AD-A035 920 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 12/2
A NEW ALTERNATING BASIS ALGORITHM FOR SEMI-ASSIGNMENT NETWORKS.(U)
JAN 77 R S BARR, F GLOVER, D KLINGMAN N00014-75-C-0616

UNCLASSIFIED CCS-264 NL

1 OF 1
AD-A
035 920

END
DATE
FILMED
3-25-77
NTIS

A NEW ALTERNATING BASIS ALGORITHM FOR SEMI-ASSIGNMENT NETWORKS

TEXAS UNIVERSITY AT AUSTIN

JANUARY 1977

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Center for Cybernetic Studies
The University of Texas

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

A New Alternating Basis Algorithm for Semi-Assignment Networks

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Richard S. Barr
Fred Glover
Darwin Klingman

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
January 1977	33	19

8a. CONTRACT OR GRANT NO.

N00014-75-C-0616; 0569 and
N00014-76-C-0383

b. PROJECT NO.

NR047-021

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

Center for Cybernetic Studies
Research Report CCS 264

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Office of Naval Research (Code 434)
Washington, D.C.

13. ABSTRACT

During the early 1970's, highly efficient special purpose computer codes were developed for solving capacitated transshipment problems based on primal extreme point algorithms. Computational comparisons disclosed that these codes were substantially superior to the best non-extreme point codes both in terms of computer time and memory requirements on all types of net work problems. More recently, non-extreme point codes have been developed that are highly specialized for specific types of uncapacitated bipartite problems--notably assignment and semi-assignment problems. Comparisons of these problem specific codes with the earlier general purpose codes have raised the question of whether non-extreme point methods may not be superior to primal extreme point methods in application to these special network problem types. Consequently, the purpose of this paper is twofold: (i) to develop a new primal extreme point algorithm which is specifically designed to take advantage of the bipartite and boolean flow structures of assignment and semi-assignment problems, and (ii) to conduct an unbiased comparison of the alternative algorithmic approaches for solving assignment and semi-assignment problems.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Networks						
Assignment problem						
Transportation						
Manpower Planning						
Linear Programming						

Research Report CCS 264

A NEW ALTERNATING BASIS ALGORITHM FOR SEMI-ASSIGNMENT NETWORKS

by

R. S. Barr*
F. Glover **
D. Klingman***

January 1977

*Assistant Professor of Management Science and Computers, Southern Methodist University, School of Business Administration, Dallas, TX 75275.

**Professor of Management Science, University of Colorado, Boulder, Colorado 80302.

***Professor of Operations Research, Statistics, and Computer Sciences and Director of Computer Science Research, BEB-608, University of Texas at Austin, Austin, TX 78712.

Presented at the Bicentennial Conference on Mathematical Programming, Nov. 30-Dec. 1, 1976 at the National Bureau of Standards, Gaithersburg, Maryland.

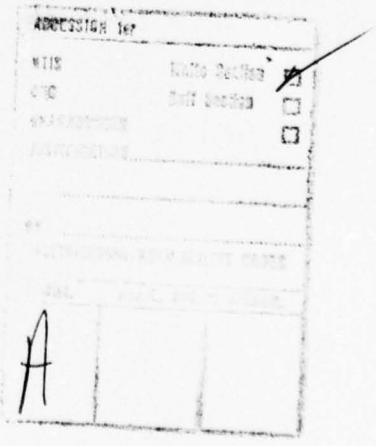
This research was partly supported by ONR Contract N00014-76-C-0383 with Decision Analysis and Research Institute and by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-C-0569 with the Center for Cybernetic Studies, The University of Texas.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 512
The University of Texas
Austin, Texas 78712
(512) 471-1821

ABSTRACT

During the early 1970's, highly efficient special purpose computer codes were developed for solving capacitated transshipment problems based on primal extreme point algorithms. Computational comparisons disclosed that these codes were substantially superior to the best non-extreme point codes both in terms of computer time and memory requirements on all types of network problems. More recently, non-extreme point codes have been developed that are highly specialized for specific types of uncapacitated bipartite problems--notably assignment and semi-assignment problems. Comparisons of these problem specific codes with the earlier general purpose codes have raised the question of whether non-extreme point methods may not be superior to primal extreme point methods in application to these special network problem types. Consequently, the purpose of this paper is twofold: (i) to develop a new primal extreme point algorithm which is specifically designed to take advantage of the bipartite and boolean flow structures of assignment and semi-assignment problems, and (ii) to conduct an unbiased comparison of the alternative algorithmic approaches for solving assignment and semi-assignment problems.



1. INTRODUCTION

The semi-assignment problem is a bipartite network problem whose demand constraints are the same as those of an assignment problem and whose supply constraints are the same as those of a transportation problem. (The roles of supplies and demands in the problem definition can be interchanged.) In the midst of the dramatic advances in network solution technology since 1969 (see, e.g., [1, 2, 5, 11, 14]) this important member of the network family has received scant attention. Falling midway between the classical assignment and classical transportation problems in its generality, it was bypassed both by those who studied the ultra-specialized assignment structures and those who studied the more general bipartite transportation structures.

The neglect of the semi-assignment problem is especially ironic in view of the fact that it occupies one of the singularly important niches in the network hierarchy. The "assignment half" captures the ubiquitous multiple choice structures of capital budgeting and planning problems as well as the special ordered set constraints of mixed integer and combinatorial programming. The "transportation half" captures arbitrary upper and lower bounds on disjoint sums of variables, and therefore can provide valid relaxations for any mixed integer program with embedded multiple choice and special ordered set structures. Still more directly, the semi-assignment structure appears in large scale scheduling and

planning problems from real world settings. For example, applications of manpower planning, scheduling, project planning, and a variety of other practical problems contain embedded semi-assignment problems.

Consequently, the purposes of this paper are to develop a new extreme point algorithm (called the alternating basis (AB) algorithm) which is specifically designed to take advantage of the bipartite and boolean flow structures of assignment and semi-assignment problems, and further, to conduct a comparison of the alternative algorithmic approaches using codes designed for solving these problems.

An important consideration for any method specialized to these types of problems is degeneracy. Computational testing has shown that approximately 90 percent of the pivots of special purpose primal extreme point algorithms [2, 11, 13] are degenerate for assignment and semi-assignment problems with more than 1000 nodes. The primal extreme point algorithm presented in this paper is designed specifically for solving semi-assignment problems in a manner that both circumvents and exploits degeneracy. Its method for accomplishing this can be viewed as an extension of the approach presented in [4] and a specialization of the approach presented in [8]. One of the principal features of the algorithm is a strong form of convergence that limits the number of degenerate steps in a far more powerful way than achieved by "lexicographic improvement," as for example, in customary LP perturbation schemes.

Each basis examined by this algorithm is restricted to have a certain topology. We show that if a semi-assignment problem has an optimal solution, then an optimal solution can be found by considering only bases of this type. The major mathematical differences between the AB algorithm

and alternative primal extreme point ("simplex-based") methods are (1) the rules of the algorithm automatically (without search) assure that all bases have the special topological structure, and bypass all other bases normally given consideration; (2) the algorithm is finitely convergent without reliance upon external techniques (such as lexicography or perturbation); and (3) in certain cases nondegenerate basis exchanges may be recognized prior to finding the representation of an incoming arc. For these reasons, this algorithm has several computational advantages over the highly efficient special purpose simplex-based codes recently developed for solving network problems.

The AB algorithm also has unique computer implementation properties. Specifically, the data required to represent its bases are substantially less than that required for general simplex bases and thus consume less computer memory. In addition, the computational results in section 6 dramatically demonstrate the superior efficiency of the AB algorithm over other algorithms for solving assignment and semi-assignment problems.

2. BACKGROUND MATERIAL

An m x n semi-assignment problem may be defined as:

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}\, x_{ij}$$

subject to:

$$\sum_{j\in\{j:(i,j)\in A\}} x_{ij} = b_i, \quad i\in I = \{1, 2, \ldots, m\}$$

$$\sum_{i\in\{i:(i,j)\in A\}} x_{ij} = 1, \quad j\in J = \{1, 2, \ldots, n\}$$

$$x_{ij} \geq 0, \quad (i,j)\in A$$

where I is called the set of origin nodes, J is called the set of destination nodes, A is the set of arcs, and c_{ij} is the cost of shipping a unit from origin node i to destination node j.

The dual of the semi-assignment problem may be stated as:

$$\text{Maximize} \quad \sum_{i \in I} R_i b_i + \sum_{j \in J} K_j$$

subject to:

$$R_i + K_j \leq c_{ij}, \quad (i,j) \in A$$

where R_i and K_j are called the node potentials of the origin and destination nodes, respectively.

An understanding of the results of this paper relies on a familiarity with graphical interpretations of the semi-assignment problem and how the primal simplex method may be applied to this problem. While these ideas are relatively direct, they, unfortunately, are not succinctly itemized in any references and will be summarized in this section for completeness.

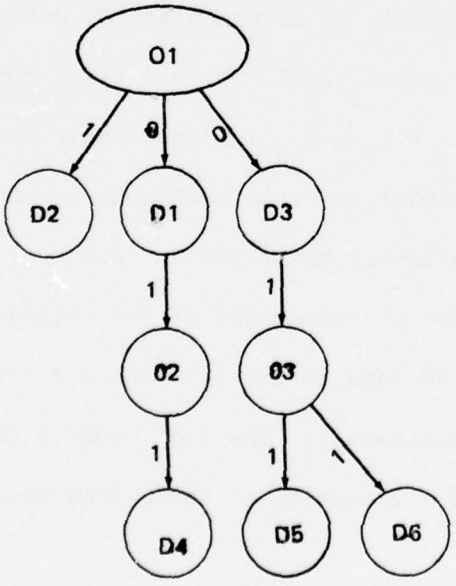
As the preceding terminology suggests, the semi-assignment problem may be represented as a bipartite graph consisting of a set of origin nodes with supplies b_i and a set of destination nodes with unit demands. Directed arcs from origin nodes to destination nodes accommodate the transmission of flow and incur a cost if flow exists. The objective is to determine a set of arc flows which satisfies the supply and demand requirements at minimum total cost.

The bases of the simplex method for solving an m x n semi-assignment problem correspond to spanning trees with m + n - 1 arcs. Exactly n of the basic arcs have an associated basic flow value of one and the other m - 1 arcs have a basic flow value of zero. Therefore, each basic solution is highly degenerate (i.e., contains a large number of zero flows). This often

causes the simplex method to examine several alternative bases for the same extreme point before moving to an adjacent extreme point.

In the graphical representation approach, the bases of the simplex method for semi-assignment problems are normally kept as rooted trees [5, 7, 12, 14, 19]. Conceptually, the root node may be thought of as the highest node in the tree with all of the other nodes hanging below it on directed paths leading downward from the root. Those nodes in the unique path from any given node i to the root are called the *ancestors* of node i, and the immediate ancestor of node i is called its *predecessor*.

Figure 1 illustrates a rooted basis tree, the predecessors of the nodes, and the basic flow values, for a 3 x 6 semi-assignment problem. Notationally, Oi denotes the ith origin node and Dj denotes the jth destination node. The number beside each link (arc) in the basis tree indicates the flow on this arc imparted by the basic solution. Predecessors of nodes are identified in the PREDECESSOR array. For example, as seen from this array, the predecessor of origin node 2 is destination node 1. The root of the tree is node O1 and has no predecessor.



NODE	PREDECESSOR
O1	None
O2	D1
O3	D3
D1	O1
D2	O1
D3	O1
D4	O2
D5	O3
D6	O3

Figure 1--Rooted Basis Tree

It is important to note that the direction of the links in Figure 1 correspond to the orientation induced by the predecessor ordering and do *not* necessarily correspond to the direction of the basis arcs in the semi-assignment problem. However, the direction of the basic arcs are known from the bipartite property of the semi-assignment problem; i.e., all problem arcs lead from origin nodes to destination nodes.

In subsequent sections the term *O-D link* and *D-O link* will be used to refer to links in a rooted basis tree that are directed from an origin node to a destination node and vice versa, according to the orientation imparted to the basic arcs by the predecessor indexing. For example, in Figure 1, O2-D4 is an O-D link while D1-O2 is a D-O link. Additionally, basic arcs with a flow of one or zero will be referred to as *1-links* and *0-links*, respectively.

The fundamental pivot step of the simplex method will now be briefly reviewed in the graphical setting. Assume that a feasible starting basis has been determined and is represented as a rooted tree. To evaluate the nonbasic arcs to determine whether any of them "price out" profitably, and therefore are candidates to enter the basis, it is necessary to determine values for the dual variables R_i, $i \in I$, and K_j, $j \in J$, which satisfy complementary slackness; i.e., which yield $R_i + K_j = c_{ij}$ for each basic arc.

There is a unique dual variable associated with each node in the basis tree. For this reason the dual variables--or their values--are often referred to as node potentials. Because of redundancy in the defining equations of the semi-assignment problem (and in network problems generally), one node potential may be specified arbitrarily. The root node is customarily selected for this purpose and assigned a potential of zero, whereupon the

potentials of the other nodes are immediately determined in a cascading fashion by moving down the tree and identifying the value for each node from its predecessor using the equation $R_i + K_j = c_{ij}$. Highly efficient labeling procedures for traversing the tree to initialize and update these node potential values are described in [5, 12, 14].

A feasible basic solution is optimal when all nonbasic arcs satisfy the dual feasibility condition $R_i + K_j \leq c_{ij}$. If the solution is not optimal, then an arc whose dual constraint is violated (i.e., for which $R_i + K_j > c_{ij}$) is selected to enter the basis. The arc to leave the basis is determined by: (1) finding the unique path in the basis tree, called the *basis equivalent path* which connects the two nodes of the entering arc, and (2) isolating a blocking arc in this path whose flow goes to zero ahead of (or at least as soon as) any others as a result of increasing the flow on the entering arc. In the basis equivalent path, all arcs an even number of links away from the entering arc are called *even* arcs, and all arcs an odd number of links away are called *odd* arcs. An increase in the flow of the incoming arc causes a corresponding increase in the flow of all even arcs and a corresponding decrease in the flow of all odd arcs. Thus, if an odd arc already has a 0 flow, then such an arc qualifies as a blocking arc and the incoming arc cannot be assigned a positive flow.

To illustrate, assume that the starting basis is the one given in Figure 1 and the entering arc is (O3,D4). The basis equivalent path for (O3,D4) is D4-O2-D1-O1-D3-O3. (Note that this path can be easily determined by tracing the chain of predecessors of O3 and D4 to their point of intersection [5, 10, 12].) As flow is increased on the entering arc, the flow on the odd arc (O1,D1) must be decreased. Since its flow is already zero, (O1,D1)

qualifies as a blocking arc so that when arc (O3,D4) is brought into the basis, arc (O1,D1) must be dropped. (There are no other blocking arcs in this case.) In addition, the pivot (or basis exchange) is degenerate since no flow change occurs.

Once the entering and leaving arcs are known, the basis exchange is completed simply by updating the flow values on the basis equivalent path and determining new node potentials for the new basis tree.

Only a subset of the node potentials change during a pivot and these can be updated rather than determined from scratch. This fact will play a crucial role in proving convergence of the algorithm to be developed.

To update the node potentials, assume that the nonbasic arc (p,q) is to enter into the basis and the basic arc (r,s) is to leave the basis. If arc (r,s) is deleted from the basis (before adding arc (p,q)), two subtrees are formed, each containing one of the two nodes of the incoming arc (p,q). Let K denote the subtree which does not contain the root node of the full basis. The node potentials for the new basis may be obtained [12] by updating only those potentials of the nodes in K, as follows. If p is in K, subtract $\delta = R_p + K_q - c_{pq} > 0$ from the potential of each origin node in K and add δ to the potential of each destination node in K. Otherwise, q is in K and $-\delta$ is used in the above operations.

3. ALTERNATING PATH BASIS DEFINITION AND PROPERTIES

The new alternating basis (AB) algorithm for semi-assignment problems developed in this section is similar to the primal simplex method as described above. Its major mathematical distinction is that it does not consider all feasible bases to be candidates for progressing to an optimal basis. That is, the simplex method allows a feasible spanning tree of any

structure whatsoever to be included in the set of those that are eligible for consideration as "improving bases" along the path to optimality. However, it will be shown that if a semi-assignment problem has an optimal solution then it also has an optimal solution with a unique basis tree structure, dubbed the alternating path (AP) structure. Furthermore, it will be shown that it is possible to restrict attention at *each step* to bases with this structure. In particular, the AB algorithm is a procedure designed to exploit the properties of the AP basis structure in a manner that substantially reduces the impact of degeneracy, the number of arithmetic operations, and the data storage locations required to solve the semi-assignment problem.

Definition: A rooted basis tree for a semi-assignment problem is an *alternating path (AP) basis* if:

1. *The root node is an origin node.*
2. All 1-links are O-D links.
3. All 0-links are D-O links.

An example of an AP basis is shown in Figure 2.

The "alternating path" designation is applied because every path from a node to any ancestor node in the tree, or vice versa, is an alternating path of 1-links and 0-links. Our attention will chiefly focus on paths from nodes to their ancestors (as would be traced along a succession of predecessors). A path that begins at an origin node and ends at an ancestor destination node will be called "0-AP" because it begins and ends with an 0-link. Similarily, a path that begins at a destination node and ends at an ancestor origin node will be called "1-AP" because it begins and ends with a 1-link.

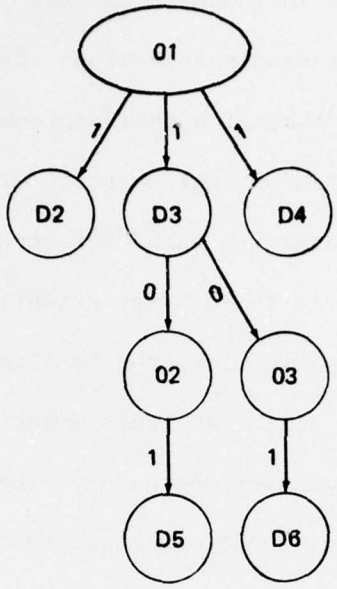


Figure 2--An AP basis for a 3 x 6 semi-assignment problem.

Remark 1: *The 1-links of any feasible semi-assignment solution* can be augmented by 0-links to create an AP basis (e.g., by adding 0-links from destination nodes to origin nodes in any fashion so that every origin node except the root node has exactly one entering 0-link.) Note if the arcs corresponding to the added 0-links do not exist in the particular semi-assignment problem then a large (big M) cost is assigned such links.

Remark 2: There are many semi-assignment bases for a given feasible solution that are not AP bases. (For example, any basis that has more than one 0-link incident to an origin node is not an AP basis regardless of the origin node chosen as the root. Figure 1 is an example of such a basis.)

Remark 3: An artificially feasible AP basis may always be constructed for an m x n semi-assignment problem by assuming that arcs exist from each origin node to all destination nodes where the nonadmissible (artificial)

arcs have a "big M" cost. The procedure is as follows:

Initially set $J' = \emptyset$, $B = \emptyset$, and $i = 1$. Go to step 1.

1. Let r be a destination node such that $c_{ir} = \min_{j \in J - J'} c_{ij}$

   Set $B = \{(i,r)\} \cup B$, $x_{ir} = 1$, $J' = J' \cup \{r\}$, and $b_i = b_i - 1$.

2. If $b_i \neq 0$ go to step 1. Otherwise, go to step 3.

3. If $i \neq m$, set $i = i + 1$ and go to step 1. Otherwise, set $i = 2$, $J' = \emptyset$ and go to step 4.

4. Let r be a destination node such that $c_{ir} = \min_{j \in J - J'} c_{ij}$ and $(i,r) \notin B$

   Set $J' = J' \cup \{j:(i,j) \in B\}$, $B = \{(i,r)\} \cup B$, and $x_{ir} = 0$.

5. If $i \neq m$, set $i = i + 1$ and go to step 4. Otherwise, go to step 6.

6. Using B, create a spanning tree rooted at node 1. The resulting spanning tree will be an AP basis.

Proof: The remark follows by construction.

Definition: Relative to any AP basis, a nonbasic arc is called a *downward arc* if it connects a destination node to an ancestor origin node, an *upward arc* if it connects an origin node to an ancestor destination node. An arc that connects an origin node and a destination node that do not have either of these ancestral relationships is called a *cross arc*. (Note that these are the only three possibilities for a nonbasic arc in a bipartite network.)

The next two remarks point out some important properties that can be exploited when applying the simplex method to an AP basis.

Remark 4: When the simplex method is applied to an AP basis, a pivot is nondegenerate if and only if the entering nonbasic arc is a downward arc.

Proof: The remark relies on the fact that a nondegenerate pivot causes the flows on the basis equivalent path to decrease and increase in a strictly alternating fashion to the odd and even links. The "if" part of the remark then follows by observing that a downward arc is 1-AP. The "only if" part of the remark follows from two observations, first that an upward arc is 0-AP, and second that a cross arc has a 0-link above the origin node incident to the entering arc (and this arc is contained in the basis equivalent path adjacent to one of the nodes of the entering arc).

Remark 5: When the simplex method is applied to an AP basis, the pivot can be carried out to give a new AP basis for any entering nonbasic arc simply by dropping the unique link in the basis equivalent path attached to the origin node of the entering arc.

Proof: The remark follows by observing that an AP basis results if a rooted tree is constructed with its root node equal to the root node of the old AP basis.

Alternating Basis (AB) Algorithm

On the basis of the preceding remarks, the rules of the AB algorithm can be stated in an extremely simple fashion.

1. Select any feasible AP basis for the semi-assignment problem (e.g., using Remark 3).

2. Successively apply the simplex pivot step keeping the root node fixed and picking the link to leave according to Remark 5.

By means of these rules, the foregoing observations imply that the AB algorithm will proceed through a sequence of AP bases, bypassing all other basis structures. Further, these remarks show that a "next" AP basis is always accessible to a given AP basis, so that the method will not be compelled to stop prematurely without being able to carry out a pivot before

the optimality (dual feasibility) criteria are satisfied. The issue to be resolved then, is whether the method may progress through a closed circle of AP bases without breaking out, and thus fail to converge. It will be shown that this cannot happen, and that, in fact, the AB algorithm is finitely converging without any reliance upon "external" techniques such as perturbation, as in the ordinary simplex method. Moreover, it will be shown that the form of convergence of the AB algorithm has a particularly strong character, in which origin node potentials and destination node potentials each change in a uniform direction throughout any sequence of degenerate pivots.

These results do not require any restrictions on the choice of the incoming variable. For example, it is not necessary to cull through pivot possibilities in an attempt to find degenerate pivot candidates. The following lemma and theorem validate these statements.

Lemma: A basis exchange with the AB algorithm gives rise to a new AP basis in which the new node potentials satisfy the following properties:

a) For a nondegenerate pivot: The changed origin node potentials strictly increase and the changed destination node potentials strictly decrease.

b) For a degenerate pivot: The changed origin node potentials strictly decrease and the changed destination node potentials strictly increase.

Proof: As already discussed, the node potential values that change may be restricted to those associated with the subtree K. By this procedure, if subtree K contains the origin node of the entering arc then all the origin node potentials in K are decreased and all destination node potentials in K are increased. The reverse is true if the destination node of the entering arc is in subtree K. The lemma then follows from Remarks 4 and 5, which

imply that subtree K always contains the destination node of the entering arc for a nondegenerate pivot and the origin node of the entering arc for a degenerate pivot.

Our main result may be stated as follows:

Theorem: The AB algorithm will obtain an optimal solution (or determine that the problem is infeasible) in a finite number of pivots, regardless of which dual infeasible arc is chosen to be the entering arc, and without any reliance on perturbation or lexicographic orderings.

Proof: It is sufficient to show that the number of degenerate pivots that occur between any two nondegenerate pivots must be finite. This follows from the second half of the lemma. Note that the node potential assigned to the root node never changes when the node potentials in subtree K are updated. Thus given the constant node potential for the root, the other node potentials are uniquely determined for each successive basis (regardless of the procedure by which they are generated), and the uniform increase of destination node potentials (for the potentials that change) implies that no basis can ever repeat during an uninterrupted sequence of degenerate pivots. This completes the proof.

4. COMPUTATIONAL CONSIDERATIONS

Some of the unique computational features of the AB method include:

a) It explicitly bypasses all "non-AP" basis solutions without requiring any embedded search procedure or computational tests.

b) It allows degenerate pivots to be recognized and performed without computing the representation of the entering arc. This can be accomplished by using the "cardinality function" of Srinivasan and Thompson [19] which indicates the number of nodes in the subtree of the basis tree below a given

node. In particular, following the labeling ideas recently proposed in [5], upward arcs and some cross arcs can be detected simply by comparing the cardinality function values of the nodes associated with the entering arc. That is, denote the cardinality function by f and the entering arc by (p,q). If $f(p) < f(q)$ then arc (p,q) is either an upward arc or a cross arc. In either case the pivot is degenerate and no flow updating is required. Remark 5, furthermore, directly specifies the link to leave the basis. Thus a degenerate pivot simply involves checking the cardinality function, inserting and deleting the appropriate links, and updating the node potential values.

c) Similar streamlining can be achieved for all other pivots. Specifically, if $f(q) < f(p)$ then the appropriate step is to find the first node z on the path from q to the root node such that $f(z) \geq f(p)$. If $z \neq p$ then arc (p,q) is a cross arc and thus the pivot may be executed as before. If $z = p$ then the arc (p,q) is a downward arc and the pivot is nondegenerate. Note that it is only in the case of a nondegenerate pivot that the entire basis equivalent path of the entering arc is traversed. Thus it is only in this case that the complete representation of the entering arc is computed. This is, of course, substantially different than for standard network methods.

Steps b and c above can also be accomplished by using the "distance function" proposed by Srinivasan and Thompson [19] which indicates the number of links in the basis tree between a given node and the root node. In particular, the distance function may be used in place of the cardinality function by reversing all the above inequalities.

d) Flow values on the basis links never have to be checked to determine the type of pivot. In fact, the structural property of an AP basis, whereby

all 1-links are O-D links, makes it unnecessary to store or update any flow values.

3) All of the above computational features may be further enhanced by observing that it is not necessary to store and update a "full" basis tree. That is, while the predecessors are required for each node, the cardinality function values and the "thread" pointers (commonly used to traverse basis subtrees [5, 7, 14] for node potential updates) need be kept for only the origin nodes. Using this observation, the AB algorithm's basis data storage requirements are roughly half that of the most efficient implementation involving specializations of the simplex method. Moreover, the compression may be used to greatly reduce the number of nodes traversed at each iteration when updating the potentials. By maintaining node potentials for only the origins and saving the unit costs for the arcs represented by the destinations and their predecessors, all data necessary for the pricing operation is available. The destination node potential K_j may be computed as needed using the complementary slackness relationship $K_j = c_{ij} - R_i$. This partial update of the node potentials was originally proposed and tested by Harris [16] for the simplex method as applied to rectangular transportation problems. (This approach has also been tested more recently for transshipment networks by Bradley, Brown, and Graves [7], who have confirmed Harris's findings of its practical merit.) Harris's proposal, however, differs from the above in that the thread pointer and potential update operation is eliminated for those destination nodes of the basis tree which have no descendents. Consequently, Harris's subtrees are twice as large as in our proposal and therefore involve twice the updating effort. In the AB algorithm, maintenance of only the thread pointers and node potentials associated with the origins does not degrade the efficiency of other parts of the algorithm. This is due to the unique structural properties of the AP basis.

5. DEVELOPMENT OF THE AB COMPUTER CODE BY SUBROUTINE

The computer code was written in FORTRAN IV, is an incore code, and was initially tested using the RUN compiler on a CDC 6600 with a maximum memory of 130,000 words. In this code, a semi-assignment problem with M origins, N destinations, and A arcs (without exploiting the word size of the machine) requires 4M + 2N + 2A + 5000 words. It would be possible by exploiting the fact that the costs, node numbers and node potentials are integer-valued, to store more than one per word and in this manner reduce these storage requirements. However, our purpose was to develop a code whose capabilities did not depend on the unique characteristics of a particular computer (i.e., word size, etc.). The obvious advantage of this approach is the ease with which it enables the code to be tested on different machines. Further, we used a "manilla" FORTRAN IV so that recoding to fit differing machine conventions would be minimized. Within these constraints, we tried to minimize our storage requirements, at the same time making sure the code could solve the "thoroughly general" semi-assignment problem. The code uses the predecessor [10], thread [14], and distance [19] functions to maintain and update the basis data.

A variety of start procedures could be used to find a starting AB basis. However, due to severe time pressure, we only implemented the start procedure of Remark 3.

An important factor influencing computational efficiency is the basis change criterion. The relevant tradeoffs for the basis change criterion involve time consumed in searching for a new arc to enter the basis and the number of pivots required to find an optimal solution (time per pivot versus total number of pivots). Computational testing [2, 7, 11, 13, 18, 19] has shown that the correct pivot criterion can reduce solution time by as

much as a factor of three. Unfortunately, we did not have time to test alternative pivot criteria. The code simply uses the row most negative rule, which was found to be the best in the studies [13, 19] for small problems. This criterion scans the arcs of each origin until it encounters the first origin containing a dual infeasibility and then selects the arc of this origin which violates dual feasibility by the largest amount to enter the basis.

The program consists of a main program and three subroutines. The total time spent in each subroutine was recorded by calling a Real Time Clock (accurate to a hundredth of a second) upon entering and leaving that subroutine. A count was also made of the number of nondegenerate and degenerate pivots performed. In the following section, we discuss total solution time (exclusive of input and output), the start time, the number of nondegenerate and degenerate pivots, the total pivot time, and the average pivot time (total pivot time divided by the number of pivots). This code will henceforth be referred to as SA-AB code, for semi-assignment AB algorithm code.

6. COMPUTATIONAL COMPARISON AND CODE REQUIREMENTS

6.1 Computational Comparison of Several Codes

Originally we planned to compare the best version of the SA-AB code (i.e., the code resulting after testing alternative start and pivot rules) with other codes which are based on other algorithms. Unfortunately, we did not have time to test alternative start and pivot procedures; thus, the following is a worst case comparison, that is, it compares an unrefined, "first cut" version of the SA-AB code with the best version of other codes.

The codes which we obtained for comparison include Bennington [6], BSRL, General Motors, SHARE, and SUPERK [3]. All of these codes are

variants of the out-of-kilter method except for Bennington [6]. In addition, the special purpose primal simplex codes ARC-II [5], PNET-I [11], and SUPERT-2 [2] were available for testing. Further, the special purpose dual simplex DNET [11] was available for testing.

The BSRL code was developed by T. Bray and C. Witzgall at the Boeing Scientific Research Laboratories, and the General Motors (GM) code is a modified version of a program originally developed by Rand Corporation.

All of these codes are in-core codes, i.e., the program and all of the problem data simultaneously reside in fast-access memory. They are all coded in FORTRAN and none of them (including the special purpose primal and dual simplex codes) have been tuned (optimized) for a particular compiler. It is important to note that all the codes except for SUPERT-2 and SA-AB codes are designed to solve capacitated transshipment problems and are not specifically designed to exploit the special structure of semi-assignment problems. Further, SUPERT-2 is designed to solve any uncapacitated transportation problem. All of the problems were solved on the CDC 6600 at the University of Texas Computation Center using the RUN compiler. The computer jobs were executed during periods when the machine load was approximately the same, and all solution times are exclusive of input and output; i.e., the total time spent solving the problem was recorded by calling a Real Time Clock upon starting to solve the problem and again when the solution was obtained.

In addition, an article by Hatch [15] compares a primal-dual code PD-AAL developed by Decision Systems Associates (DSA) against PNET-I on five assignment problems generated by NETGEN [17]. Since the DSA code is proprietary, we could not obtain a copy of it for comparison. However,

with their published times on a CDC 6600, we felt that some comparison could be made if we ran the same problems on a CDC 6600. Thus, in order to compare our results with the results of [15], we solved the same five assignment problems. These results are contained in Table I. When comparing these results, it is important to note that we are not sure that the code PD-AAL is an all FORTRAN code and it is our understanding that the code is fully optimized to exploit the special hardware features of the CDC 6600. This type of specialization could easily increase the performance of all the other codes by a factor of 2 or 3.

Table 1

TOTAL SOLUTION TIMES ON 200 X 200 ASSIGNMENT PROBLEMS
(IN SECONDS) ON A CDC 6600 WITH A COST RANGE OF 1-100

No. of Arcs	1500	2250	3000	3750	4500	Sum of Times
ARC-II	1.45	1.95	2.47	2.67	3.13	11.67
BENN	17.44	20.31	24.92	27.40	--	--
BSRL	30.39	22.08	20.02	23.11	21.08	116.68
DNET	19.87	26.58	27.98	30.15	31.57	136.15
GM	35.67	28.43	31.39	18.62	23.48	137.59
PD-AAL	1.63	1.14	1.89	1.29	1.80	7.75
PNET-I	2.31	3.71	3.47	3.44	4.79	17.72
SA-AB	.97	1.12	1.48	1.61	1.68	6.86
SHARE	19.93	21.17	25.81	24.95	27.05	118.91
SUPERK	6.44	6.47	7.25	6.95	7.56	34.67
SUPERT-2	1.26	1.57	1.98	2.17	2.53	9.51

A noteworthy feature of the computational results is that SA-AB, PD-AAL, SUPERT-2, and ARC-II are in this order superior to the other codes. Based on the sum of the solution times, SA-AB is roughly fifteen percent faster than its closest competitor (the machine-tuned code) and is roughly fifty percent faster than its next closest competitor. Further, the computational results indicate that the AB algorithm reduces the number of pivots by 25% over the simplex algorithm. Additionally, the results indicate that the AB algorithm not only reduces the number of degenerate pivots but also reduces the number of nondegenerate pivots performed on the denser problems. Moreover, the reduction in the nondegenerate pivots (that is, the number of extreme points visited) versus the simplex algorithm increases as the number of arcs increases.

Further, these results indicate that this first implementation of the AB algorithm is 1 1/2 times as fast as the fastest uncapacitated primal simplex transportation code SUPERT-2. Historically, it has always been possible to improve the solution speed of the first implementation of an algorithm by a factor of 2 or 3. Thus, coupling this with the fact that the start and pivot rules of SA-AB have not been computationally investigated, it appears that the AB algorithm is probably twice as fast as other algorithms for solving assignment problems. This result is extremely important since it completely contradicts the older folklore that primal-dual and out-of-kilter algorithms are the fastest and the more recent folklore that special purpose primal simplex based codes are the fastest.

Looking at the out-of-kilter and dual simplex codes' solution times, it is interesting to note that these solution times are much slower than the

other codes; further, their times are much more dependent on the number of arcs (holding all other parameters of the problem constant) than the other codes. Another important result which can be gleaned from Table I is that the dual simplex method is not competitive with any of the other algorithms.

After comparing all of the codes on the assignment problems, we chose the three fastest available codes (note that the proprietary PD-AAL code was not available) to compare on semi-assignment problems. In contrast to the assignment test problems, the specifications of the semi-assignment test problems vary greatly in both the number of nodes and arcs. As shown in Table II, the eleven test problems vary in size from 50 origins and 500 destinations to 400 origins and 4000 destinations. The number of arcs varies from 2000 arcs to 20,000 arcs. The cost range of the test problems is 1-1000.

The solution times in Table II again indicate that the SA-AB code is substantially superior to the simplex codes. The SA-AB times strictly dominate the times of the other codes. Comparing the sum of the total solution times, the SA-AB code is 2.55 times faster than one of the fastest simplex transportation codes, SUPERT-2.

Another noteworthy feature of the computational results is that the SA-AB code is increasingly superior on the largest test problems. The SA-AB is a full three times faster than the SUPERT-2 code on the last two test problems which each have 400 origins and 4000 destinations.

Table II

TOTAL SOLUTION TIMES ON SEMI-ASSIGNMENT PROBLEMS
(IN SECONDS) ON A CDC 6600 WITH A COST RANGE OF 1-1000.

No. of Nodes m x n	No. of Arcs	ARC-II	SA-AB	SUPERT-2
50 x 500	2,000	2.37	1.14	2.85
50 x 500	5,000	3.53	2.29	3.51
50 x 500	10,000	6.56	4.00	5.53
50 x 1000	4,000	4.27	2.75	6.15
50 x 1000	10,000	9.34	5.64	11.64
50 x 1000	20,000	DNR	8.17	17.02
100 x 1000	4,000	5.59	3.20	6.22
100 x 1000	10,000	10.25	5.62	10.99
100 x 1000	16,000	15.40	8.16	14.27
400 x 4000	10,000	DNR	21.47	61.13
400 x 4000	16,000	DNR	27.81	91.27
Sum of Total Times		-	90.25	230.58

DNR--Did not run as a result of memory limitations.

Based on the results of this testing, the fact that the SA-AB code is the first implementation of the AB algorithm, and the fact that the SA-AB code has not been computationally refined (i.e., alternative start and pivot rules were not examined), the manifest superiority of the AB algorithm provides only a conservative indication of its efficiency for solving both assignment and semi-assignment problems. This result is extremely encouraging since we have recently extended the concepts of this algorithm to arbitrarily capacitated generalized transshipment problems.

6.2 Memory Requirements of the Codes

Table III indicates the number of origin, destination, and arc length arrays required in each of the codes tested for solving assignment and semi-assignment problems except for the PD-AAL code. The storage requirements of this code were not available. It should be noted that memory requirements of all of the codes tested were quite close (within 1500 words) excluding the array requirements. Thus, the important factor in comparing the codes is the number of origin, destination, and arc length arrays.

Looking at Table III and keeping in mind that any meaningful problem has to have more arcs than nodes, it is clear that the AB, primal simplex, and dual simplex codes have a distinct advantage (in terms of memory requirements) over all of the other codes. Further, this advantage greatly increases as the number of arcs increases. For a problem which has ten times as many arcs as nodes, ARC-II, DNET, PNET-I, SUPERT-2, or SA-AB require only about one-half the memory required by the best of the other codes. This enables the AB and simplex based codes to solve much larger problems than other codes. Further, it is important to note that the AB based code, SA-AB, requires the least amount of memory of all of the codes.

Table III

Code Specifications

	Developer	Name	Type	Number of Arrays
1.	Barr	SUPERT-2	Primal Simplex Transportation	5 (M + N) + 2A
2.	Barr, Glover, Klingman	ARC-II	Primal Simplex Network	7 (M + N) + 2A
3.	Barr, Glover, Klingman	SA-AB	AB Algorithm	4M + 2N + 2A
4.	Barr, Glover, Klingman	SUPERK	Out-of-kilter	4 (M + N) + 9A
5.	Bennington	BENN	Non-simplex	6 (M + N) + 11A
6.	Bray and Witzgall	BSRL	Out-of-kilter	6 (M + N) + 8A
7.	Clasen	SHARE	Out-of-kilter	6 (M + N) + 7A
8.	Decision System Associates	PD-AAL	Primal-Dual	Not available
9.	Glover, Karney, Klingman	DNET	Dual Simplex Network	7 (M + N) + 2A
10.	Glover, Karney, Klingman, Stutz	PNET-I	Primal Simplex Network	6 (M + N) + 2A
11.	General Motors	GM	Out-of-kilter	3 (M + N) + 5A

M = Origin Length

N = Destination Length

A = Arc Length

ACKNOWLEDGEMENTS

This research was partly supported by Project NR074-146, ONR Contract N00014-76-C-0383 with Decision Analysis and Research Institute and by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00017-75-C-0569 with the Center for Cybernetic Studies, The University of Texas. The authors wish to acknowledge the cooperation of the staff of The University of Texas Computation Center, The University of Texas Business School Computation Center, and the Southern Methodist University Computation Center.

The authors also wish to acknowledge the indispensible assistance of Dr. John Hultz and Mr. David Karney, systems analysts for Analysis, Research, and Computation, Inc., in the solution testing phase of this study.

REFERENCES

1. Analysis, Research, and Computation, Inc., "Development and Computational Testing on Large Scale Primal Simplex Network Codes," ARC Technical Research Report, P.O. Box 4067, Austin, TX 78765 (1974).

2. R. S. Barr, "Streamlining Primal Simplex Transportation Codes," Research Report to appear, Center for Cybernetic Studies, University of Texas, Austin, Texas.

3. R. S. Barr, F. Glover, and D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," *Mathematical Programming*, 7, 1, 60-87 (1974).

4. R. S. Barr, F. Glover, and D. Klingman, "The Alternating Basis Algorithm for Assignment Problems," Research Report CCS 263, Center for Cybernetic Studies, University of Texas at Austin, Austin, Texas 78712 (1976).

5. R. S. Barr, F. Glover, and D. Klingman, "Enhancements to Spanning Tree Labeling Procedures for Network Optimization," Research Report CCS 262, Center for Cybernetic Studies, University of Texas at Austin, Austin, Texas 78712 (1976).

6. G. E. Bennington, "An Efficient Minimal Cost Flow Algorithm," *Management Science*, 19, 9, 1021-1051 (1973).

7. G. Bradley, G. Brown, G. Graves, "Tailoring Primal Network Codes to Classes of Problems with Common Structure," ORSA/TIMS Conference, Las Vegas (1975).

8. W. H. Cunningham, "A Network Simplex Method," Technical Report No. 207, Department of Mathematical Sciences, Johns Hopkins University (1974).

9. M. Florian and M. Klein, "An Experimental Evaluation of Some Methods of Solving the Assignment Problem," Technical Report No. 41, Operations Research Group, Columbia University, New York (1969).

10. F. Glover and D. Klingman, "Locating Stepping-Stone Paths in Distribution Problems Via the Predecessor Index Method," *Transportation Science*, 4, 220-226 (1970).

11. F. Glover, D. Karney, and D. Klingman, "Implementation and Computational Study on Start Procedures and Basis Change Criteria for a Primal Network Code," *Networks*, 4, 3, 191-212 (1974).

12. F. Glover, D. Karney, and D. Klingman, "Augmented Predecessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices in Distribution Problems," *Transportation Science*, 6, 1, 171-181 (1972).

13. F. Glover, D. Karney, D. Klingman, and A. Napier, "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," *Management Science*, 20, 5, 793-819 (1974).

14. F. Glover, D. Klingman, and J. Stutz, "Augmented Threaded Index Method for Network Optimization," *INFOR*, 12, 3, 293-298 (1974).

15. R. S. Hatch, "Bench Marks Comparing Transportation Codes Based on Primal Simplex and Primal-Dual Algorithms," *Operations Research*, 23, 6, 1167-1171 (1975).

16. B. Harris, "A Code for the Transportation Problem of Linear Programming," *JACM*, 23, 1, 155-157 (1976).

17. D. Klingman, A. Napier, and J. Stutz, "NETGEN-A Program for Generating Large Scale (Un)Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," *Management Science*, 20, 5, 814-822 (1974).

18. J. Mulvey, "Column Weighting Factors and Other Enhancements to the Augmented Threaded Index Method for Network Optimization," Joint ORSA/TIMS Conference, San Juan, Puerto Rico (1974).

19. V. Srinivasan and G. L. Thompson, "Accelerated Algorithms for Labeling and Relabeling of Trees with Application for Distribution Problems," *JACM*, 19, 4, 712-726 (1972).