RAIRO. RECHERCHE OPÉRATIONNELLE

KONSTANTINOS PAPARRIZOS

A non-dual signature method for the assignment problem and a generalization of the dual simplex method for the transportation problem

RAIRO. Recherche opérationnelle, tome 22, n° 3 (1988), p. 269-289

<http://www.numdam.org/item?id=RO_1988__22_3_269_0>

Recherche opérationnelle/Operations Research
(vol. 22, n° 3, 1988, p. 269 à 289)

A NON-DUAL SIGNATURE METHOD FOR THE ASSIGNMENT PROBLEM AND A GENERALIZATION OF THE DUAL SIMPLEX METHOD FOR THE TRANSPORTATION PROBLEM (*)

by Konstantinos PAPARRIZOS (1)

Abstract. — *A combination of a new kind of dual relaxation with the signature idea provides a non-dual signature method for the assignment problem. The dual relaxation is also extended to transportation problems.*

The algorithms, which consist of consequitive stages each one being a set of simplex type pivots, can be considered as a generalization of the dual simplex method. From the computational point of view, the algorithms possess a new monotonic property, namely, during a stage of the algorithms the per unit improvement of the objective function is non-decreasing.

Keywords : Assignment problem, Transportation problem, Dual simplex method, Signatures, Pivoting.

Résumé. — *Une combinaison d'un nouveau type de relaxation duel avec l'idée de la signature fournit une non-duel méthode de signature pour le problème d'assignation. La relaxation duel se prolonge aussi au problème de transportation.*

Les algorithmes qui sont formés par des stages consécutives dont chacun d'eux étant un ensemble simplex de type de pivot, peuvent être considérer comme une généralization de la méthode de simplex duel. Du point de vue computation, les algorithmes possédent une nouvelle propriété monotone, autrement dit, durant un stage des algorithmes l'amélioration par unité de la fonction objective est non décroissante.

Mots clés : Problème d'assignation, problème de transportation, méthode de simplex duel, signatures, pivotation.

1. INTRODUCTION

The assignment problem is perhaps the most studied and well solved problem in mathematical programming and numerous algorithms have been discovered to solve it in polynomial time. It has been generalized to bottleneck, quadratic and algebraic cases and has many applications. Some algorithms, initially constructed for the assignment problem, have been

(*) Received
(1) Democritus University of Thrace, Department of Mathematics, 67100 Xanthi, Greece.

Recherche opérationnelle/Operations Research, 0399-0559/88/03 269 21/$ 4.10

extended to network flow problems and even to general linear programming. A classical example in this case is Kuhn's [15] so called Hungarian method.

Solution procedures for the assignment problem vary from network flows [9, 10, 11], recursive [20], cost parametric [19], relaxation [12, 14], primal dual [6, 15, 17, 21], primal simplex [5, 13, 18] and dual simplex methods [2, 3, 12].

Dual in nature algorithms have (at the present stage of knowledge) better worst case bounds than primal in nature methods. An efficient version of Kuhn's [15] Hungarian method (*see* [16, p. 205]), a hybrid algorithm due to Bertsekas [6], a relaxation method due to Hung and Rom [14] and three recently discovered signature methods (Balinski [2, 3] and Goldfarb [12]) have worst case bound $O(n^3)$. All these algorithms are dual in nature. On the other hand, the primal simplex algorithm due to Roohy-Laleh [18] and the algorithm due to Hung [13], which is based upon Cunningham's [7] strong feasible trees (*see* also Barr *et al.* [5], who call such trees alternating path bases) run in $O(n^5)$ and $O(n^5 \ln \Delta)$ time respectively, where Δ is the difference between the initial and the final values of objective. An exception is an efficient version of Balinski's and Gomory's [4] primal algorithm due to Cunningham and Marsh [8], which runs in $O(n^3)$ time.

The signature idea has been introduced by Balinski [1]. The same author used this idea later to construct two particularly simple and efficient algorithms (Balinski [2, 3]). Both algorithms are dual simplex methods and run in $O(n^3)$ time. Balinski [3] reports that preliminary computational studies are very encouraging for his algorithms. In addition, the method in [3] includes a theory of strong dual feasible trees with remarkable properties. Goldfarb [12] used the signature idea inductively and developed a relaxation $O(n^3)$ algorithm, which produces an optimal solution to the $k \times k$ problem from an optimal solution to the $(k-1) \times (k-1)$ problem.

A signature of a basic tree (solution) to the assignment problem is the vector of its column or row degrees. Signature methods work as follows. They start with a dual feasible tree. Then, choosing properly the edge to be deleted, construct a sequence of dual feasible trees and stop when a tree with a desired signature is found. The desired signature is of type $(2, 2, \ldots, 2, 1)$, and quarantees primal feasibility of the tree (Balinski [2, 3]). Since dual feasibility is kept throughout the computation, the final tree is also dual feasible and therefore it is an optimal solution to the assignment problem. It is obvious that dual feasibility is not required throughout the computation. What is really needed, is dual feasibility of the first tree having the desired signature.

Motivated by this observation we investigated whether dual feasibility can be relaxed. Our effort resulted in a new dual relaxation procedure, which can be seen as a generalization of the dual simplex method for network flow problems. The algorithms constructed in this way, consist of consecutive stages. Each one of these stages is a set of pivots that resemble to the pivots of the dual simplex method. The algorithms start with a dual feasible tree and assure that the first tree in every stage is also dual feasible. A stage is initialized by defining properly a set of edges, which are candidates to be deleted in subsequent iterations. We call this set "candidate set" and the edges in it "candidate edges". The iterations are similar, in a sense, to the iterations of the dual simplex method, particularly in choosing the non-tree edge to enter the basis. When the entering edge is adjoined to the current tree, it determines a unique candidate edge, which is deleted to construct the adjacent tree. In that sense our algorithms look like a primal simplex method; first a non-tree edge is adjoined and then an in-tree edge is deleted. The stage is completed when all candidate edges are deleted. Intermediate trees in a stage are not dual feasible, except possibly in the case of dual degeneracy. When the outcome of the algorithms is a primal feasible tree, then, this tree is an optimal solution. This is because the primal feasible tree comes out after the completion of a stage. This is always the case for full dense problems. Otherwise, the algorithms stop when no edge eligible to enter the basis exists, indicating that the problem has no optimal solution. This case may happen when sparse problems are considered. When all stages consist of a single iteration the algorithms coincide with the dual simplex method. An interesting new monotonic property possessed by the algorithms is that the per unit improvement of the objective function during the iterations of a stage, is nondecreasing.

In paragraph 3 we combine the dual relaxation procedure with the signature idea. In particular, the set of candidate edges consists of edges that can be deleted by Balinski's [3] competitive dual simplex algorithm. For this reason the reader is advised to be aware of the results in [3]. In paragraph 4 we show that the $n \times n$ assignment problem is solved in at most $(n-1)(n-2)/2$ iterations and in at most $O(n^4)$ time. The generalization to transportation problems is presented in paragraph 5. This is accomplished by introducing two types of stages. Section 6 describes how sparsity is handled. The algorithm for transportation problems is easily extended to minimum cost transhipment problems. However, we have not been able to use the dual relaxation procedure in phase one. That is the reason why the algorithm for transhipment problems is briefly described in the last section, where some properties

and ideas for future research are also presented. The next section includes preliminary results.

The main purpose of this paper is to present a new approach to attack network flow problems. We believe that this approach deserves further attention for three reasons.

(1) Since it is in the earliest stage, future research can be initiated. The algorithms are very flexible in choosing the candidate set and the edge to be deleted. From our perspective, the three most interesting open research problems are: (i) Can the procedure be generalized to upper bounding network flow problems? (ii) Can the procedure be incorporated into a phase one? (iii) Is there any variant consisting of a single stage? (2) The approach seems promising from the computational point of view. The new monotonic property, which accompanies the procedure (at least) as long as it is applied to network flow problems, seems that it may reduce the average number of iterations. In addition, advanced techniques similar to those used in [2, 3, 12] that transform signature methods into $O(n^3)$ algorithms, can be used to reduce the number of comparisons per iteration. Note that the main work of algorithms for network flow problems is done on comparisons. Moreover, since the largest the candidate set the faster the algorithms, effectiveness is expected in large scale problems. (3) The algorithms construct paths to the optimal solution passing outside the feasible region. In our opinion this fact is very important from the theoretical point of view (and probably from the computational). It is well known that there are always shortcuts to the optimal solution passing through the infeasible region. We consider the results presented here as the first step towards constructing such algorithms.

2. PRELIMINARIES

Consider the full dense assignment problem

$$(\mathrm{AP}):\ \min \Sigma_i \Sigma_j c_{ij} x_{ij},\ \text{s. t. } \Sigma_j x_{ij} = 1, \quad \Sigma_i x_{ij} = 1, \quad x_{ij} \geqq 0, \quad i \in I,\ j \in J,$$

where $I = \{1, 2, \ldots, n\}$ is the set of row nodes (in the figures they are drown as circles), $J = \{1, 2, \ldots, n\}$ is the set of column nodes (in the figures they are drawn as squares) and c_{ij} are real numbers.

A basic solution to (AP) can be equivalently represented by a spanning tree in the bipartite graph $G = (I, J, E)$, where I is the set of row nodes, J the set of column nodes and $E = \{(i, j): i \in I, j \in J\}$ is the edge set. A spanning tree of G contains exactly $2n - 1$ edges. Sometimes we will call spanning trees

simply trees or bases. To every spanning tree, T, there are associated the primal variables x_{ij}, $i \in I$, $j \in J$, the dual row variables $u=(u_1, u_2, \ldots, u_n)$, the dual column variables $v=(v_1, v_2, \ldots, v_n)$ and the reduced costs, w_{ij}, $i \in I$, $j \in J$ denoted by $x_{ij}(T)$, $u_i(T)$, $v_j(T)$ and $w_{ij}(T)$ respectively.

Consider first the primal variables. If an edge $(i, j) \in T$, it is basic, otherwise nonbasic. Similarly a primal variable x_{ij} is basic (nonbasic) if (i, j) is basic (nonbasic). The values $x_{ij}(T)$ are determined as follows. Set first $x_{ij}(T)=0$ for every nonbasic variable. Then, compute the values of basic variables so that equations of (AP) are satisfied. It is well known that the values $x_{ij}(T)$ are integers and easily computed recursively. The values of dual variables are again easily determined by solving recursively the system of equations

$$u_i(T)+v_j(T)=c_{ij}, \qquad (i, j) \in T.$$

It is easily verified that $u_i(T)$ and $v_j(T)$ are integer valued whenever all costs c_{ij} are integers. With the dual variables at hand the reduced costs are directly computed via the relations

$$w_{ij}(T)=c_{ij}-u_i(T)-v_j(T), \qquad i \in I, \quad j \in J$$

and they are integer valued whenever the costs are integers.

A spanning tree, T, is said to be "primal feasible" if $x_{ij}(T) \geqq 0$ for $i \in I$, $j \in J$ and "dual feasible" if $w_{ij}(T) \geqq 0$ for $i \in I$, $j \in J$. Primal simplex algorithms decrease (not always strictly) the objective function moving from one primal feasible tree to an adjacent one. Dual simplex algorithms increase the objective function moving between adjacent dual feasible bases. Primal and dual simplex algorithms differ among each other in the way the adjacent trees are constructed. A primal simplex algorithm adjoins first an edge $(g, h) \notin T$ to the current primal feasible tree and then deletes an edge $(k, l) \in T \cup (g, h)$ chosen so that the adjacent tree is also primal feasible. On the contrary, dual methods delete first an edge $(k, l) \in T$ and then adjoin an edge (g, h) chosen so that $\{T \sim (k, l)\} \cup (g, h)$ is a dual feasible tree.

A pivot operation on the current tree, T, consists in constructing the adjacent tree, T', and in computing the values $x_{ij}(T')$, $u_i(T')$, $v_j(T')$ and $w_{ij}(T')$. These values are easier computed from the corresponding values associated with T. The procedure is the same for primal and dual algorithms.

Consider first the primal variables. $T \cup (g, h)$ contains a unique cycle, $\Phi(g, h)$, that necessarily includes edge $(g, h) \notin T$. Since graph G is bipartite, $\Phi(g, h)$ contains an even number of edges. Therefore, we can label adjacent edges in $\Phi(g, h)$ so that one is "minus" the other "plus". The deleted

edge $(k, l) \neq (g, h)$ belongs in $\Phi(g, h)$, and it is labeled minus. Clearly, $T' = \{T \cup (g, h)\} \sim (k, l)$ is a spanning tree. The values of the primal variables associated with the new tree T' are given by

$$\begin{aligned} x_{ij}(T') &= x_{ij}(T) - \varepsilon, \quad \text{if} \quad (i, j) \in \Phi(g, h) \quad \text{is minus,} \\ &= x_{ij}(T) + \varepsilon, \quad \text{if} \quad (i, j) \in \Phi(g, h) \quad \text{is plus,} \\ &= x_{ij}(T), \quad \text{otherwise,} \end{aligned}$$

where $\varepsilon = x_{kl}(T)$. Figure 1 illustrates these operations.

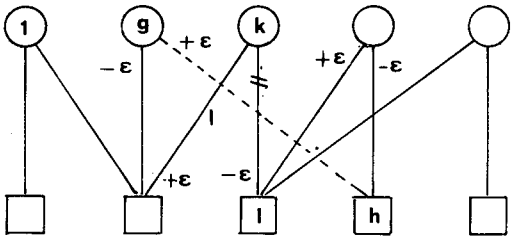

Figure 1. — A spanning tree T; plus-minus labeling of edges in $\Phi(g, h)$ and values of new primal variables $x_{ij}(T')$.

Consider now the remaining variables. $T \sim (k, l)$ consists of two subtrees. The one, T^r, contains the row mode k, the other, T^c, contains column mode l (see *Fig.* 2). The entering edge is chosen so that $g \in T^c$ and $h \in T^r$. Setting $\delta = w_{gh}(T)$ the values of new dual variables are given by the relations

$$\left.\begin{aligned} u_i(T') &= u_i(T), \quad i \in T^r; \qquad v_j(T') = v_j(T), \quad j \in T^r \\ u_i(T') &= u_i(T) - \delta, \quad i \in T^c; \qquad v_j(T') = v_j(T) + \delta, \quad j \in T^c. \end{aligned}\right\} \tag{1}$$

These operations are illustrated in Figure 2. Using relations (1) we can easily verify that

$$\left.\begin{aligned} w_{ij}(T') &= w_{ij}(T) - \delta, \qquad i \in T^c, \quad j \in T^r, \\ &= w_{ij}(T) + \delta, \qquad i \in T^r, \quad j \in T^c, \\ &= w_{ij}(T), \quad \text{otherwise.} \end{aligned}\right\} \tag{2}$$

A basic solution to (AP), which is primal and dual feasible is an optimal solution (by the duality theorem of linear programming). The per unit increase or decrease of the objective function is $w_{gh}(T)$. Also, it is easy to see that the value of the objective function changes at each iteration by the amount $x_{gh}(T')\, w_{gh}(T)$. In dual simplex algorithms it is always $w_{gh}(T) \geqq 0$. Consequently, an increase of the objective function occurs if the adjoined edge

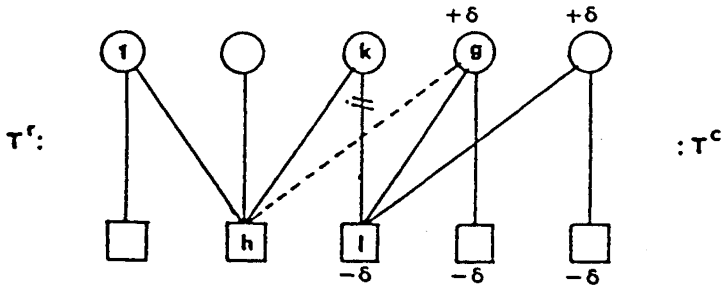

Figure 2. — A spanning tree T and subtrees T^r, T^c; values of new dual variables; (g, h) adjoined edge, (k, l) deleted edge.

(g, h) is chosen so that $x_{gh}(T') \geqq 0$. Since for the deleted edge (k, l), which is first determined and has label minus, it is $x_{kl}(T) = \varepsilon < 0$, (g, h) must be chosen so that it has label minus. This fact is accomplished by choosing (g, h) between the edges (i, j) such that $i \in T^c$, $j \in T^r$. Now, using relations (2) it is easily verified that dual feasibility of T' is quaranteed by choosing (g, h) via the relation

$$w_{gh}(T) = \min \{w_{ij}(T) : i \in T^c, j \in T^r\}. \tag{3}$$

To summarize, a dual simplex algorithm deletes first an edge (k, l), $x_{kl}(T) < 0$ and then adjoins an edge determined by (3). The procedure is repeated and stops when a primal feasible tree is found.

Reasoning similarly we can see that a primal simplex method adjoins an edge (g, h), $w_{gh}(T) < 0$, to the current primal feasible tree T and then deletes another edge $(k, l) \in \Phi(g, h)$, $x_{kl}(T) \geqq 0$, chosen so that (k, l) has label minus and (g, h) has label plus.

DEFINITION 1: We say that a pivot operation on T is dual if the deleted and adjoined edges are both minus.

3. THE NON-DUAL SIGNATURE METHOD

Our signature method is a variant of Balinski's [3] competitive (dual) simplex algorithm. It constructs rooted trees that have precisely the same primal structure as Balinski's strong dual feasible bases. The root of the trees is always row node one and it is kept fixed throughout the computation. The root directs the edges away from it. If an edge $(i, j) \in T$ is directed from row node i to column node j it is called "odd", otherwise "even". An edge or

node of T is "higher" than a second edge or node (and the second is then "lower") if the first is in the path joining the second to the root.

Given a tree, T, the column signature is the vector $a=(a_1, a_2, \ldots, a_n)$ of its column node degrees (Balinski [2, 3]). The row signature $b=(b_1, b_2, \ldots, b_n)$ of T is similarly defined.

DEFINITION 2: If T is a rooted tree satisfying, $(i, j)\in T$ and even, implies $x_{ij}(T) \geqq 1$, and $(i, j)\in T$ and odd, and not an edge $(1, j)$ with j of degree 1, implies $x_{ij} \leqq 0$, then T is said to be a "strong basis".

Recall now that Balinski's [3] strong dual feasible trees are our strong trees (as described here) with the additional condition that they are dual feasible. Consequently the following Lemma 1 holds for strong bases (for the proof *see* [3]).

LEMMA 1: *If T is a strong basis that contains a column node j of degree at least* 3, *then* $x_{ij}(T) \leqq -1$ *for the unique odd edge* $(i, j)\in T$.

The initial tree, T, has row signature $b=(n, 1, 1, \ldots, 1)$ and it is constructed as in other signature methods. It consists of n edges $(1, j)\in T$, $w_{1j}(T)=0$ for $j\in J$, found by setting $u_1(T)=0$, $v_j(T)=c_{1j}$; one edge $(i, r)\in T$ for each $i\neq 1$, $w_{ir}(T)=0$, found by setting

$$u_i(T)=c_{ir}-v_r(T)=\min_j(c_{ij}-v_j(T)).$$

Each stage of the algorithm starts out with a dual feasible strong tree T. Then, some edges of T are chosen in the following manner. Start tracing down the paths of T starting from the root. The first time a column node, j, of degree at least three is met, delete the unique edge in the path incident to column node j. Denote this edge by (i, j) and call it a "candidate edge". The deletion of the candidate edge cuts off from the root a subtree T_j which is called "candidate subtree". The candidate subtree T_j is removed from T and another path in the subtree containing the root is traced down determining a new candidate edge and a new candidate subtree. The procedure is repeated and stops when the subtree including the root contains no odd edge of degree at least three. Figure 3 illustrates the candidate edges and subtrees of a strong basis. Notice that a candidate edge (i, j) is an odd edge in T and that the path joining column node j to the root contains no other column node of degree at least 3.

The procedure described above is only used to determine the candidate edges and subtrees. No candidate edge is immediately deleted. The choice of the edge to be deleted is made after the entering edge has been determined. Denote by (i_1, j_1), $(i_2, j_2)\ldots(i_k, j_k)$ and $T_{j_1}, T_{j_2}, \ldots, T_{j_k}$ the candidate edges

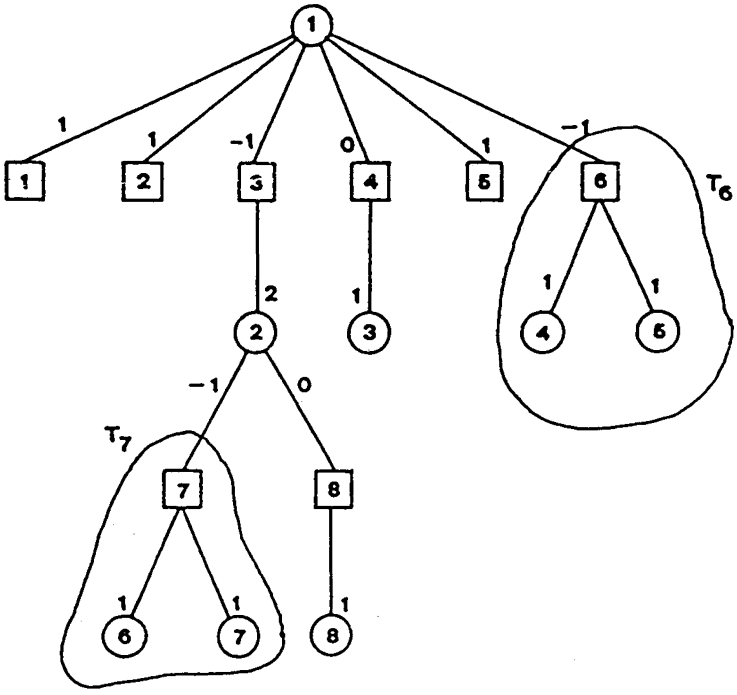

Figure 3. — A strong basis; (2,7), (1,6) candidate edges; T_6, T_7 candidate subtrees.

and subtrees respectively and set $T_- = \bigcup_{r=1}^{k} T_{j_r}$ and $T_+ = T \sim T_-$. At the first iteration of a stage, the adjoined edge (g, h) is determined via the relation

$$w_{gh}(T) = \min \{w_{ij}(T) : i \in T_-, j \in T_+\}. \tag{4}$$

Edge (g, h) is then adjoined to T and the unique cycle $\Phi(g, h)$ is determined. $\Phi(g, h)$ contains necessarily a unique candidate edge, say (i_σ, j_σ). Candidate edge (i_σ, j_σ) is deleted in order to produce the adjacent tree T', which may not be dual feasible. At the next iteration, the new subtrees $T'_- = T_- \sim T_{j_\sigma}$ and $T'_+ = T_+ \cup T_{j_\sigma}$ are computed. T'_- and T'_+ are used in (4) to determine a new entering edge (g', h'). $\Phi(g', h')$ contains a new unique candidate edge, which is deleted. The procedure is repeated until all candidate edges are deleted. Then, the next stage of the algorithm starts. The algorithm stops when no candidate edge can be found. Observe that Lemma 1 quarantees

that candidate edges exist as long as the current tree contains column nodes of degree at least three.

Notice that for full dense assignment problems there are always edges eligible for entering the current basis. This is not true for sparse problems.

4. JUSTIFICATION OF THE ALGORITHM

In this section we show that the algorthm finds the optimal solution in at most $(n-1)(n-2)/2$ iterations. In addition to Lemma 1 we will use the following two lemmas from Balinski [3].

LEMMA 2: *If a spanning tree contains a unique column node of degree one, then it is primal feasible.*

LEMMA 3: *If T is a strong basis and a dual pivot is made by deleting an edge (i, j) with $x_{ij}(T) \leqq -1$ and j a column of degree at least 3, then the adjacent tree T' is also a strong basis.*

Our algorithm performs always dual pivot operations (in the sense of Definition 1). Since the first tree is a strong basis, by Lemma 3 all trees constructed by the algorithm are also strong bases. Suppose now that the algorithm stops. Then, there is no candidate edge, meaning that there is no odd edge (i, j) incident to a column node of degree at least 3. Consequently, all column nodes are of degree 2 or 1. Since there are $2n-1$ edges the algorithm stops with a strong basis, T, containing a unique column node of degree 1 and by Lemma 2, T is primal feasible. Therefore we have shown the following.

THEOREM 4: *The algorithm stops with a primal feasible strong basis.*

To prove that the algorithm terminates with an optimal solution to (AP) it remains to be shown that the last strong basis is dual feasible. To this end it suffices to show that the first tree in every stage is dual feasible. Recall that the algorithm stops always at the beginning of a stage.

Let T^k, T^{k+1} be two consecutive strong bases in a stage of the algorithm. Denote by T^k_j the candidate subtrees associated with the strong tree T^k and set

$$T^k_- = U_j T^k_j, \qquad T^k_+ = T^k \sim T^k_-.$$

Also, set

$$\delta_k = \min \{w_{ij}(T^k) : i \in T^k_-, j \in T^k_+\},$$
$$\delta'_k = \min \{w_{ij}(T^k) : i \in T^k_+, j \in T^k_-\}.$$

DEFINITION 3: Given a subtree, T', of a spanning tree, T, we say that T' is dual feasible if $w_{ij}(T) \geqq 0$ for every edge (i, j) such that $i \in T'$, $j \in T'$.

LEMMA 5: *Let T^k, T^{k+1} be two consecutive strong bases in a stage of the algorithm. If T^k_- is dual feasible and $\delta_k \geqq 0$, then $\delta_{k+1} \geqq \delta_k$.*

Proof: Let T^k_1, be the candidate subtree cut off from T^k. Since $T^{k+1}_+ = T^k_+ \cup T^k_1$ we can set

$$\delta_{k+1} = \min \{w_{ij}(T^{K+1}) : i \in T^{k+1}_-, j \in T^{k+1}_+\} \tag{5}$$
$$= \min \{a_1, a_2\},$$

where

$$a_1 = \min \{w_{ij}(T^{k+1}) : i \in T^{k+1}_-, j \in T^k_+\},$$
$$a_2 = \min \{w_{ij}(T^{k+1}) : i \in T^{k+1}_-, j \in T^k_1\}.$$

For $i \in T^{k+1}_- = T^k_- \sim T^k_1$ and $j \in T^k_+$ it is $w_{ij}(T^{k+1}) = w_{ij}(T^k)$. Since $T^{k+1}_- \subset T^k_-$ we conclude that $a_1 \geqq \delta_k$. Candidate tree T^k_1 contains the column node to which the deleted edge is incident. Using relations (2) we get $w_{ij}(T^{k+1}) = w_{ij}(T^k) + \delta_k$ for $i \in T^{k+1}_-$, $j \in T^k_1$. Since T^k_- is dual feasible, $w_{ij}(T^k) \geqq 0$, $i \in T^{k+1}_- \subset T^k_-$, $j \in T^k_1 \subset T^k_-$. Therefore, it is $a_2 \geqq \delta_k$ and by (5) it is $\delta_{k+1} \geqq \delta_k$ completing the proof. $\triangle$

LEMMA 6: *Let T^k, $k = 1, 2, \ldots$ be consecutive strong trees in a stage of the algorithm. If T^1 is dual feasible, then $\delta'_{k+1} \geqq -\delta_k$, $k = 1, 2, \ldots$*

Proof: Let T^1_1 be the candidate subtree cut off from T^1. Then

$$\delta'_2 = \min \{w_{ij}(T^2) : i \in T^2_+, j \in T^2_-\}$$
$$= \min \{b_1, b_2\},$$

where

$$b_1 = \min \{w_{ij}(T^2) : i \in T^1_+, j \in T^2_-\},$$
$$b_2 = \min \{w_{ij}(T^2) : i \in T^1_1, j \in T^2_-\}.$$

By relations (2), $w_{ij}(T^2) = w_{ij}(T^1)$ for $i \in T^1_+$, $j \in T^2_-$. Dual feasibility of T^1 implies that $b_1 \geqq 0$. For $i \in T^1_1$, $j \in T^2_-$ we get (again) from relations (2) that

$w_{ij}(T^2)=w_{ij}(T^1)-\delta_1$. Since $\delta_1 \geqq 0$, $w_{ij}(T^2) \geqq -\delta_1$, $i \in T_1^1$, $j \in T_-^2$. Therefore

$$\delta_2' = \min\{b_1, b_2\} \geqq \{0, -\delta_1\} \geqq -\delta_1.$$

Assume that the lemma is true for k, $k \geqq 2$. By Lemma 5 and dual feasibility of T^1 we get inductively

$$0 \leqq \delta_1 \leqq \delta_2 \leqq \ldots \leqq \delta_k \leqq \delta_{k+1}.$$

Let T_1^{k+1} be the candidate subtree cut off from T^{k+1}.

Then

$$\begin{aligned}\delta_{k+2}' &= \min\{w_{ij}(T^{k+1}) : i \in T_+^{k+2}, j \in T_-^{k+2}\} \\ &= \min\{b_1', b_2'\}\end{aligned}$$

where

$$b_1' = \min\{w_{ij}(T^{k+2}) : i \in T_+^{k+1}, j \in T_-^{k+2}\},$$
$$b_2' = \min\{w_{ij}(T^{k+2}) : i \in T_1^{k+1}, j \in T_-^{k+2}\}.$$

Reasoning as in the case $k=1$, we get $b_2' \geqq -\delta_{k+1}$. Since $T_-^{k+2} \subset T_-^{k+1}$ and $w_{ij}(T^{k+2})=w_{ij}(T^{k+1})$ for $i \in T_+^{k+1}$, $j \in T_-^{k+2}$, $b_1' \geqq \delta_{k+1}'$. By the induction hypothesis, $b_1' \geqq \delta_{k+1}' \geqq -\delta_k$. Since

$$0 \leqq \delta_k \leqq \delta_{k+1}, \qquad \delta_{k+2}' = \min\{b_1', b_2'\} \geqq -\delta_{k+1}. \quad \triangle$$

THEOREM 7: *The first tree in every stage of the algorithm is dual feasible.*

Proof: It suffices to show the statement: if the initial tree in a stage is dual feasible, the initial tree in the next stage is also dual feasible. The theorem is then true because the initial tree in the first stage is dual feasible.

Let T^1, T^2, ... be the consequitive trees in a stage. We will show inductively that T_+^k are dual feasible subtrees. Clearly, T_+^1 is dual feasible (because T^1 is dual feasible). Assume T_+^k is dual feasible. Then $T_+^{k+1} = T_+^k \cup T_1^k$, where T_1^k is the candidate subtree cut off from T^k. By the induction hypothesis $w_{ij}(T^{k+1}) = w_{ij}(T^k) \geqq 0$ for $i \in T_+^k$, $j \in T_+^k$. For the edges (i, j), $i \in T_1^k$, $j \in T_1^k$, it is $w_{ij}(T^{k+1}) = w_{ij}(T^k) = w_{ij}(T^1) \geqq 0$. For edges (i, j), $i \in T_1^k$, $j \in T_+^k$, it is $w_{ij}(T^{k+1}) = w_{ij}(T^k) - \delta_k \geqq 0$. For edges (i, j), $i \in T_+^k$, $j \in T_1^k$ using Lemma 6 we get

$$w_{ij}(T^{k+1}) = w_{ij}(T^k) + \delta_k \geqq \delta_k' + \delta_k \geqq -\delta_{k-1} + \delta_k \geqq 0$$

because $w_{ij}(T^k) \geqq \delta_k'$ for $i \in T_+^k$, $j \in T_1^k \subset T_-^k$ and $\delta_k \geqq \delta_{k-1}$ (by Lemma 5). Therefore, for every edge (i, j), $i \in T_+^{k+1}$, $j \in T_+^{k+1}$ it is $w_{ij}(T^{k+1}) \geqq 0$.

Consider now the first tree at the next stage. Since all candidate subtrees of the previous stage have been cut off, it is dual feasible. △

THEOREM 8: *The algorithm stops with an optimal solution to the (AP).*

Proof: It comes easily from Theorems 4 and 7. △

The counting process for determining the maximum number of iterations uses the notion of the "level" of a tree. The level of a tree is the number of 1's in its column (or row) signature, Balinski [2, 3]. Using levels Lemma 2 can be stated: if a tree is in level 1, it is primal feasible. We will also use the following notation. T^1 is the first tree in level σ, $1 \leqq \sigma \leqq n-1$, and $T^2, T^3, \ldots, T^q$ are the first trees in the next, second next and so on stages respectively. Every tree, T^i, $i=1, 2, \ldots, q$ is in level σ and T^{q+1} in level $\sigma-1$. Notice that if a column node has degree at least two it will never become of degree one. Therefore the level is non-increasing. Let now $n(T)$ denote the number of column nodes of tree (or subtree) T. Finally, let t_1 be the number of the remaining iterations at the first stage and t_i, $i=2, 3, \ldots, q$ the number of iterations at stage i.

THEOREM 9: *The algorithm solves the $n \times n$ assignment problem in at most $(n-1)(n-2)/2$ iterations.*

Proof: If a candidate subtree is cut off and the level is not reduced, the cut off subtree will be part of a candidate subtree at the next stage. This is so because the candidate edges are incident to column nodes of degree at least three and the degree of column node h is at least two. Therefore,

$$n(T^1_-) < n(T^2_-) < \ldots < n(T^q_-).$$

Clearly, $t_1 \leqq n(T^1_-)$. For $i=2, 3, \ldots, q$ it is

$$t_i \leqq n(T^i_-) - n(T^{i-1}_-),$$

because the candidate edges at T^i are incident to column nodes not in T^{i-1}_-.

Therefore, the number of iterations, t, at level σ is

$$t = \sum_{i=1}^{q} t_i \leqq n(T^q_-) \leqq n - \sigma.$$

(T^q_- contains column nodes of degree at least 2). Since $1 \leqq \sigma \leqq n-1$ and the algorithm stops when the first tree in level 1 is reached, the maximum number

of iterations is

$$1+2+\ldots+n-2=(n-1)(n-2)/2. \quad \triangle$$

THEOREM 10: *$(n-1)(n-2)/2$ is the best bound.*

Proof: If at each stage there is a unique candidate edge, the algorithm coincides with Balinski's [3] competitive (dual) simplex algorithm. Therefore, the algorithm applied to Balinski's [2] assignment problem, defined by $c_{ij}=(m-i)(n-j)$, takes precisely $(n-1)(n-2)/2$ iterations. $\triangle$

THEOREM 11: *The algorithm solves the $n\times n$ assignment problem in at most $O(n^4)$ time.*

Proof: Clearly, each iteration of the algorithm requires at most $O(n^2)$ time. The work required in determining the candidate set is done in no more than $2n-1$ comparisons. Therefore, the $n\times n$ assignment problem is solved in at most $O(n^4)$ time. $\triangle$

Note 1: The counting tricks used in [2, 3, 12] that transform signature methods into $O(n^3)$ algorithms don't seem to be combinable with our algorithm. The reason is that several stages may be required to reduce the level by one.

5. AN ALGORITHM FOR FULL DENSE TRANSPORTATION PROBLEMS

Everything said in section 2 can be easily extended to transportation problems

$$\left.\begin{array}{l} \min \Sigma_i \Sigma_j c_{ij} x_{ij} \\ \text{s. t.} \quad \Sigma_j x_{ij}=a_i, \qquad \Sigma_i x_{ij}=b_j, \qquad x_{ij}\geqq 0, \end{array}\right\} \qquad \text{(TP)}$$

where $i\in I=\{1, 2, \ldots, m\}$, $j\in J=\{1, 2, \ldots, n\}$, $\Sigma_i a_i=\Sigma_j b_j$, $a_i\geqq 1$, $i\in I$, and $b_j\geqq 1$, $j\in J$.

A basic solution to (TP) is a tree containing $m+n-1$ edges. The values $x_{ij}(T)$, $u_i(T)$, $v_j(T)$ and $w_{ij}(T)$, associated with the tree T, are determined in a similar manner, i. e., the values $x_{ij}(T)$ are determined via the relations $x_{ij}(T)=0$ for $(i, j)\notin T$, the values $u_i(T)$, $v_j(T)$ via the relations $u_i(T)+v_j(T)=c_{ij}$, $(i, j)\in T$ and $w_{ij}(T)=c_{ij}-u_i(T)-v_j(T)$.

The initial rooted tree (the root is always row node 1) is constructed as in the algorithm for the (AP). Notice that at the initial tree only odd edges can be primal infeasible. All primal infeasible edges $(1, j)$ are the candidate edges

to start the first stage. The adjoined and deleted edges are chosen as in a stage of the algorithm for the (AP). When the first stage is completed either a primal feasible tree has been constructed and the algorithm stops with an optimal solution (because the current tree is also dual feasible) or not. If not, a new stage starts.

The algorithm for (TP), unlike the algorithm for (AP), consist of two types of stages. We call them "odd" or "even". A stage is said to be "odd" (even) if all candidate edges are "odd" (even). The first stage of the algorithm is always odd. Similarly, a candidate subtree is "odd" (even) if the candidate edge associated with it, is odd (even). The candidate edges are determined in a manner similar to that in the algorithm for the (AP). For instance, the candidate edges of an odd (even) stage are determined as follows. Start tracing down a path starting from the root; the highest odd (even) edge in the path, which is primal infeasible, is a candidate edge. Then, choose another path and stop when all paths have been examined.

The pivot operation of an even stage is slightly different than that of an odd stage. Let T be the current tree in an even stage and T_1 a candidate subtree associated with the candidate edge (k, l), $k \in I$, $l \in J$. Since now $k \in T_1$ the entering edge (g, h) must be chosen so that $h \in T_1$. Suppose (k, l) is deleted and (g, h) is adjoined to produce the next tree T'. Setting $w_{gh}(T) = \delta$ and applying relations (1) it is easily varified that

$$\begin{aligned} w_{ij}(T') &= w_{ij}(T) + \delta, \qquad i \in T_1, \quad j \in T \sim T_1, \\ &= w_{ij}(T) - \delta, \qquad i \in T \sim T_1, \quad j \in T_1, \\ &= w_{ij}(T), \quad \text{otherwise.} \end{aligned} \tag{6}$$

The edge to be a adjoined is chosen by the relation

$$w_{gh}(T) = \min \{w_{ij}(T) : i \in T_+, j \in T_-\}. \tag{7}$$

Using relations (6) and (7) and slightly modifying the arguments of section 4 (interchange the roles of row and column nodes), it can be easily verified that an even stage is completed with a dual feasible tree.

Now, it is clear how the algorithm for the (TP) works. When a stage is completed, examine whether the current tree is primal feasible. If it is, stop. Otherwise, either an odd or an even stage can be applied to continue.

6. HANDLING SPARSITY

The two algorithms described earlier work provided they are applied to full dense problems. When dealing with sparsity there are two difficulties that we must overcome. First, an initial dual feasible tree may not be constructed as in the full dense case and second, there might be the case where no edge eligible to enter the basis is available. We describe how the algorithm for (TP) is modified to handle sparse transportation problems.

Suppose some of the costs c_{ij} are undefined because the associated edges (i, j) do not exist. Then, approach the problem as follows. Assign very high costs to not existing edges (i, j) and make them temporarily admissible. Such an edge is called "artificial". Then construct the initial dual feasible tree by applying the algorithm to this transformed as before but consider in computation only existing edges. The initial dual feasible tree can be constructed provided that each row of the cost matrix has at least one (non artificial) edge. Otherwise, there is no primal feasible solution to (TP). These conditions are easily checked.

The algorithm terminates either when a primal feasible solution is found or when no edge eligible for entering the basis exists. Denote by T^* the last tree.

LEMMA 12: *If T^* is a primal feasible tree of the transformed problem containing at least one strictly positive artificial edge, then (TP) is (primal) infeasible. Otherwise, T^* provides an optimal solution to (TP).*

Proof: Clearly, if T^* contains positive artificial edges, (TP) is primal infeasible.

Suppose T^* does not contain any positive artificial edge. Then, either T^* does not contain any artificial edge or all artificial edges of T^* are zero valued. In the former case T^* is a primal feasible solution to (TP). Since primal feasibility is achieved at the beginning of a stage, T^* is also dual feasible. Therefore, T^* is an optimal solution to (TP). In the latter case, delete all zero valued artificial edges from T^*. Then T^* decomposes into subtrees. It is easy to see that every subtree is an optimal solution of a

subproblem of (TP) having as constraints all row and column equalities for which the associated nodes are in the subtree under consideration. $\triangle$

LEMMA 13: *If T^* is not primal feasible and no edge eligible for entering the basis exists, then (TP) has no optimal solution.*

Proof: Case (1). The stage is odd. Let u_i, v_j, w_{ij} be the values of dual variables and reduced costs associated with T^*. We will show that the dual problem of the transformed problem is an unbounded linear program. Notice that T^* may not be dual feasible. For $i \in I$ and $j \in J$ we set

$$\begin{aligned} \bar{u}_i &= u_i + \delta, \qquad i \in T^*_-, \\ &= u_i, \qquad i \in T^*_+, \end{aligned}$$

and

$$\begin{aligned} \bar{v}_j &= v_j - \delta, \qquad j \in T^*_-, \\ &= v_j, \qquad j \in T^*_+. \end{aligned}$$

Since there are no edges (i, j), $i \in T^*$, $j \in T^*_+$ the variables $\bar{u}_i$, $\bar{v}_j$ constitute a dual feasible solution of the transformed problem for sufficiently large δ. To see this observe that

$$\begin{aligned} \bar{w}_{ij} = c_{ij} - \bar{u}_i - \bar{v}_j &= w_{ij} + \delta, \qquad \text{for} \quad i \in T^*_+, j \in TT^*_- \\ &= w_{ij}, \quad \text{otherwise.} \end{aligned}$$

It suffices to take $\delta \geqq -\min\{w_{ij}: i \in T^*_+, j \in T^*_-\}$. Then $w_{ij} + \delta \geqq 0$ for $i \in T^*_+$, $j \in T^*_-$ and because of dual feasibility of T^*_-, T^*_+, it is $\bar{w}_{ij} \geqq 0$ whenever i, $j \in T^*_-$ or $i, j \in T^*_+$.

The value of the objective function, $f(u, v)$, associated with the dual variables u, v is given by

$$f(u, v) = \Sigma_i u_i a_i + \Sigma_j v_j b_j.$$

Let Δ be the sum of the candidate variables. Then $\Delta < 0$. Subtracting all column equations associated with column nodes in T^*_- from row equations associated with row nodes in T^*_- results in

$$\sum_{i \in T^*_-} a_i - \sum_{j \in T^*_-} b_j = -\Delta > 0.$$

Now, it is easily verified that

$$f(\bar{u}, \bar{v}) = f(u, v) - \delta\Delta.$$

So, the maximum of the dual objective is unbounded. The dual of (TP) and the dual of the transformed problem have the same objective function (row and column equations of (TP) are constraints in the transformed problem). The feasible region of the dual of the transformed problem is contained in that of the dual of (TP) because the constraints associated with artificial edges do not appear in the constraint set of the dual of (TP). Therefore the dual of (TP) is also feasible and unbounded. This completes the proof for case (1).

Case (2). The stage is even. The proof of the Lemma for case (2) is made by interchanging the roles of $T_{\pm}^{*}$ and T_{+}^{*} and following the arguments in case (1). $\triangle$

The conclusion that the method processes sparse (TP) comes easily from Lemmas 12 and 13. The specialization to (AP) is obvious.

7. CONCLUDING REMARKS

1. The dual relaxation procedure can be easily generalized to minimum cost transhipment problems so long as an initial dual feasible basis is available. A transhipment problem is a transportation problem with intermediate nodes, i. e., some of the a_i's and b_j's are zero valued. In addition all edges are directed. In terms of graphs, the transhipment problem is represented by a general (not bipartite) directed graph. The reader acquainted with the dual simplex method for transhipment problems will have no difficulty in verifying that our dual relaxation procedure works provided the candidate edges in a stage are chosen so that all are directed either towards the root or away from the root. The construction of the initial dual feasible tree is not that simple as it is in transportation and assignment problems. Restoring to a phase one, destroys the graphical representation of the problem. On the other hand, adopting a big M method similar to that used for handling sparsity seems more promising.

2. Most of the time in simplex type algorithms for assignment and transportation problems is consumed on comparisons that determine the entering edge. When our algorithm is used, the number of comparisons required per iteration can be substantially reduced if the entering edges in a stage are chosen as follows. For every row node $i \in T_{-}$ set $\delta_i = \min\{w_{ij}(T) : j \in T_{+}\}$ and choose the entering edge by the relation $w_{gh}(T) = \min\{\delta_i : i \in T_{-}\}$. At next iteration for $i \in T'_{-}$ find $\delta'_j = \min\{\delta_i, \min\{w_{ij}(T') : j \in T_1\}\}$ where T_1 is the candidate subtree cut off from T. Then $w_{gh}(T') = \min\{\delta'_i : i \in T'_{-}\}$. When

this trick is employed the algorithm for (TP) becomes $O(k\,m\,n)$ [the signature method becomes $O(kn^2)$], where k is the number of stages. It is therefore desirable to have k as small as possible. This fact somehow implies that the stages must be large, i. e., the number of candidate edges in each stage must be relatively big.

3. There is another reason to believe that the algorithms may perform well provided the stages are large. Examine for example, what is happening during a stage, say an odd stage. At the first iteration, it is $\delta = \min\{w_{ij}: i \in T_-, j \in T_+\}$. If the first candidate subtree T_1 is cut off by a classical dual simplex algorithm the per unit improvement of the objective function is $\delta' = \min\{w_{ij}: i \in T_1, j \in T \sim T_1\}$. Since $T_1 \subset T_-$, $\delta \geqq \delta'$ meaning that the real improvement of the objective function is larger when our dual relaxation procedure is employed. Examine also, the effect of the new monotonic property in subsequent iterations. In particular, the per unit improvement at the last iteration of a stage is larger (probably much larger) than it would be if the same candidate subtree is cut off (from the initial tree in the stage) by a purely dual method. Clearly, safe conclusions cannot be derived from such comparisons since different algorithms follow different paths to the optimal solution. However, our arguments indicate that the per iteration improvement of the objective function is probably (in average) larger with our algorithms.

4. We described the algorithms in such a way that the deleted edge is always a candidate edge. However, this is not necessary. The algorithms work provided (*a*) the objective function does not decrease and (*b*) the first tree at the next stage is dual feasible. It is easy to see that (*b*) is quaranteed by deleting any edge belonging to $\Phi(g, h) \cap T_-$ and (*a*) by deleting either a primal feasible edge with label minus or a primal infeasible edge with label plus. From the computational point of view it seems promising to enlarge as much as possible the set of edges eligible to be deleted. Then, the edge to be deleted can be chosen to be the best one among all eligible edges. Many criteria for choosing the best edge can be used. For example choose the edge inferring the maximal improvement of the objective function. We have already been able to take care of this problem. Actually, we have constructed an algorithm for the transportation problems consisting of a single stage. This work will be the subject of a forthcoming paper.

5. It is well known that simplex type algorithms for network flow problem suffer from degeneracy (stalling), particularly when the data are integers. However, there is no known pivoting rule that prevents cycling in the network dual simplex method analogous to the elegant cycling free rules of the network

primal simplex methods as it is for example Cunningham's [7] method of strong feasible trees. Moreover, it is not quite so obvious that cycling free pivoting rules for general linear programming problems can be easily adapted to our dual relaxation method. This area seems to be promising for future research.

6. Enlarging the stages seems to be promising from the computational point of view. A question coming to someone's mind is if enlargements can be done by combining the two types of stages. In other words, does the method work when the candidate edges in a stage are both even and odd edges? Unfortunately, the answer is no as long as the entering edge is chosen in the way we have described. The reason is that the first tree at the next stage may be not dual feasible. However, other rules for choosing the entering edge may work.

REFERENCES

1. M. L. BALINSKI, *Signatures des points extrêmes du polyhèdres dual du problème de transport*, C.R. Acad. Sci. Paris, 296, Série I, pp. 456-459, 1983.
2. M. L. BALINSKI, *Signature Methods for the Assignment Problem*, Operations Research, 33, 1985, pp. 527-537.
3. M. L. BALINSKI, *A Competitive (dual) Simplex Method for the Assignment Problem*, Mathematical Programming, 34, 1986, pp. 125-141.
4. M. L. BALINSKI and R. E. GOMORY, *A Primal Method for the Assignment and Transportation Problems*, Management Science, 10, 1964, pp. 578-593.
5. R. BARR, F. GLOVER and D. KLINGMAN, *The Alternating path Basis Algorithm for Assignment Problems*, Mathematical Programming, Vol. 13, 1977, pp. 1-13.
6. D. BERTSEKAS, *A new algorithm for the Assignment Problem*, Mathematical Programming, Vol. 21, 1981, pp. 152-171.
7. W. H. CUNNINGHAM, *A Network Simplex Method*, Mathematical Programming, Vol. 11, 1976, pp. 105-116.
8. W. H. CUNNINGHAM and A. B. MARSH, III, *A Primal Algorithm for Optimum Matching*, Mathematical Programming Study, 8, 1978, pp. 50-72.
9. E. A. DINIC and M. A. KRONRAD, *An Algorithm for the Solution of the Assignment Problem*, Soviet Mathematics Doklady, Vol. 10, 1969, pp. 248-264.
10. J. EDMONDS and R. M. KARP, *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems*, Journal of the Association for Computing Machinery, Vol. 19, 1972, pp. 248-264.
11. L. R. FORD and D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, 1962.
12. D. GOLDFARB, *Efficient dual Simplex Methods for the Assignment Problem*, Mathematical Programming, Vol. 33, 1985, pp. 127-203.
13. M. S. HUNG, *A Polynomial Simplex Method for the Assignment Problem*, Operations Research, Vol. 31, 1983, pp. 595-600.

14. M. S. HUNG and W. O. ROM, *Solving the Assignment Problem by Relaxation*, Operations Research, Vol. 28, 1980, pp. 969-982.
15. H. W. KUHN, *The Hungarian Method for the Assignment Problem*, Naval Research Logistics Quarterly, Vol. 2, 1955, pp. 83-97.
16. E. LAWLER, *Combinatorial Optimization, Network and Matroids*, Holt, Rinchart and Winston, New York, 1976.
17. J. MUNKRES, *Algorithms for the Assignment and Transportation Problems*, S.I.A.M. Journal, Vol. 5, 1957, pp. 83-97.
18. E. ROOHY-LALEH, *Improvement to the Theoretical Efficiency of the Network Simplex Method*, Ph. D. Thesis, Carleton University, 1981.
19. V. SRINIVASEN and G. L. THOMSON, *Cost Operator Algorithms for the Transportation Problem*, Mathematical Programming, Vol. 12, 1977, pp. 372-391.
20. G. L. THOMSON, *A Recursive Method for the Assignment Problems*, Annals of Discrete Mathematics, Vol. 11, 1981, pp. 319-343.
21. N. TOMIZAWA, *On some Techniques Usefull for Solution of Transportation Network Problems*, Network, Vol. 1, 1972, pp. 173-194.