REVUE FRANÇAISE D'AUTOMATIQUE, D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

KONSTANTINOS PAPARRIZOS

A non improving simplex algorithm for transportation problems

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 30, n° 1 (1996), p. 1-15.

<http://www.numdam.org/item?id=RO_1996__30_1_1_0>

© AFCET, 1996, tous droits réservés.

L'accès aux archives de la revue « Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/legal.php). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme Numérisation de documents anciens mathématiques
http://www.numdam.org/

Recherche opérationnelle/Operations Research
(vol. 30, n° 1, 1996, pp. 1-15)

A NON IMPROVING SIMPLEX ALGORITHM FOR TRANSPORTATION PROBLEMS (*)

by Konstantinos PAPARRIZOS ([1])

Communicated by Jacques A. FERLAND

Abstract. – *A simplex type algorithm for the Transportation Problem (TP) is presented. TPs with total demand D, m supply and n demand nodes are shown to be solved at most* $O(\mu(m+n)D)$ *time and in at most* $\mu D - \mu(\mu-1)/2$ *iterations, where* $\mu = \min\{m, n\}$. *The algorithm can be initialized with a solution that is neither primal nor dual feasible. The objective function decreases until a dual feasible solution is constructed and then it starts increasing.*

Keywords: Transportation problem, simplex method, pivoting, strong bases, forest algorithms, exterior point algorithms.

Résumé. – *Un algorithme du type simplexe pour le problème de transport est présenté. Nous prouvons que cette sorte de problèmes a demande totale D avec n points de demandes et m points d'approvisionnements est résolu en temps* $O(\mu)m+n)D)$ *et avec* $\mu D - \mu(\mu-1)/2$ *itérations au plus,* $\mu = \min\{m, n\}$. *L'algorithme peut être initialisé avec une solution qui est ni primale, ni duale réalisable. La fonction objective décroît jusqu'à l'obtention d'une solution duale réalisable et puis commence à accroître.*

Mots clés : Problème de transport, méthode du simplex, pivot, bases portes, algorithmes de Forest, algorithmes de point extérieur.

1. INTRODUCTION

Recently, a number of a new type of Exterior Point Simplex Algorithms (EPSA) for network flow problems [1, 10 11] and the general linear programming problem [3, 12] have been developed Contrary to the well known "Criss-Cross" method [13, 14], which is a non improving exterior point simplex algorithm employing a completely combinatorial pivoting rule, the new algorithms improve the objective function. The dual in nature EPSAs are initialized with a dual feasible basic solution. Surprisingly enough, they employ a pivoting rule, which is similar to that of the primal simplex

(*) Received July 1991.

([1]) University of Macedonia, Department of Applied Informatics, 156 Egnatia Str, P.O. Box 1591, 540 06 Thessaloniki, Greece.

Recherche opérationnelle/Operations Research, 0399-0559/96/01/$ 4.00
© AFCET-Gauthier-Villars

algorithm in the sense that the variable entering the basis is first chosen and then the leaving one is chosen. A consequence of the pivoting rule is that dual feasibility may very well be destroyed in intermediate iterations. Preliminary computational results on assignment problems indicate that EPSAs compare favorably to other approaches, *see* [1].

The nature of EPSAs raise the following question, which is stated in [10]. Is there an EPSA that can be initialized with a solution that is neither primal nor dual feasible? In this paper we present an algorithm of this kind for the Transplantation Problem (TP). It is shown that transplantation problems with total demand D, m supply and n demand notes are solved in at most $O(\mu)m + n)\, D)$ elementary operations and in at mos $\mu D - \mu(\mu - 1)/2$ iterations, where $\mu = \min\{m, n\}$.

The algorithm uses forests instead of trees. If artificial variables are introduced to the primal problem, the algorithms can be considered as a simplex type method updating trees. In that case, the algorithm is in fact a Phase I. However, contrary to the known Phase I methods, which are only used to construct a feasible solution to initialize Phase II, our Phase I algorithm is used to solve the original TP. Although the sum of the artificial variables is minimized, the original objective function is also considered. If the initial solution is not dual feasible, the algorithm is directed towards finding a dual solution. During this search the objective function is minimized. If the first dual solution constructed by the algorithm is not an optimal solution to the TP, the algorithm starts a new search for finding the optimal solution. During this second search the objective function value increases from iteration to iteration and dual feasibility is destroyed to be restored again only when optimallity is reached. All these characteristics of the algorithm come from the fact that the reduced cost of the incoming arc increases from iteration to iteration. If the reduced cost is negative (positive), the objective function of the TP decreases (increases). Dual feasibility is reached for the first time when the reduced cost of the incoming arc becomes zero valued.

When our algorithm is specialized to assignment problems it becomes a generalization of the signature method presented in [1], which imposes no restriction on the value of the incoming arc reduced cost. Signature methods for assignment problems, *see* [4, 5 and 8], are $O(n^3)$ dual simplex algorithms. Another algorithm similar to ours is the primal simplex algorithm developed by Akgül [2]. Akgül's algorithm consists of stages. During the iterations of a stage the reduced cost of the incoming arc increases. Our

algorithm generalizes Akgül's algorithm in two aspects. It consists of a single stage and permits the reduced cost of the incoming arc to be unrestricted in value.

The remaining of the paper is organized as follows. Next section is devoted to preliminaries. The algorithm is formally described in Section 3 and its correctness and complexity are shown in Section 4. In the last section some properties of the algorithm are described and some computational efficiency matters are discussed.

2. PRELIMINARIES

The Transportation Problem (TP) can be mathematically stated as follows:

$$\text{TP} \quad \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = a_i, \qquad i = 1, 2, \ldots, m,$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \qquad j = 1, 2, \ldots, n,$$

$$x_{ij} \geq 0, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n,$$

where a_i and b_j are positive integers such that $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = D$.

A *basic solution* of (TP) is a spanning tree of the bipartite graph $G = \{I, J, E\}$, where $I = \{1, 2, \ldots, m\}$ is the set of *row nodes*, $J = \{1, 2, \ldots, n\}$ is the set of *column nodes* and $E = \{(i, j) : i \in I \text{ and } i \in J\}$ is the arc set. An arc $(i, j) \in E$ is directed from the row node i to the column node j. Associated with a spanning tree T there are the *primal variables* $x_{ij}(T)$, computed from the relations $x_{ij}(T) = 0$ for $(i, j) \notin T$, and the *dual variables* $u_i(T)$, $i \in I$, and $v_j(T)$, $j \in J$, computed from the relations $u_i(T) + v_j(T) = c_{ij}$, $(i, j) \in T$. An arc $(i, j) \in T$ is called *degenerate, positive* or *negative*, if $x_{ij}(T)$ is zero, positive or negative, respectively.

Let T be a spanning tree rooted at a node r. The root can be either a row or a column node. An arc $(i, j) \in T$ is *upward*, if it belongs to the path joining the root with the row node i. Otherwise, it is a *downward* arc.

DEFINITION 1: (Cunningham [6]). A rooted tree T is called a strong tree or a strong basis if every degenerate arc of T is downward.

A *spanning forest* F of G is a set of subtrees of G that span all nodes of G. Given a forest F we can associate values to the primal variables $x_{ij}(F)$, $(i, j) \in E$ and to the dual variables $u_i(F)$, $i \in I$ and $v_j(F)$, $j \in J$. It is easily seen that if $x_{ij}(F)$, $u_i(F)$ and $v_j(F)$ satisfy $x_{ij}(F) = 0$ for $(i, j) \notin F$ and $u_i(F) + v_j(F) = c_{ij}$ for $(i, j) \in F$, then complimentaritly slackness condition holds, *i.e.*, it is $x_{ij}(F)\, w_{ij}(F) = 0$ for each arc $(i, j) \in E$, where $w_{ij}(F) = c_{ij} - u_i(F) - v_j(F)$, is the *reduced cost* associated with the arc (i, j). A forest F is called *primal feasible*, if the primal values $x_{ij}(F) \geq 0$, $(i, j) \in E$, satisfy all the equations of (TP). F is called *dual feasible*, if $w_{ij}(F) \geq 0$ for all $(i, j) \in E$. As the primal and dual solutions associated with a forest satisfy the complimentarity slackness condition, a forest that is both primal and dual feasible is an optimal solution to (TP). The trees that constitute a forest are called *component trees* of the forest.

DEFINITION 2: A spanning forest F of the graph G is called a strong forest if every component (tree) of F is a strong tree.

A consequence of Definition 2 is that every component tree of a strong forest is rooted. A forest F' is adjacent to a forest F, if $F' = F \backslash (k, l) \cup (g, h)$, where $(k, l) \in F$ and $(g, h) \notin F$ or $F' = F \cup (g, h)$, $(g, h) \notin F$.

3. ALGORITHM DESCRIPTION

The algorithm presented in this section moves between adjacent strong forests F satisfying $x_{ij}(F) \geq 0$. It stops when a primal feasible forest is constructed. We leave to the reader to describe the equivalent simplex algorithm that updates trees, *i.e.*, the on that works one the phase I problem constructed by introducing artificial variables to (TP).

The algorithm partitions every strong forest into two subforests F^S and F^D. Forest F^S is called the *surplus* subforest while the subforest F^D is called the *deficit* subforest. The surplus (deficit) forest contains all the components of F, which are rooted at a row (column) node. The solution $x_{ij}(F) \geq 0$ computed by the algorithm satisfies all the equations of (TP) associated with the nodes that are not roots. Given a strong forest F and a row (column) node $p\,(q)$ we define the quantities

$$A_p(F) = a_p - \Sigma\{\, x_{pj}(F) : (p, j) \in F \,\}\,(B_q(F)$$
$$= b_q - \Sigma\{\, (x_{iq}(F) : (i, q) \in F \,\}).$$

Clearly, F is primal feasible if $A_p(F) = 0$ for $p \in I$ and $B_q(F) = 0$ for $q \in J$. The quantities $A_p(F)$ and $B_q(F)$ are in fact the values of the artificial variables associated with the equations of (TP) corresponding to nodes p and q, respectively. As the algorithm is of simplex type the reader will have no difficulty in understanding it from the following formal description.

STEP 0. (*Initialization*). Start with the initial forest F_0, which contains no arc. Set $F_0^S = I$ and $F_0^D = J$. Also, set $A_i(F_0) = a_i$, $i = 1, 2, \ldots, m$ and $B_j(F_0) = b_j$, $j = 1, 2, \ldots, n$. Make every node a root and set $t = 0$.

STEP 1. (*Entering arc determination*). If $F_t^S = \varnothing$, STOP (F_t is an optimal forest to (TP)). Otherwise, compute

$$\delta_t = w_{gh}(F_t) = \min\{ w_{ij}(F_t) : i \in F_t^S, i \in F_t^D \}$$

and adjoin arc (g, h) to F_t.

STEP 2. (*Leaving arc determination*). Let $p(q)$ be the root of the component tree containing row node g (column node h) and denote by P_t the unique path of $F_t \cup (g, h)$ joining the root nodes p and q. Let also P_t^+ (P_t^-) be the set of arcs of P_t having the same (opposite) direction with the incoming arc (g, h). Compute

$$x_{kl}(F_t) = \min\{ x_{ij}(F_t) : (i, j) \in P_t^- \} \tag{1}$$

and

$$\varepsilon_t = \min\{ A_p(F_t), x_{kl}(F_t), B_q(F_t) \}. \tag{2}$$

In case of ties in relation (1), arc (k, l) is the first eligible arc met when the path P_t is traced from p to q. Also, in case of ties in relation (2), ε_t is the first eligible quantity among $A_p(F_t)$, $x_{kl}(F_t)$ and $B_q(F_t)$ examined in this order.

STEP 3. Update F_t^S, F_t^D according to the following cases:

Case 1: (*a*) $\varepsilon_t = B_q(F_t)$, (*b*) $\varepsilon_t = x_{kl}(F_t)$ and $(k, l) \in F_t^D$,

Case 2: (*a*) $\varepsilon_t = A_p(F_t)$, (*b*) $\varepsilon_t = x_{kl}(F_t)$ and $(k, l) \in F_t^S$.

Case (1*a*): Let F_t^* be the component containing column node h. Set $F_{t+1}^D = F_t^D \backslash F_t^*$ and $F_{t+1}^S = F_t^S \cup F_t^* \cup (g, h)$. The root q is no longer a root.

Case (1*b*): Let F_t^* be the subtree cut off from the component tree containing column node h when (k, l) is deleted from F_t. Set $F_{t+1}^D = F_t^D \backslash F_t^*$ and $F_{t+1}^S = F_t^S \cup F_t^* \cup (g, h)$. The roots do not change.

Case (2*a*): Let F_t^* be the component tree containing row node g. Set $F_{t+1}^D = F_t^D \cup F_t^D \cup (g, h)$ and $F_{t+1}^S = F_t^S \backslash F_t^*$. The root p is no longer a root.

Case (2*b*). Let F_t^* be the subtree cut off from the component tree containing g when arc (k, l) is deleted. Set $F_{t+1}^D = F_t^D \cup F_t^D \cup (g, h)$ and $F_{t+1}^S = F_t^S \backslash F_t^*$. The roots do not change.

STEP 4. (*Computation of new values of variables*).

(*a*) Update the primal variables by the relations:

$$\begin{aligned} x_{ij}(F_{t+1}) &= x_{ij}(F_t) + \varepsilon_t, && \text{if} \quad (i, j) \in P_t^+, \\ &= x_{ij}(F_t) - \varepsilon_t, && \text{if} \quad (i, j) \in P_t^-, \\ &= x_{ij}(F_t) && \text{otherwise.} \end{aligned} \tag{3}$$

(*b*) Set

$$\begin{aligned} A_i(F_{t+1}) &= A_i(F_t) - \varepsilon_t, && \text{if} \quad i = p, \\ &= A_i(F_t), && \text{if} \quad \text{otherwise, and} \\ B_j(F_{t+1}) &= A_j(F_t) - \varepsilon_t, && \text{if} \quad j = q, \\ &= A_j(F_t), && \text{otherwise.} \end{aligned}$$

(*c*) Update dual variables as follows:

Dual updates for Case 1:

$$u_i(F_{t+1}) = u_i(F_t) - \delta_t, \quad \text{if} \quad i \in F_t^*,$$
$$v_j(F_{t+1}) = v_j(F_t) + \delta_t, \quad \text{if} \quad j \in F_t^*,$$
$$u_i(F_{t+1}) = u_j(F_t) \quad \text{and} \quad v_j(F_{t+1}) = v_j(F_t), \text{ otherwise.}$$

Dual updates for Case 2:

$$u_i(F_{t+1}) = u_i(F_t) + \delta_t, \quad \text{if} \quad i \in F_t^*,$$
$$v_j(F_{t+1}) = v_j(F_t) - \delta_t, \quad \text{if} \quad j \in F_t^*,$$
$$u_i(F_{t+1}) = u_j(F_t) \quad \text{and} \quad v_j(F_{t+1}) = v_j(F_t), \text{ otherwise.}$$

(*d*) Set $t \leftarrow t + 1$ and go to Step 1.

Observe that dual updates take place only in the cut off subtree F_t^*. It is also easily verified that $w_{gh}(F_{t+1}) = 0$ and that complementarity slackness holds throughout the computation.

The algorithm performs four types of iterations that correspond to the four Cases 1*a*, 1*b*, 2*a*, and 2*b* of Step 3. In the iterations of type 1*a* or 1*b* (2*a* or 2*b*) a tree component or a subtree is transferred from the surplus (deficit) forest to the deficit (surplus) forest. Figure 1 illustrates a type 1*b* iteration.

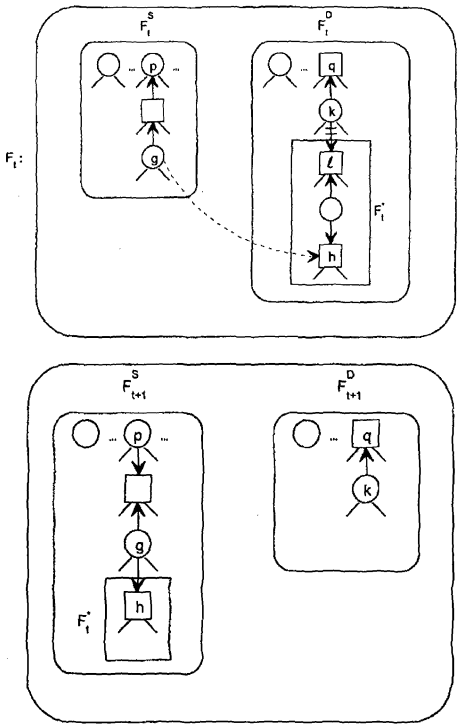

Figure 1. – An iteration of type 1*b*.

4. ALGORITHM CORRECTNESS AND COMPLEXITY

The aim of this section is to show the following result.

THEOREM 1: *The algorithm solves (TP) correctly in at most* $O(\mu(m+n)D)$ *elementary operations, where* $\mu = \min\{m, n\}$.

The proof of the theorem is based on the following lemmas. We first show correctness of the algorithm.

LEMMA 2: *Let* F *be the last forest generated by the algorithm. Then* F *is primal feasible.*

Proof: The pivoting rule and the way primal variables are updated assure that $x_{ij}(F) \geq 0$, $(i, j) \in E$. The algorithm has stopped because $F^S = \varnothing$. This implies that $F(x)$ satisfies all the equations of (TP) that correspond to

row nodes. This in turn implies that the equations of (TP) that correspond to column nodes are also satisfied. Hence, F is primal feasible. □

The following two lemmas will be used to show that a forest F is dual feasible in the case $F^S = \emptyset$. These lemmas generalize lemmas 2 and 3 of [1] so that δ_t can be negative. Although some parts of the proofs are similar to the ones presented in [1], we present complete proofs for the sake of completeness.

In the following lemmas we will use the following sets of arcs

$$A(F) = \{ (i, j) \in E : i \in F^S, i \in R \text{ and } j \in F^D, j \in C \}$$
$$B(F) = \{ (i, j) \in E : i \in F^D, i \in R \text{ and } j \in F^S, j \in C \},$$

where R is the set of row nodes and C is the set of column nodes. Equivalently, the set $A(F)$ $(B(F))$ contains the arcs directed from a row node of F^S (F^D) to a column node of F^D (F^S).

Let

$$\delta'_t = \min \{ w_{ij}(F_t) : (i, j) \in B(F_t) \}, \quad \text{if} \quad B(F_t) \neq \emptyset$$
$$= \infty \quad \text{if} \quad B(F_t) = \emptyset.$$

LEMMA 3: *If F_t^D and F_t^S are dual feasible and $\delta_t + \delta'_t \geq 0$ then $\delta_{t+1} + \delta'_{t+1} \geq 0$.*

Proof: We prove the lemma only for iterations of type 1. The proof for iterations of type 2 is omitted as being completely analogous.

Assume that F_{t+1} is obtained from F_t by an iteration of type 1. Let

$$A_1(F_t) = \{ (i, j) \in E : i \in F_t^S, j \in F_{t+1}^D \}$$

and

$$A_2(F_t) = \{ (i, j) \in E : i \in F_t^*, j \in F_{t+1}^D \}.$$

Then $\delta_{t+1} = \min \{ \varepsilon_1, \varepsilon_2 \}$, where

$$\varepsilon_1 = \min \{ w_{ij}(F_{t+1}) : (i, j) \in A_1(F_t) \}$$

and

$$\varepsilon_2 = \min \{ w_{ij}(F_{t+1}) : (i, j) \in A_2(F_t) \}.$$

As $F_{t+1}^D \subseteq F_t^D$, we have $A_1(F_t) \subseteq A_2(F_t)$. As it is

$$w_{ij}(F_{t+1}) = w_{ij}(F_t) \quad \text{for} \quad (i, j) \in A_1(F_t),$$

we have $\varepsilon_1 \geq \delta_t$. For the arcs $(i, j) \in A_2(F_t)$ we have

$$w_{ij}(F_{t+1}) = c_{ij} - (u_i(F_t) - \delta_t) - v_j(F_t) = w_{ij}(F_t) + \delta_t.$$

As nodes $i,\ j \in F_t^D$ and F_t^D is dual feasible, we have $w_{ij}(F_t) \geq 0$ and, hence, $\varepsilon_2 \geq \delta_t$. Clearly, $\delta_{t+1} = \min\{\varepsilon_1,\ \varepsilon_2\} \geq \delta_t$.

Let

$$B_1(F_t) = \{(i,\ j) \in E : i \in F_{t+1}^D,\ j \in F_t^S\}$$

and

$$B_2(F_t) = \{(i,\ j) \in E : i \in F_{t+1}^D,\ j \in F_t^*\}.$$

Then $\delta'_{t+1} = \min\{\varepsilon'_1,\ \varepsilon'_2\}$, where

$$\varepsilon'_1 = \min\{w_{ij}(F_{t+1}) : (i,\ j) \in B_1(F_t)\}$$

and

$$\varepsilon'_2 = \min\{w_{ij}(F_{t+1}) : (i,\ j) \in B_2(F_t)\}.$$

As $B_1(F_t) \subseteq B_2(F_t)$ and $w_{ij}(F_{t+1}) = w_{ij}(F_t)$ for all $(i,\ j) \in B_1(F_t)$, we have $\varepsilon'_1 \geq \delta'_t \geq -\delta_t$ (by the induction hypothesis. For $(i,\ j) \in B_2(F_t)$ we have

$$w_{ij}(F_{t+1}) = c_{ij} - u_i(F_t) - (v_j(F_t) + \delta_t) = w_{ij}(F_t) - \delta_t.$$

As nodes $i,\ j \in F_t^D$ and F_t^D is dual feasible, we have $w_{ij}(F_t) \geq 0$ and hence, $\varepsilon_2 \geq -\delta_t$. Therefore $\delta'_{t+1} \geq -\delta_t$.

Now we can easily show that, $\delta_{t+1} + \delta'_{t+1} \geq \delta_t - \delta_t = 0$. □

LEMMA 4: *If F_t^S and F_t^D are dual feasible and $\delta_t + \delta'_t \geq 0$, then F_{t+1}^S and F_{t+1}^D are also dual feasible.*

Proof: As in Lemma 3 we are restricted on iterations of type 1.

As $F_{t+1}^D \subseteq F_t^D$ and F_t^D is dual feasible, F_{t+1}^D is also feasible $(w_{ij}(F_{t+1}) = w_{ij}(F_t))$.

As $F_{t+1}^S = F_t^S \cup F_t^* \cup (g,\ h)$ and F_t^S, F_t^* are dual feasible, it remains to examine edges $(i,\ j)$ such that $i \in F_t^S$, $j \in F_t^*$ or $i \in F_t^*$, $j \in F_t^S$. If $i \in F_t^S$ and $j \in F_t^*$, then

$$w_{ij} = (F_{t+1}) = c_{ij} - u_i(F_t) - (u_j(F_t) + \delta_t) = w_{ij}(F_t) - \delta_t.$$

As $(i,\ j) \in A(F_t)$, we have $w_{ij}(F_t) \geq \delta_t$ and, hence,

$$w_{ij}(F_{t+1}) \geq \delta_t - \delta'_t = 0$$

If $i \in F_t^*$ and $j \in F_t^S$, then

$$w_{ij} = (F_{t+1}) = c_{ij} - (u_i(F_t) - \delta_t)\delta_t) - u_j(F_t) = w_{ij}(F_t) + \delta_t.$$

As $(i, j) \in B(F_t)$, we have $w_{ij}(F_t) \geq \delta'_t$ and, hence

$$w_{ij}(F_{t+1}) \geq \delta_t + \delta'_t \geq 0$$

(by the induction hypothesis). □

LEMMA 5: *The last forest F generated by the algorithm is otptimal to (TP).*

Proof: Using Lemmas 3 and 4 and a simple induction on the number of iterations t, we show that F^D is dual feasible. As for the last forest F it is $F = F^D$ (because $F^S = \varnothing$), F is dual feasible. By Lemma 2, F is also primal feasible. As complementarity slackness holds, F is optimal to (TP). □

Our analysis so far showed that the algorithm is correct. The following lemmas will be used to device upper bounds on the number of iterations and the number of elementary operations.

LEMMA 6: *Every forest generated by the algorithm is a strong forest.*

Proof: The proof is by induction on the number of iterations. The initial forest F_0 contains no arc. Hence, it is a strong forest. Assume now that F_t is a strong forest. In order to show that F_{t+1} is also a strong forest it suffices to examine the arcs of P_t, becaue the arcs $(i, j) \notin P$ change neither status (they are upward or downward in F_t and remain upward or downward in F_{t+1}) nor value *i.e.*, $x_{ij}(F_{t+1}) = x_{ij}(F_t)$ for $(i, j) \notin P$, *see* Step 4*a* in the algorithm description.

Let $e \in F_{t+1}$, be an upward arc. There are two cases to be considered, $\varepsilon_1 > 0$ and $\varepsilon_t = 0$.

Case 1. $\varepsilon_t = x_{kl}(F_t) > 0$. There are four subcases to be considered, one for each type of iteration. We first prove the lemma for the type 1*b* iterations. In that case $(k, l) \in F_t^D$.

If e is an arc of the subpath of P_t joining the root row node $p \in F_t^S$ and the column node $h \in F_t^D$, then $e \in P_t^-$. From the pivoting rule described in Step 2 we conclude that $x_e(F_t) > \varepsilon_t$. Otherwise, *i.e.*, if $x_e(F_t) = \varepsilon_t = x_{kl}(F_t)$, (k, l) would not be the outcoming arc at iteration t. From relation (3) we have $x_e(F_{t+1}) = x_e(F_t) - \varepsilon_t > 0$, as desired.

If e is an arc of the path joining row node $k \in F_t^D$ and the root column node q then $e \in P_t^+$. As $x_e(F_t) \geq 0$ and $\varepsilon_t > 0$ we have from relations (3) that $x_e(F_{t+1}) = x_e(F_t) + \varepsilon_t > 0$.

The proof for type $1a$ iterations is identical except that the path joining nodes k and q does not exist. The proof for the iterations of type $2a$ and $2b$ is similar.

Case 2. $\varepsilon_t = 0$. We show first that $A_p(F_t) > 0$. This implies that iteration t is not of type $2a$. Assume not, *i.e.*, assume that $A_p(F_t) = 0$. Consider the iteration $0 \leq r \leq t-1$ such that $A_p(F_r) > 0$ and $A_p(F_{r+1}) = 0$. As $p \in F_t^S$ is a root node, iteration r was not of type $2a$. Now, relation $A_p(F_{r+1}) = 0$ implies that $A_p(F_r) = \varepsilon_r$, a contradiction. It is also easily verified that iteration t is not of type $2b$. Hence, iteration t is either of type $1a$ or of type $1b$. In both cases it is easily seen that no degenerate arc of F_t becomes an upward arc of F_{t+1}. Hence, every degenerate arc of F_{t+1} is downward. □

A pivot operation is called degenerate if $\varepsilon_t = 0$. We give now some additional definitions, which will be used to derive an upper bound on the number of consecutive degenerate pivots.

DEFINITION 3: A column node j is called degenerate if it is either a root (of the deficit forest) such that $B_j(F) = 0$ or a non-root node such that $x_{ij}(F) = 0$, where (i, j) is the unique downward arc incident to j. Degenerate row nodes are similarly defined.

DEFINITION 4: The level $L(F)$ of a strong forest F is given by the relation: $L(F) = \Sigma(B_q(F) : q \text{ is a root of } F_t^D)$.

It is easily verified that $L(F) = \Sigma(A_p(F) : p \in I \text{ is a root of } F^S)$. Also, an easy inductive argument on the number of iterations shows that every component of the surplus forest is rooted at a non-degenerate row node. Consequently, we have $L(F) = 0$ for the last forest F generated by the algorithm.

LEMMA 7: *If F is not a primal feasible strong forest generated by the algorithm, the number of degenerate column nodes of F_t^D is no greater than* $M = \min\{m - 1, n - 1\}$.

Proof: As forest F is strong (by Lemma 6) but not primal feasible, there is at least one (column) root of F^D, say q, such that $B_q(F) > 0$. hence, $M \leq n - 1$. Observe now that if a column node is degenerate, its degree is at least one. Hence, every row node can generate at most one degenerate column node. As F is not primal feasible, F^S contains at least one row node. Consequently, it is also $M \leq m - 1$ and the proof follows. □

Proof of Theorem 1: In Lemma 5 we showed that the last forest is optimal to (TP). We derive now the complexity bound stated in the theorem.

It is easily seen that the level of the forests generated by the algorithm is not increasing. Let F_i be the first forest in level, say r, and F_j, $i < j$, be the first forest in level less than r. From the pivoting rule we easily see that if a degenerate pivot is performed on F_t, $i \leq t \leq j \leq 1$, at least one degenerate column node of F_t^D is transferred to F_{t+1}^S. As by Lemma 7 there are at most M degenerate column nodes in F_i^D, after at most M iterations a strong forest F_{j-1} containing no degenerate column node in F_{j-1}^D is constructed. Then a level reducing iteration of type 2 is performed. Hence, it takes at most $\mu = M + 1 = \min\{m, n\}$ iterations to reduce the level by at least one unit. As the level of the initial strong forest is D, after at most μD iterations a strong forest F of level zero is constructed.

The work required to update a forest is precisely that of updating a tree, *i.e.*, $O(mn)$. This bound comes from the number of comparisons needed to determine the arc to be adjoined. It is well known that all the other work is $O(m+n)$. Hence, we have shown so far that the complexity bound on the number of elementary operations is $O(\mu mn D)$.

In order to derive the bound stated in the theorem, it suffices to show that the local work required to determine all the arcs adjoined in a single level is $O(mn)$. During the iterations in a given level the set $J \cap F_t^D$ is decreasing and the set $I \cap F_t^S$ is increasing. We can store for each fixed $j \in J \cap F_t^D$, the row node $i \in I \cap F_t^S$, where w_{ij} is minimal. After an iteration we only have to compare this value to those w_{kj}, where k has been added to the surplus forest. Hence, every arc is examined at most once. Hence, the work for determining all incoming arcs in a level is $O(mm)$. As the remaining work in the level is $O(\mu(m+n))$, the proof follows. □

In Theorem 1 we have shown that the number of iterations is bounded by μD. In the next theorem a better bound is derived.

THEOREM 8: *The algorithm stops after at most* $\mu D - \mu(\mu-1)/2$ *iterations.*

Proof: It suffice to show the statement:

(*) The number of iterations in level $D-k$, $k = 0, 1, 2, \ldots, D-1$, is at most $k+1$. Indeed, if statement (*) is correct, there are at most $1 + 2 + \ldots + (\mu - 1) = \mu(\mu-1)/2$ iterations in the first $\mu - 1$ levels ($k = 0, 1, \ldots, \mu-2$ and (by Lemma 7 and Theorem 1) at most $\mu(D-\mu+1)$ iterations in the remaining $D-(\mu-1) = D-\mu+1$ levels. Hence, the number of iterations is at most $\mu(\mu-1)/2 + \mu(D-\mu+1) = \mu D - \mu(\mu-1)/2$.

We show now that statement (*) is correct. Let F be the first forest in level $D-k$, $M(F^D)$ be the number of degenerate columns in F^D and $N(F^D)$ the number of row nodes in F^D. It is well known that:

$$\begin{aligned} L(F) &= \Sigma(B_q(F) : q \text{ is a root of } F^D), \\ &= \Sigma(b_j : j \in J \text{ and } j \in F^D) - \Sigma(a_i : i \in I \text{ and } i \in F^D). \end{aligned}$$

Clearly, $\Sigma(b_j : j \in J \text{ and } j \in F^D) \leq D$. As $a_i \geq 1$, we have $\Sigma(a_i : i \in I \text{ and } i \in F^D) \geq N(F^D)$.

From Lemma 7 we have $N(F^D) \geq M(F^D)$ and, hence

$$-\Sigma(a_i : i \in I \text{ and } i \in F^D) \leq -M(F^D).$$

Therefore, we have

$$D - k = L(F) \leq D - M(F^D)$$

and, hence, $M(F^D) \leq k$. By Theorem 1 now the number of iterations in level $D-k$ is bounded by $M(F^D)+1 \leq k+1$ completing the proof. □

In [1], it has been shown that $\delta_t \leq \delta_{t+1}$. It is easily seen that this relation holds for our algorithm as well. Consequently, if $\delta_0 < 0$, the objective function initially decreases until a forest F_t such that $\delta_t \geq 0$ is constructed. In subsequent iterations the objective function decreases.

Closing this section, we point out that the algorithm handles sparse TPs in a very simple way. If the algorithm is applied to a sparse (TP) there might be no arc eligible to be adjoined at some iteration. In this case, it follows easily from the structure of the forest that (TP) has no primal feasible solution.

5. CONCLUDING REMARKS

Besides the obvious problems of extending the algorithm to more general linear problems, the worst case analysis of the current algorithm remains an open problem. It particular, the bound on the number of iterations may not be the best one. In order to illustrate this fact we introduce the notion of the stage of the algorithm. A stage of the algorithm consists of the maximum number of consecutive forests (iterations) F_t, such that $F_t^D \supset F_{t+1}^D$. From the analysis of Section 4, we can easily see that the level may decrease more than once during the iterations of a stage. It is also easily seen that the maximum number of iterations in a stage is n, implying that the complexity of a stages $O(n(m+n))$. Consequently, the determination of the maximum number of stages is crucial in deriving the algorithm complexity.

The computational efficiency of algorithms for TPs (and assignment problems) in highly dependent on the initial solution. As our algorithm improves primal feasibility, the initial solution is the worst starting point (the level is the maximum possible). We can very easily remedy this computational disadvantage as follows. Construct a dual feasible forest F by joining row node i to column node q such that $c_{iq} = \min \{ c_{ij} : j \in J \}$. Make all the column nodes roots and compute $B_j(F)$ for all $j \in J$. If $B_j(F) \geq 0$, the component rooted at j belongs to F^D. The remaining components belong to the surplus forest and they are rooted at properly chosen (row) roots so that they are strong trees (the details are left to the reader). The forest F is in fact the Hung-Rom [7] heuristic extended to TPs.. In [9], it has been shown computationally that the Hung-Rom tree is a good starting heuristic for assignment problems. So, our algorithm can be initialized with a good starting forest. As this modified algorithm possesses all the good computational properties of the efficient algorithm for assignment problems described in [1], we expect that it is also efficient in practice. However, this remains to be verified computationally.

ACKNOWLEDGEMENTS

The author wishes to thank two unknown referees for their constructive comments that greatly improved the presentation of the paper.

REFERENCES

1. H. Achatz, P. Kleinschmidt and K. Paparrizos, A Dual Forest Algorithm for the Assignment Problem, *DIMACS*, 1989, *4*, pp. 1-10.
2. M. Akgül, *A Genuinly polynomial Primal Simplex Algorithm for the Assignment Problem*, SERC Report, IEOR 87-07, 1987, Bilkent University.
3. K. Anstriecher and T. Terlaky, A Monotonic Build-Up Simplex Algorithm for Linear Programming, *Operations Research*, 1994, *42*, pp. 556-561.
4. M. L. Balinski, Signature Methods for the Assignment Problem, *Operations Research*, 1985, *33*, pp. 527-537.
5. M. L. Balinski, A Competitive (Dual) Simplex Method for the Assignment Problem, *Mathematical Programming*, 1986, *34*, pp. 125-141.
6. W. H. Cunningham, A Network Simplex Algorithm., *Mathematical Programming*, 1976, *11*, pp. 105-116.
7. M. S. Hung and W. O. Rom, Solving the Assignment Problem by Relaxation, *Operations Research*, 1980, *28*, pp. 969-982.
8. D. Goldfarb, Efficient Dual Simplex Algorithms for the Assignment Problem, *Mathematical Programming*, 1985, *33*, pp. 969-982.

9. P. KLEINSCHMIDT, C. W. LEE and H. SCANNATH, Transportation Problems Which can be Solved by the Use of Hirsch-paths for the Dual Problems, *Mathematical Programming*, 1987, *37*, pp. 153-168.
10. K. PAPARRIZOS, An Infeasible (Exterior Point) Simplex Algorithm for Assignment Problems, *Mathematical Programming*, 1988, *51*, pp. 45-54.
11. K. PAPARRIZOS, *A Network Exterior Point Algorithm*, Presented at the EURO X Conference on Operational Research, Beograd, Yug, 1989.
12. K. PAPARRIZOS, An Exterior Point Simplex Algorithm for General Linear Problems, *Annals of Operations Research*, 1993, *32*, pp. 497-508.
13. T. TERLAKY, A Convergent Criss-Cross Method, *Math. Oper. und Stat. ser. Optimization*, 1985, *16*, pp. 683-690.
14. S. ZIONTS, The Criss-Cross Method for Solving Linear Programming Problems, *Management Science*, 1969, *15*, pp. 426-445.