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

M. AKGÜL
O. EKIN

A dual feasible forest algorithm for the linear assignment problem

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 25, n° 4 (1991), p. 403-411.

<http://www.numdam.org/item?id=RO_1991__25_4_403_0>

© AFCET, 1991, 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. 25, n° 4, 1991, p. 403 à 411)

A DUAL FEASIBLE FOREST ALGORITHM FOR THE LINEAR ASSIGNMENT PROBLEM (*)

by M. AKGÜL (1) and O. EKIN (1)

Abstract. – *We present a dual feasible forest algorithm for the assignment problem. The algorithm is guided by the signature of a strongly feasible tree and terminates with such a tree. It has the time complexity $O(n^3)$ for dense problems and $O(n^2 \log n + nm)$ for sparse graphs.*

Keywords : Assignment; dual simplex method; signatures; pivoting; strongly feasible trees; average behavior.

Résumé. – *Nous présentons un algorithme dual pour le problème d'affectation. L'algorithme est guidé par la signature d'un arbre fortement réalisable et se termine avec un tel arbre. Il a une complexité temporelle $O(n^3)$ pour les problèmes denses et $O(n^2 \log n + nm)$ pour les graphes creux.*

Mots clés : Affectation; méthode duale du simplex; signatures; pivotage; arbres fortement réalisables; comportement moyen.

Balinski [3] introduced the signature method for the linear assignment problem which requires $O(n^2)$ pivots and $O(n^3)$ time. He later [4] gave a purely dual-simplex algorithm having the same complexity as the signature method. Goldfarb [8] and Akgül [1] gave sequential versions of the above algorithms. Paparrizos [9] introduced a non-dual signature method which solves the n by n assignment problem in at most $O(n^2)$ pivots and $O(n^4)$ time.

Here, we present a dual-feasible signature-guided forest algorithm which terminates with a strongly feasible tree. It is a modification of Paparrizos' algorithm. The algorithm has $O(n^3)$ complexity for dense problems using elementary data structures. For sparse graphs, it has $O(n^2 \log n + nm)$ complexity using Fibonacci-heaps of Fredman and Tarjan [7].

(*) Received August 1990.
(1) Department of Industrial Engineering, Bilkent University, 06533, Bilkent, Ankara, Turkey.

Recherche opérationnelle/Operations Research, 0399-0559/91/04 403 09/$ 2.90
© AFCET-Gauthier-Villars

1. PRELIMINARIES

We will view the assignment problem (AP) as a transshipment problem over a directed bipartite graph $G=(U, V, E)$ with node set $N=U \cup V$, and edge set E. Each edge $e \in E$ is directed from its tail $t(e) \in U$ (source or row node) to its head $h(e) \in V$ (sink or column node), has flow x_e and cost c_e.

For a graph $G=(N, E)$, and disjoint sets X, $Y \subset N$, we let

$$\gamma(X)=\{e \in E: t(e) \in X, h(e) \in X\}$$
$$\delta(X, Y)=\{e \in E: t(e) \in X, h(e) \in Y\}$$

We can cast AP as

$$\min\{cx: Ax=b, x \geqq 0\}$$

where A is the node-edge incidence matrix, and

$$b_u=-1, \qquad u \in U,$$
$$b_v=+1, \qquad v \in V.$$

The dual LP is

$$\max\{yb: y_{h(e)}-y_{t(e)} \leqq c_e, e \in E\}.$$

Reduced cost of edge $e=(i, j)$ is

$$w_e=w_{ij}=c_e-y_j+y_i.$$

Given a tree rooted at node r, $f \in T$ is **reverse** ($f \in R$) if f is directed towards r. Otherwise, it is **forward** ($f \in F$). $\forall v \in N$, $v \neq r$, there is a unique node called the parent of v, $p(v)$ and parent pointers define the tree uniquely. Strongly Feasible Trees (SFT) are introduced by Barr *et al.* [5] and independently by Cunningham [6] to speed up the primal simplex method for AP and to prevent cycling in the network simplex method respectively. SFTs have been used by Akgül [2] and many others, in polynomial primal algorithms. Relevant properties of SFTs can be summarized as follows:

LEMMA 1: *Let T be a spanning tree for the AP rooted at a sink node, r. Then the following are equivalent.*

(i) *T is a SFT,*

(ii) *Every reverse tree edge has flow* 1, *and every forward tree edge has flow* 0,

(iii) $\deg(r)=1$, $\deg(v)=2$, $\forall v \neq r$, $v \in V$ *where* $\deg(.)$ *denotes the degree of the specific node.* □

Clearly (ii) implies that T is primal feasible and (iii) implies that the column signature of T, *i.e.*, the degree sequence of the column (sink) nodes is (2, 2, 2, . . ., 2, 1). Moreover, if any T has column signature as such, then rooted at the node of degree 1, such a T is SFT.

Clearly, a dual-feasible tree which is also SFT is an optimal tree. A signature-guided method changes the tree by linking and cutting edges to obtain a tree having the desired signature, *i.e.*, (2, 2, 2, . . ., 2, 1).

2. THE ALGORITHM

First, we will describe Paparrizos' [9] algorithm in our notation. His algorithm works with, what we call, **layers**.

Initial tree is dual-feasible and is rooted at a source node and all sink nodes of degree 1 are attached to this source node, *i.e.*, Balinski tree. A **layer** consists of two parts: **decompose** and **link**. To **decompose** a tree, a sink node of degree $\geqq 3$ which is minimal in distance to the root is identified. If there is no such node, then T is SFT and hence it is optimal. Let $v \in V$ be such a node. Then the edge $(p(v), v)$ is deleted and the cutoff subtree rooted at v is identified as a "candidate tree", and is denoted as say, T_v. The process is continued until the tree rooted at r contains no sink nodes of degree $\geqq 3$. The tree rooted at r is called T_+ and T_- is the collection of candidate trees. The **link** part of the algorithm is as follows.

while $T_- \neq \varnothing$ **do**
  begin
    $e \leftarrow \arg\min \{ w_e : e \in \delta(T_-, T_+) \}$
    $\varepsilon \leftarrow w_e$
    Let $t(e) \in T_k$
    $y_v \leftarrow y_v - \varepsilon \; \forall v \in T_k$
    $T_- \leftarrow T_- \setminus T_k$
    $T_+ \leftarrow T_+ + T_k + e$
  end { **while** }

The main invariant during **link** is that the subtree T_+ is dual-feasible, *i.e.*, edges in $\gamma(T_+)$ are dual-feasible. Consequently, when a **layer** is finished, the new tree is dual-feasible. Since **layer algorithm** is continued until T is SFT, the algorithm stops with an optimal tree. The pivot bound is $O(n^2)$ but the

number of layers also has the same bound. This results in an $O(n^4)$ algorithm. Moreover, during a layer, dual-feasibility may be violated.

In the new algorithm, we abandon the layer concept altogether. After linking a subtree to T_+ via sink node v, instead of linking other trees in T_- to T_+, we apply *decompose* if possible. So our algorithm performs a simpler form of *link* and decompose alternatively (some decompose could be vacuous). We divide the whole process into **stages** which will facilitate an efficient implementation of the algorithm.

We also make dual variable changes on the whole T_- rather than on a subtree of it. Consequently, we obtain a dual feasible algorithm with the state of the art complexity.

Now, we describe the new algorithm.

For a tree (forest) T, let $\sigma_1 = \sigma_1(T)$, σ_2, σ_3 be the number of sink nodes of degree 1, degree 2 and degree at least 3 respectively. Hence, T is SFT if and only if $\sigma_1 = 1$, $\sigma_2 = n-1$, $\sigma_3 = 0$. The **level** of a tree is $\sigma_1(T)$. Our algorithm works with **stages** through each of which σ_1 is reduced by 1. The computational cost of a stage will be $O(n^2)$ for the dense case and $O(n \log n + m)$ for the sparse case.

We start with the well-known "Balinski-tree" rooted at a source node r. We then apply **decompose**. Thus, we obtain T_+, and $T_- = \bigcup_{i=1}^{l} T_i$ and $l \leqq \sigma_3$.

Our **link** routine (at say k-th iteration) is as follows:

begin
$e = (u, v) = \arg\min \{ w_e : e \in \delta(T_-, T_+) \}$
Let $\varepsilon = w_e$ and $t(e) = u \in T_q$
$y'_z \leftarrow y_z - \varepsilon \forall z \in T_-$
$T'_+ \leftarrow T_+ + T_q + e$
$T'_- \leftarrow T_- \setminus T_q$
end

where $T = T_- \cup T_+$ is the forest at the k-th iteration and $T' = T'_- \cup T'_+$ is the forest obtained after the k-th link.

A *link* followed by a, possibly vacuous, *decompose* is called a **pivot**. Let $d(v)$ be the degree of v in T'_+. Depending on $d(v)$ where $v = h(e)$ (e is the link-edge at k-th link), we identify 3 types of pivots.

$d(v) = 3$: In this case, we cut the edge $(p(v), v)$ from T'_+, and add the cutoff subtree rooted at v to T'_-. This is called a **type 1** pivot.

$d(v) = 2$: In this case, a stage is over. Here, we check whether the subtree of T'_+ rooted at v, which is $T_q + e$ contains any sink node(s) of degree $\geqq 3$.

If so, we apply decompose and add the resulting subtrees to the collection T'_-. Otherwise, we just continue. The former case is called **type 2** pivot and the latter **type 3** pivot. In **type 2** pivots, the number of subtrees in T'_- may increase by more than one. In **type 1** pivots, the number of subtrees in T'_- is the same as that of T_-, and in **type 3** pivots the number of subtrees in T'_- is one less than that of T_-.

The algorithm continues until $T_- = \varnothing$ and terminates with a strongly feasible and hence an optimal tree T_+.

LEMMA 2: *The new forest* $T' = (T'_+, T'_-)$ *is dual-feasible.*

Proof: It suffices to show that with respect to dual variables y', forest T is dual-feasible and the reduced cost of the link-edge e is zero.

Clearly, the reduced costs of the edges in $\gamma(T_-)$ and $\gamma(T_+)$ do not change. The reduced costs of the edges in $\delta(T_-, T_+)$ decrease by ε and those in $\delta(T_+, T_-)$ increase by ε. Since $\varepsilon \geqq 0$, edges in $\delta(T_+, T_-)$ remain dual-feasible. Edges in $\delta(T_-, T_+)$ are also dual-feasible simply because of the way link-edge e is chosen. With respect to y', edge e has zero reduced cost. Therefore, $T+e$ is dual feasible. Clearly, decompose routine does not affect dual-feasibility. As a result, T' is dual-feasible. □

Since the algorithm maintains dual-feasibility and stops with a SFT, it is valid.

Now, we bound the total number of pivots.

THEOREM 1: *The algorithm requires at most* $(n-1)(n-2)/2$ *pivots.*

Proof: Let $\sigma'_i = \sigma_i(T_+)$, $i = 1, 2, 3$ at the beginning of a **stage**. A **stage** ends with a pivot of **type 2** or **3**. A pivot of **type 1** will decrease σ'_2 by 1. Hence, the number of pivots during a stage is bounded by

$$\sigma'_2 + 1 \leqq \sigma_2 + 1 = n - \sigma_1 - \sigma_3 + 1 \leqq n - \sigma_1,$$

since $\sigma_3 \geqq 1$. (Here we assume that root of a tree in T_- contributes 1 to σ_3.) Because σ_1 is at most $n-1$ and at least 2 at the beginning of a **stage**, the maximum number of pivots is

$$\sum_{\sigma_1 = 2}^{n-1} (n - \sigma_1) = \sum_{j=1}^{n-2} j = \frac{(n-1)(n-2)}{2} \quad \square.$$

Now, we give the time complexity of the algorithm.

THEOREM 2: *The algorithm can be implemented so that it has* $O(n^3)$ *time complexity for dense graphs and* $O(n^2 \log n + nm)$ *for sparse graphs.*

Proof: It suffices to show that a **stage** can be implemented at $O(n^2)$ and $O(n \log n + m)$ time for dense and sparse graphs respectively. First we consider the dense case. Clearly, other than the selection of link-edge, everything else in a pivot can be performed in $O(n)$ time per stage. To achieve $O(n^2)$ bound per stage, we need to analyze the cost of selection of link-edges altogether in a **stage**. Since, each such edge has its head in T_+, we store enough information attached to these nodes. Specifically, let

$$s(v) = \min \{ w_{iv} : i \in T_- \}, \qquad \forall v \in T_+ \cap V$$
$$nb(v) = j \qquad \text{if} \quad w_{jv} = s(v).$$

In other words, $s(v)$ is the smallet reduced cost among the edges in $\delta(T_-, v)$ and $nb(v)$ is the tail of such an edge. After a pivot, we have

$$s(k) \leftarrow s(k) - \varepsilon, \qquad k \in T_+ \cap V,$$

and we update $s(k)$'s accordingly.

For a **type 1** pivot, at least one source node, say node v, is transferred from T_+ to T_-. Let T'' be the subtree obtained by deleting edge $(p(v), v)$. In other words, T'' is the subtree rooted at v before the link. We visit the source nodes in T'', for each edge e in $\delta(T'', T_+ \setminus T'')$ compute reduced cost of e, compare with $s(h(e))$ and update $s(h(e))$ and $nb(h(e))$ if necessary. Thus, during a stage, each edge is examined at most once for the computation of $s(v)$ and $nb(v)$. We maintain a list representing $T_+ \cap V$. Sink nodes in T'' are deleted from the list.

For a type 2 or type 3 pivot, a stage is over. After updating dual variables and $s(v)$'s as above, we compute afresh $s(v)$'s for sink nodes added to T_+ during the last pivot.

In order to determine pivot or link edge, we compute

$$\min \{ s(v) : v \in T_+ \cap V \}$$

and the pivot edge is $(nb(v), v)$ for a minimizing v. This completes the dense case.

For the sparse case, we store $s(v)$'s in Fibonacci heaps [7]. Thus, the cost of updating $s(v)$'s and selection of pivot edges will be $O(n \log n)$. Since we may have to examine every edge at least once during a stage, total cost of these operations will be $O(n \log n + m)$ per stage. Since, in any stage, we can perform $O(n)$ pivots, updating dual variables after each pivot is not acceptable. As is shown in Akgül [1], and Goldfarb [8], the total cost of dual updates and tree/forest operations in a **stage** can be bounded in $O(n)$ time.

The basic idea is to maintain an offset between actual reduced costs and those stored in $s(v)$'s and compute ε with respect to $\min\{s(v): v \in T_+ \cap V\}$ and offset. Dual variables can be updated when a stage is over.

3. AVERAGE BEHAVIOR

Using the "equally likely signature model" of Balinski [4] which supposes that at every pivot step, each possible succeeding column signature is equally likely, *i.e.*, every column node that is eligible to increase in degree is equally likely to increase, we can now give a bound on the expected number of pivots required in each stage. Let $g(k, j)$ be the expected number of pivots remaining on a stage of level k at the j-th pivot. Then,

$$g(k,j) = \frac{\sigma'_1}{\sigma'_1+\sigma'_2-j} + \frac{\sigma'_2-j}{\sigma'_1+\sigma'_2-j}(1+g(k,j+1))$$

$$= 1 + \frac{\sigma'_2-j}{\sigma'_1+\sigma'_2-j} g(k,j+1) \quad (1)$$

where $\sigma'_i = \#$ of sink nodes of degree i in T_+ at the beginning of level k stage for $i=1, 2$. Clearly $\sigma'_3 = 0$ for all stages.

Then, it is easy to deduce by induction that

LEMMA 3:

$$g(k,j) = \frac{\sigma'_1+\sigma'_2-j+1}{\sigma'_1+1}.$$

Proof (by induction):

base case. We known that, at the σ'_2-th pivot, the expected number of pivots remaining on this stage is 1.

$$g(k, \sigma'_2) = \frac{\sigma'_1+\sigma'_2-\sigma'_2+1}{\sigma'_1+1} = 1$$

so base case holds.

inductive case. Assume that the argument holds for $(j+1)$-st pivot. We will show that it also holds for j-th pivot as well. By induction hypothesis:

$$g(k,j+1) = \frac{\sigma'_1+\sigma'_2-j}{\sigma'_1+1}$$

By (1), we known that

$$g(k,j)=1+\frac{\sigma_2'-j}{\sigma_1'+\sigma_2'-j}g(k,j+1)=\frac{\sigma_1'+\sigma_2'-j+1}{\sigma_1'+1}$$

So the argument holds for the j-th pivot as well. Hence the proof is complete. □

Since functions $t\mapsto(\alpha+t)/t$ for fixed α, and $\alpha\mapsto(\alpha+t)/t$ for fixed t are monotonic, it follows that $g(k,j)\leqq f(k,j)$ where $f(k,j)$ is defined for Balinski's algorithm [4]. Thus the expected number of pivots is $O(n\log n)$ in our algorithm similar to Balinski's algorithm.

4. VARIATIONS

It is not necessary to start "Balinski tree"; the algorithm works as long as sink nodes of degree 1 are incident with the root. We can work with trees and dual variables obtained by familiar row-minimum, column-minimum method. For each $i\in U$, let $c_{ij(i)}=\min\{c_{ij}: j\in V\}$. The edges $E_0=\{(i,j(i)),\ i\in U\}$ form a part of the initial forest. Let $Q\subset V$ be the set of isolated sink nodes in (U, V, E_0).

Setting $y_i=-c_{ij(i)}$, $\forall i\in U$, and $y_i=0$, $\forall i\in V$ we obtain a dual-feasible solution for which edges in E_0 have zero reduced costs. Let r be a new (artificial) source node. By adding artificial edges (r,j) $\forall j\in V$ we obtain an initial tree. We set $y_r=-K$, and assign cost K to all artificial edges. One can add a new source node, say i_0, and edge (i_0, r) with cost 0 to the initial tree formed. (This last step is not necessary, it is only introduced so that the new graph has a matching provided that the old graph has one.) One may also apply column-minimum operation to nodes in Q to obtain better dual variables, but one does not need to add any edges to E_0.

The value of K is not important, e. g., one can set $K=0$. During the course of the algorithm r (and i_0) will not have any dual-variable changes, and none of the artificial edges will be a link-edge. So, once the initial subtrees T_- and T_+ are constructed, the artificial edges may be deleted. In this version, T_+ will be a forest of SFTs as opposed to being a single SFT.

REFERENCES

1. M. AKGÜL, A Sequential Dual Simplex Algorithm for the Linear Assignment Problem, *Oper. Res. Lett.*, 1988, *7*, pp. 155-158; 1989, *8*, p. 117.

2. M. AKGÜL, A Genuinely Polynomial Primal Simplex Algorithm for the Assignment Problem, SERC Report IEOR 87-07, Bilkent University, 1987 (To appear in *Discrete Appl. Math.*).
3. M. L. BALINSKI, Signature Method for the Assignment Problem, *Oper. Res.*, 1985, *33*, pp. 527-536.
4. M. L. BALINSKI, A Competitive (Dual) Simplex Method for the Assignment Problem, *Math. Programming*, 1986, *34*, pp. 125-141.
5. R. BARR, F. GLOVER and D. KLINGMAN, The Alternating Basis Algorithm for Assignment Problems, *Math. Programming*, 1977, *13*, pp. 1-13.
6. W. H. CUNNINGHAM, A Network Simplex Method, *Math. Programming*, 1976, *11*, pp. 105-116.
7. M. FREDMAN and R. TARJAN, Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms, *J. A.C.M.*, 1987, *34*, pp. 596-615.
8. D. GOLDFARB, Efficient Dual Simplex Algorithms for the Assignment Problem, *Math. Programming*, 1985, *33*, pp. 187-203.
9. K. PAPARRIZOS, A Non-Dual Signature Method for the Assignment Problem and a Generalization of the Dual Simplex Method for the Transportation Problem, *R.A.I.R.O. Rech. Opér.*, 1988, *22*, pp. 269-289.