\}\backslash\{(\bar{\imath}, \bar{\jmath})\}$. That is T_2 is obtained from T_1 by a*

degenerate pivot for which (i^*, j^*) *is the entering arc and* $(\bar{\imath}, \bar{\jmath})$ *is the leaving arc. Suppose* $\delta = c_{i^*j^*} - u_{i^*} - v_{j^*}$. *Then*

$$v_j(T_2) = \begin{cases} v_j(T_1), \text{ if } j \in R_{\bar{\imath}\bar{\jmath}}(T_1) = R_{i^*j^*}(T_2) \\ v_j(T_1) + \delta, j \notin R_{\bar{\imath}\bar{\jmath}}(T_1) \end{cases}$$

and

$$u_i(T_2) = \begin{cases} u_i(T_1), \text{ if } i \in R_{\bar{\imath}\bar{\jmath}}(T_1) \\ u_i - \delta, i \notin R_{\bar{\imath}\bar{\jmath}}(T_1). \end{cases}$$

Since an entering arc (i^*, j^*) is chosen only when $\delta < 0$, the above remark indicates that a degenerate pivot results in a strict decrease for some v_j's and a strict increase for some u_i's.

Our algorithm is based on constructing the optimal A-P basis at the optimal extreme point for a sequence of relaxed problems. A basis is optimal if the corresponding dual solution is feasible.

THEOREM 1. *Let* T_k, $k = 1, \cdots$ *be A-P bases at the optimal extreme point, each having the same root r.* T_{k^*} *is optimal if* $v_j(T_{k^*}) = \min v_j(T_k)$ *for each* $j \in T$ *and all k. That is,* T_{k^*} *is optimal if every* $v_j(T_{k^*})$ *is the distance of the shortest A-P from destination node j to the root r.*

Proof. Follows directly from Remark 1.

One can therefore construct the optimal basis by finding the shortest A-P from every column to the root. To find the shortest A-P, one may first change the sign of the cost of the positive arcs. Or one may first subtract the cost of the positive arc from the cost of every arc on the same row (emanating from the same origin) since every A-P going through the row must go through the positive arc. Then one can use a shortest path algorithm that allows negative costs. Such algorithms all have a computational bound of $0(m^3)$ for a dense network of m nodes (see Christofides [5] or Lawler [13]). Dijkstra's algorithm [8] has a bound of $0(m^2)$ but does not allow negative costs. The following algorithm relaxes AP so that Dijkstra's algorithm can be used.

II. THE RELAXATION ALGORITHM

1. Relaxed Problem

The idea behind the new algorithm is to relax the column constraints (1) so that an optimal (and basic) solution to the relaxed problem can be found easily. Then a violated constraint is enforced and a new optimal (and basic) solution is constructed. At every intermediate step of the algorithm the column index set is partitioned into three subsets, J_0, J_1 and J_2. J_1 is the set of column constraints in (1) whose left-hand-sides are equal to 1, i.e., the satisfied constraints. J_0 is the set of column constraints

whose left-hand-sides are equal to 0 and J_2 is that whose left-hand-sides are at least 2. Specifically, we relax the original problem (AP) to

$$(\mathrm{AP}_r) \qquad \text{Min } \sum_i \sum_j c_{ij}x_{ij}$$

subject to

$$\sum_i x_{ij} = \begin{cases} 0, j \in J_0 \\ 1, j \in j_1 \\ b_j, j \in J_2 \end{cases} \tag{1'}$$

and (2), (3), where $b_j \geq 2$ integer and $\sum_{j \in J_2} b_j + |J_1| = n$. $|S|$ denotes the number of elements in set S.

(AP_r) is a network flow problem and in which every basic solution consists of n positive variables and $|J_1| + |J_2| - 1$ degenerate basic variables. Thus an A-P basis is applicable. Since the dual problem of AP_r has the same constraints as D, Remark 1 and Theorem 1 also remain applicable.

Furthermore, an A-P basis makes enforcing a constraint in J_0 extremely easy. Suppose a column $j^* \in J_2$ is chosen as the root, and an alternating path basis for AP_r is available (see Figure 2). If a column constraint $j \in J_0$ is to be enforced and it is determined that $x_{i^*j} = 1$, i.e., Row i^* is to be assigned to Column j, there is a unique path from Row i^* to the root. An A-P from a row node to the root alternates between a positive arc and a degenerate arc. Therefore, by letting the positive arcs in the path become degenerate arcs and degenerate arcs become positive arcs, a feasible solution is obtained for the new relaxed problem in which b_{j^*} is reduced by 1 and column constraint j is enforced.

It may be easier to see the above argument by the example on Figure 2, particularly the basis matrix. Suppose Column 4 is to be enforced and Row 4 is to be assigned to it, i.e., $(i^*, j) = (4,4)$. Then an alternating path to the root (which is Column 1) is easily identified as {(4,4), (4,2), (3,2), (3,1)} and a change of variable values along the path will make possible the assignment of Row 4 to Column 4. Note that the alternating path for enforcing column constraints $j \in J_0$ need not always end at the root column. It can end at any column with more than one assigned cell, that is, a column in the set J_2. For example, if Row 1 is to be assigned to Row 4, the alternating path can be just {(1,4), (1,3)} so that, loosely speaking, the assignment of Row 1 is passed from Column 3 to Column 4.

2. Algorithm

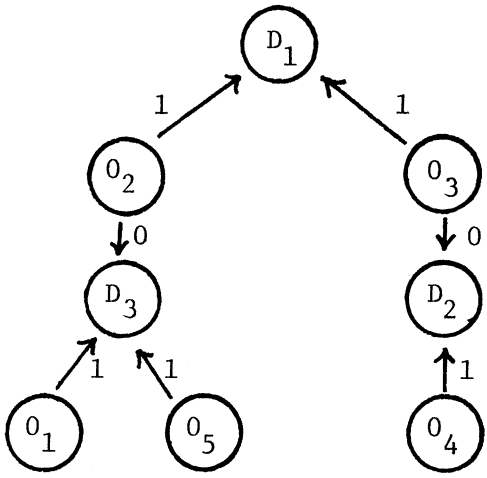
Step 0. Initialization.

Assign every row to a column with the minimum cost in that row. Therefore, column index set J is partitioned into J_0, J_1 and J_2 and an optimal solution to the relaxed problem is clearly

obtained by setting the variables of the assigned cells to 1. Set $v_j = 0$ for $j \in J$.

Step 1. Construction of basis.

This step determines the optimal A-P basis and the corresponding dual solution.



The Basis in Matrix Form

Origin \ Destination	1	2	3	4	5
1			1		
2	1		0		
3	1	0			
4		1			
5			1		

Figure 2. An alternating path basis for the relaxed problem (AP_r).

1.1. Select a column $j^* \in J_2$ as the root, and let $v_j^* = 0$ (if it was not zero) and $u_i = c_{ij^*}$ if $x_{ij^*} = 1$. At this moment, the basis tree consists of the root and the row nodes assigned to the root.

1.2. For each column $j \in J_1 \cup J_2$ not in the tree, compute the minimum distance of attaching it to the tree. That is, for

each such j, compute $k_j = \text{Min}_i(c_{ij} - u_i)$ over rows i in the tree.

1.3 Select a column, say $\hat{j}$, with the minimum $k_j - v_j$ and attach it to the tree. (This statement anticipates that v_j may not be zero after the initial cycle.) Suppose $k_{\hat{j}}$ occurs in Row $r(\hat{j})$, then $x_{r(\hat{j}),\hat{j}}$ is chosen as the degenerate basic variable for Column $\hat{j}$. Determine the appropriate dual variables. That is, let $v_{\hat{j}} = k_{\hat{j}}$ and $u_i = c_{i\hat{j}} - v_{\hat{j}}$ if $x_{i\hat{j}} = 1$. If every column of J_1 or J_2 is in the tree go to Step 2. Otherwise, return to Step 1.2.

Step 2. Enforcing violated constraints.

In this step, a column constraint $\bar{j} \in J_0$ is enforced.

2.1. Select a column $\bar{j} \in J_0$ and compute $v_{\bar{j}} = \text{Min}_{i \in I}(c_{i\bar{j}} - u_i)$. Suppose $v_{\bar{j}}$ occurs on Row i^*. Then trace the alternating path from Row i^* to a column in J_2.

2.2. Change every degenerate arc in the path to a positive arc and every positive arc to a degenerate arc. Assign row i^* to column $\bar{j}$, i.e., let $x_{i^*\bar{j}} = 1$. Constraint $\bar{j}$ of (1) is now enforced, so remove $\bar{j}$ from J_0 and place it in set J_1. Note that the number of assignments (b_j) of the last column in the alternating path is now reduced by 1. If the new number is 1, move the column from set J_2 to J_1. If $J_0 = \emptyset$, stop because the optimal solution to AP is found. Otherwise, a cycle is complete (or if an optimal basis to AP is desired), return to Step 1.

3. Example

Consider the following 7 × 7 example adapted from Christofides [5, p. 376].

	1	2	3	4	5	6	7	u
1	13	21	20	12	[8]	26	22	3
2	12*	36	25	41	40	11	[4]	4
3	35	32	[13]	36	26	21	13√	13
4	34	54	[7]	8√	12√	22	11	7
5	21	[6]	45	18	24	34	12	−4
6	42	19√	39	15	[14]	16	28	9
7	16	34	38	[3]	34	40	22	2
v		10^4	0	1^2	5^3		0^1	

Cycle 1

CYCLE 1:

Step 0. The assignments are on the boxed (□) cells which yield the initial partition of $J_0 = \{1,6\}$, J_1 $\{2,4,7\}$ and $J_2 = \{3,5\}$. $v_j = 0$ for all j.

Step 1. Suppose the root is Column 3. The dual variables are determined for Rows 3 and 4. Iterations through Steps 1.2 and 1.3 yield the necessary degenerate basic variables which are checked (√), and the dual variables. The superscript on the v values indicates the order in which they are determined.

Step 2. Suppose Column 1 is to be enforced. The pivot element is (2,1) because $v_1 = 8 = \min_i(c_{i1} - u_i)$. The alternating path from Row 2 is {(2,7), (3,7), (3,3)}. Therefore, Row 2 is now assigned to Column 1, Row 3 to Column 7.

	1	2	3	4	5	6	7	u
1	13√	21	20	12	□8	26	22	8
2	□12	36	25	41	40	11	4√	7
3	35	32	13√	36	26	21	□13	16
4	34	54	□7	8√	12	22	11	10
5	21	□6	45	18	24	34	12	1
6	42	19√	39	15	□14	16*	28	14
7	16	34	38	□3	34	40	22	5
v	5^2	5^1	-3^4	-2^5	0		-3^3	

Cycle 2

CYCLE 2:

Step 1. The optimal basis is shown above. The root is now Column 5.

Step 2. Column 6 is enforced. $v_6 = 2$ and the alternating path is {(6,6), (6,5)}. So Row 6 is assigned to Column 6. The algorithm terminates with an optimal solution of $x_{15} = x_{21} = x_{37} = x_{43} = x_{52} = x_{66} = x_{74} = 1$ and all other $x_{ij} = 0$. The optimal value is 65.

4. Proof of Algorithm

To prove that the preceding algorithm does produce an optimal solution to AP in a finite number of cycles requires establishing the following:

a. Step 1 yields an optimal A-P basis to the relaxed problem.

b. Step 2 results in an optimal solution to the new relaxed problem.

We shall prove (b) first. Let G_1 be the network of the old relaxed

problem AP_r. G_1 has arcs (i, j) where $i \in I$ and $j \in J_1 \cup J_2$. Assume that Step 1 is correct and therefore there is an optimal A-P basis T_1, which is a tree on G_1. Then

$$c_{ij} - u_i - v_j \begin{cases} = 0 \text{ for } (i,j) \in T_1 \\ \geq 0 \quad\quad (i,j) \notin T_1. \end{cases} \tag{4}$$

Now a column constraint $\sum_i x_{i\bar{j}} = 1$ for a $\bar{j} \in J_0$ is added to AP_r. Let G_2 be the network of the new relaxed problem. Then G_2 has arcs (i, j) where $i \in I, j \in J_1 \cup J_2 \cup \{\bar{j}\}$. Let $T_2 = T_1 \cup \{i^*, \bar{j}\}$ (T_2 also has the additional node $\bar{j}$) where $c_{i^*\bar{j}} - u_i^* = \min_{i \in I} (c_{i\bar{j}} - u_i)$, as indicated in Step 2.1. Then after the augmentation carried out in Step 2.2, T_2 is an optimal basis (but not on A-P basis) for the new relaxed problem since relations (4) hold true for T_2 also. Thus (b) is proven.

To prove (a), recall that an optimal A-P basis can be constructed by a shortest path algorithm after the cost of every positive variable is subtracted from the cost of all variables in the same row. When Step 1 is entered from Step 0, the tasks performed in Step 0 ensure that this subtraction will leave every cost nonnegative.

When Step 1 is entered from Step 2, there is an optimal, but not A-P, basis T_2. Since relations (4) hold for T_2, if u_i and v_j are subtracted from cost c_{ij}, every new cost is nonnegative.

So in either case, an $0(n^2)$ algorithm like Dijkstra's can be used to construct a new optimal A-P basis. And Dijkstra's is the algorithm we used in Step 1.

In Step 1.2, $k_j - v_j$ is the temporary label (current shortest distance) from column $j \in G_2$ to the root. In Step 1.3, the smallest temporary label is fixed (see [8] or [5, Chap. 8] for more details). Therefore it is concluded that Step 1 produces an optimal A-P basis.

Finally, since there are initially at most $m - 1$ columns in J_0 and every cycle of the algorithm reduces the number by 1, the algorithm is finite and terminates with an optimal solution. Furthermore, since every cycle has a bound of $0(n^2)$, the following is clear.

THEOREM 2. *The algorithm has a computational bound of* $0(n^3)$.

The discussions so far also establish that the algorithm can be used to solve the following problem with the same worst-case bound.

$$\begin{aligned} &\text{Min } \textstyle\sum_i \sum_j c_{ij} x_{ij} \\ &\textstyle\sum_j x_{ij} = 1,\ i \in I = \{1, \cdots, n\} \\ &\textstyle\sum_i x_{ij} = b_j,\ j \in J = \{1, \cdots, m\} \\ &\quad x_{ij} \geq 0 \text{ for all } (i,j) \end{aligned}$$

where $m \leq n$, $b_j > 0$ and integer, and $\sum_j b_j = n$.

III. COMPUTATIONAL CONSIDERATIONS AND EXPERIENCE

There are a few additional comments we would like to mention regarding the implementation of the algorithm.

1. In the initial step (Step 0) one may subtract constants from rows and columns of the cost matrix in order to minimize the number of columns in J_0 initially. For the previous example, if this had been performed, then Row 1 would be assigned to Column 1 and Cycle 1 would thus be avoided.
2. In Step 2 of the algorithm, one may enforce more than one column constraint at a time, as long as the alternating paths do not overlap. For example, in Cycle 1 of the previous example, one may enforce Columns 1 and 6 because the alternating path from Column 6 is {(6,6), (6,5)}. Cycle 2 is thus saved.
3. One need not construct the optimal basis from the very beginning every time Step 1 is entered. If the root remains unchanged, then all the paths still connected to the root will remain optimal.
4. There may be more than one pivot element for column $\bar{j}$ when it is enforced. This happens when there are ties in the minimum of $c_{i\bar{j}} - u_i$. Thus in every cycle one may investigate several possible combinations of pivots among the J_0 columns in order to bring in as many columns as possible.
5. One may modify the algorithm somewhat by considering all J_2 columns as a collective root. This will simplify the bookkeeping chores because dual variables need to be computed only for columns in J_1. In other words, one can further relax constraint (1′) in (AP_r) into

$$\sum_i x_{ij} = \begin{cases} 0, j \in J_0 \\ 1, j \in J_1 \end{cases}$$

$$\sum_{j \in J_2} \sum_i x_{ij} = \sum_{j \in J_2} b_j.$$

The code used for the following computational experience incorporates all the above modifications except those in (4).

Computational Experience

The proposed algorithm has been extensively tested against other algorithms for the assignment problem. The competing algorithms selected are Silver's [14] code of the Hungarian method [12], Brown and Obee's [4] code of the graph theoretic algorithm of Desler and Hakimi [7], and the recent primal method of Barr, Glover and Klingman [3]. The first two, which will be denoted by "Hungarian" and "Graphic" respectively, are directly, statement for statement, translated from their original language ALGOL into FORTRAN.

Our code of Barr et. al.'s algorithm, which will be denoted by "Primal" was coded from the description in [3]. In our code, we have used most of the work-saving techniques the authors suggested [3] and some tactics which were not explicit in their paper:

a. Their algorithm uses a row as the root. To determine the entering basic variable, we search for the most violated dual constraint (i.e., the most negative $c_{ij} - u_i - v_j$) in a row, starting from the row of the last pivot.
b. The initial solution is chosen as suggested in [3]. That is, assign a row to the least cost column which does not have an assignment.

TABLE I

COMPUTATIONAL RESULTS ON RANDOM PROBLEMS: SET 1[a]

n	Relaxation		Relative Efficiency[b]		
	Time[c]	No. of cycles[d]	Hungarian	Graphic	Primal
20	0.032	1.85	0.509	0.636	0.548
30	0.089	2.65	0.451	0.569	0.534
40	0.174	2.90	0.540	0.588	0.549
50	0.297	3.65	0.591	0.568	0.532
60	0.454	3.85	0.656	0.587	0.545
70	0.663	4.20	0.689	0.552	0.560
80	1.002	4.80	0.818	0.624	0.596
90	1.163	4.45	0.760	0.563	0.502
100	1.412	4.35	0.770	0.516	0.497

[a] Cost coefficients range from 1 to 100. There are 20 problems in each problem size (n).
[b] (Average time of relaxation)/(average time of algorithm x).
[c] CPU seconds on an IBM 370/158, per problem over 3 runs.
[d] Average per problem.

After that, we use our Step 1 to locate the degenerate basic variables. Certainly this will not yield a basis requiring no degenerate pivots, but we found it tends to reduce the number of degenerate pivots.

All the programs were compiled and run under IBM FORTRAN IV G level compiler and on an IBM 370-158 multiprogramming system. The algorithms were put into subroutines and were run two at a time back to back. Since significant discrepancies in the time of even the same program on the same problem were detected, we resorted to using the ratio between the times of the two algorithms as a common measure. The ratios reported in Tables I–III are the time of our algorithm divided by that of the competing algorithm. Thus a ratio of 0.5 under algorithm X would mean that our algorithm was twice as fast as X.

Three sets of test problems were used. In Set 1, the results of which are shown in Table I, the cost coefficients are randomly generated between 1 and 100. In Set 2, shown in Table II, the cost coefficients are randomly chosen between 1 and 1,000; and in Set 3, between 1 and 10,000.

TABLE II

COMPUTATIONAL RESULTS ON RANDOM PROBLEMS: SET 2[a]

n	Relaxation		Relative Efficiency		
	Time	No. of cycles	Hungarian	Graphic	Primal
20	0.036	2.15	0.342	0.595	0.524
30	0.087	2.65	0.245	0.511	0.484
40	0.176	3.05	0.229	0.497	0.541
50	0.301	3.65	0.199	0.480	0.511
60	0.446	3.70	0.204	0.509	0.484
70	0.662	4.15	0.185	0.531	0.492
80	0.890	4.35	0.199	0.464	0.479
90	1.150	4.45	0.186	0.433	0.467
100	1.465	4.50	0.195	0.447	0.463

[a] Cost coefficients range from 1 to 1000. See other footnotes to Table I.

The reason for this design is due to Brown and Obee [4] who found in their computational study that algorithms can behave differently when the range of cost coefficients varies. From our computational experience, the Hungarian method is most sensitive to this variation of data. The Graphic method is also sensitive to data but the effect is less pronounced.

The tables also show the number of cycles required by our algorithm. This is the number of times an optimal A-P basis to AP_r must be constructed. This is the most time consuming part of the algorithm and the reason the algorithm performs so well is clearly due to the fact that relatively few cycles are required. In fact, not only are the average number of cycles low, but also, among the 540 test problems, only 1 problem required 8 cycles and only 2 required 7. The increase in the number of cycles also seems to grow rather slowly with the increase in problem size.

IV. CONCLUDING REMARKS

After the completion of the computational study for this paper, we distributed the computer code of our relaxation algorithm to a few

TABLE III

COMPUTATIONAL RESULTS ON RANDOM PROBLEMS: SET 3[a]

n	Relaxation		Relative Efficiency		
	Time	No. of cycles	Hungarian	Graphic	Primal
20	0.037	2.25	0.316	0.610	0.559
30	0.099	2.95	0.230	0.623	0.570
40	0.186	3.20	0.210	0.558	0.543
50	0.306	3.55	0.162	0.508	0.524
60	0.473	4.00	0.145	0.483	0.515
70	0.640	3.95	0.129	0.498	0.490
80	0.906	4.45	0.112	0.479	0.467
90	1.239	4.75	0.104	0.415	0.506
100	1.523	4.85	0.102	0.437	0.482

[a] Cost coefficients range from 1 to 10000. See other footnotes to Table I.

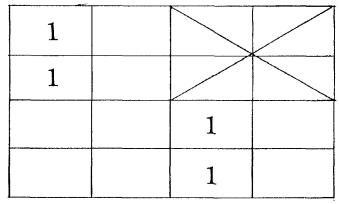
researchers for further computational experiments. Glover [11] compared our algorithm with the original code of Barr, Glover and Klingman's and confirmed that our algorithm is about twice as fast as the latter on totally dense problems. But since our code was not designed for the sparse problem and theirs was, the relative efficiency of our algorithm decreases as the problem gets sparser.

We also learned of the existence of two other $0(n^3)$ assignment algorithms after the completion of this paper. One is that of Dinic and Kronrod [9] and the other an implementation of Edmonds and Karp's [10] algorithm for general network flow problems. Dinic and Kronrod's algorithm relaxes AP to our AP_r. But the method of enforcing a constraint in J_0 is based on subtracting constants from the cost matrix so that a cost in a column of J_0 may become a minimum on a certain row and hence that row may be assigned to that column in J_0.

The general algorithm of Edmonds and Karp can be specialized in several ways to solve the assignment problem. But we think in all ways the original problem AP is relaxed in both column and row constraints and the relaxed problem is a smaller assignment problem. Every cycle then enforces both a column and a row constraints. In this regard, the alternating path basis can also be used for the relaxed problem.

The efficiency of our algorithm is derived from being able to enforce several constraints in a cycle, as seen from the small number of cycles in our computational experience. We do not know of any good way of accomplishing this in either Dinic and Kronrod's or Edmonds and Karp's algorithm.

Finally, we would like to mention an interesting aspect of using A-P basis for the sparse problems. It is possible that one may not be able to construct an A-P basis for a sparse problem, either AP or AP_r. But in this situation Cunningham [6] shows that the problem is decomposable into smaller, separate problems. The following solution matrix for AP_r is an example. Crossed-out cells mean nonexistent arcs.



If one designates Column 1 as the root, then Column 3 cannot be connected to the root according to the A-P basis. Then this assignment problem can be decomposed into two parts, one part made up of Rows 1, 2 and Columns 1 and 2, and the other part of Rows 3, 4 and Columns 3 and 4.

REFERENCES

1. M. L. BALINSKI, AND R. E. GOMORY, "A Primal Method for the Assignment and Transportation Problems," *Mgmt. Sci.* **10,** 578–593 (1964).
2. M. L. BALINSKI, AND A. RUSSAKOFF, "The Assignment Polytope," *SIAM Rev.* **16,** 516–525 (1974).
3. R. S. BARR, F. GLOVER AND D. KLINGMAN, "The Alternating Path Algorithm for Assignment Problems," *Math. Prog.* **13,** 1–13 (1977).
4. J. R. BROWN, AND R. W. OBEE, "Efficient Assignment Algorithms," Working paper, Graduate School of Business Administration, Kent State University, 1976.
5. N. CHRISTOFIDES, *Graph Theory*, Academic Press, New York, 1976.
6. W. CUNNINGHAM, "A Network Simplex Method," *Math. Prog.* **11,** 105–116 (1976).
7. J. F. DESLER, AND S. L. HAKIMI, "A Graph-Theoretic Approach to a Class of Integer Programming Problems," *Opns. Res.* **17,** 1017–1033 (1969).
8. E. W. DIJKSTRA, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik* **1,** 269 (1969).
9. E. A. DINIC, AND M. A. KRONROD, "An Algorithm for the Solution of the Assignment Problem," *Soviet Math. Dokl.* **10,** 1324–1326 (1969).
10. J. EDMONDS, AND R. M. KARP, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. Assoc. Comput. Mach.* **19,** 248–264 (1972).
11. F. GLOVER, private communication, 1979.
12. H. W. KUHN, "The Hungarian Method for the Assignment Problem," *Naval Res. Log. Quart.* **12,** 83–97 (1955).
13. E. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart & Winston, New York, 1976.
14. R. SILVER, "An Algorithm for the Assignment Problem," *Commun. Assoc. Comput. Mach.* **3,** 603–606 (1960).