A SEQUENTIAL DUAL SIMPLEX ALGORITHM FOR THE LINEAR ASSIGNMENT PROBLEM

Mustafa AKGÜL

Dept. of Industrial Engineering, Bilkent University, P.K. 8, 06572, Maltepe, Ankara, Turkey

Received May 1987
Revised April 1988

We present a sequential dual-simplex algorithm for the linear problem which has the same complexity as the algorithms of Balinski [3,4] and Goldfarb [8]: $O(n^2)$ pivots, $O(n^2 \log n + nm)$ time. Our algorithm works with the (dual) strongly feasible trees and can handle rectangular systems quite naturally.

linear assignment problem * dual-simplex * strongly feasible trees * polynomial algorithms

Balinski [3] introduced the signature method for the linear assignment problem which requires $O(n^2)$ pivots and $O(n^3)$ time. Goldfarb [8] introduced a sequential version of the signature method and gave an efficient implementation for sparse graphs. Balinski [4] later gave a purely dual-simplex algorithm having the same complexity as the signature method. The algorithm works with dual strongly feasible trees.

Here we present a sequential dual-simplex algorithm for the assignment problem that has the same complexity as the above algorithms. We solve a sequence of problems defined over the subgraphs of the original graph. Our algorithm works with dual strongly feasible trees and can handle rectangular systems quite naturally.

1. Preliminaries

We will view the assignment problem (AP) as an instance of the 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 $t(e) \in U$ to its head $h(e) \in V$, and has flow x_e and unit cost w_e. For a graph $G = (N, E)$, and disjoint sets $X, Y \subset N$, we let $\gamma(X) = \{e \in E: t(e) \in X, h(e) \in X\}$, $G[X] = (X, \gamma(X))$ (the mode induced subgraph of G), and $\delta(X, Y = \{e \in E: t(e) \in X, h(e) \in Y\}$, $\delta^-(Y) = \delta(Y^c, Y)$, $\delta^+(Y) = \delta(Y, Y^c)$, where $Y^c = N - Y$. For $v \in N$, $d(v) = d_T(v)$ is the degree of a node in the tree T. For a subgraph H of G, $N(H)$, and $E(H)$ represent the node set and the edge set of H. We use $+$ and $-$ to denote set union and set difference, when it is convenient.

We can cast AP as

$$\min\{wx: Ax = b, x \geqslant 0\}, \tag{1}$$

where A is the node edge incidence matrix, and $b_u = -1$, $u \in U$, $b_v = +1$, $v \in V$. The dual LP is:

$$\max\{y^b: y_{h(e)} - y_{t(e)} \leqslant w_e, e \in E\}. \tag{2}$$

The dual simplex method for the transshipment problem starts with a dual feasible tree. If $x_f \geqslant 0$, $\forall f \in T$, then T is optimal. Otherwise the algorithm chooses an $f \in T$ with $x_f < 0$, as the leaving edge (cut-edge), and chooses a co-tree edge $e \in T^\perp = E - T$ as the entering (pivot) edge to satisfy the dual-feasibility via

$$\varepsilon = \bar{w}_e = \min\{\bar{w}_j: j \in \delta^-(Y)\}, \tag{3}$$

where $\bar{w}_j = \bar{w}_j(y) = w_j - y_{h(j)} + y_{t(j)}$ is the reduced cost of the edge j, and Y is the component of $T - f$ containing $t(f)$. Thus, the result of a pivot is the new tree $T' = T + e - f$. A pivot will in-

crease flows on the edges $C^+(T, e)$ by $\theta = -x_f$, decrease flows on $C^-(T, e)$ by θ, and increase the reduced cost of the edges in $\delta^+(Y)$ by ε and decrease that of edges in $\delta^-(Y)$ by ε. (Here $C(T, e)$ denotes the fundamental cycle associated with tree T and co-tree edge e, while $C^+(T, e)$ denotes the edges in the cycle with the same orientation as e.)

Given a tree rooted at node r, $f \in T$ is *reverse* ($f \in \mathrm{R}$) if f is directed toward r. Otherwise it is *forward* ($f \in F$). A primal feasible tree is a *Strongly Feasible Tree* (SFT) if $x_f = 0$, $f \in T \Rightarrow f \in \mathrm{R}$. For a feasible tree T of the assignment problem rooted at a source node r, we have

Lemma 1. *The following are equivalent*:

(i) *T is SFT*,

(ii) $f \in \mathrm{R}$, $f \in T \Leftrightarrow x_f = 0$ *and* $f \in \mathrm{F}$, $f \in T \Leftrightarrow x_f = 1$,

(iii) $d(r) = 1$, $d(u) = 2$, $u \neq r$, $u \in U$. □

SFT's are introduced by Cunningham [6] and Barr, Glover and Klingman [5] and have been used by Akgül [1], and many others, in polynomial primal simplex algorithms.

Since a tree has one less edge than the number of nodes; there is a natural 1-1 correspondence between $N(T) - r$ and $E(T)$, i.e., between the non-root nodes and edges of the tree. When T is reoriented as a branching $\vec{T}$ with root r, the mapping is, say, g: $N(T) - r \to E(T)$, $g(v) = (p(v), v)$, where $p(v)$ is the parent of node $v \in N$, $v \neq r$, in $\vec{T}$. Let $L \cup N - r$ be set of leaf nodes of T, i.e., $L = \{v \in N - r: d(v) = 1\}$, $\tilde{L} = \{v \in L: (v, r) \in E(T)\}$ and let $L_U = L \cap U$. We will also view L as a set of edges via the map g.

A dual feasible tree is a *Dual Strongly Feasible Tree* (DSFT) for AP [4] if

(i) $f \in \mathrm{R} - \tilde{L} \Rightarrow x_j \leqslant 0$,

(ii) $f \in \mathrm{F} \Rightarrow x_f \geqslant 1$.

Note that automatically we have $f \in L \Rightarrow x_f = 1$. At tree which is both a SFT and a DSFT is optimal, (with possibly different roots). Actually, a DSFT has much stronger properties. However, it does not seem possible to extend this definition to the general transshipment problem. Our definition is slightly different from that of Balinski: the roles of forward and reverse edges are interchanged and our tree is rooted at a sink node.

Balinski [4] proved the following

Lemma 2. *Let T be a DSFT (rooted at a sink node r). Consider $u \in U$, (hence $g(u) \in \mathrm{R}$), with $d(u) \geqslant 3$. Then*

(i) $x_{g(u)} \leqslant -1$,

(ii) *the selection of $g(u)$ as the cut-edge maintains DSFT.* □

In other words, if we restrict the selection of cut edges to those $f \in T$ with $f \in \mathrm{R}$ and $d(t(f)) \geqslant 3$, we will maintain a DSFT.

Balinski's dual simplex algorithm [4] works in *stages*. Let $S = \{u \in U: d(u) \geqslant 3\}$. The algorithm for a *stage* (a signature step or a level), for $s \in S$, can be described as:

Algorithm A1 (s).
while $d(s) \geqslant 3$ **do**
cut $f = g(s)$, and let $e \in T^{\perp}$ be the pivot edge via (3)
$T \leftarrow T + e - f$
$s \leftarrow t(e)$
end {**while**}

In a stage, the algorithm starts with $s \in S$ and performs dual-simplex pivots until it reaches a node in L_U. Since Y's are monotonically increasing, the number of pivots in a stage is bounded by $|U - L_U|$. When $S = \emptyset$ or $|L_U| = 1$, T is optimal via (iii) of Lemma 1. Clearly the total number of pivots is bounded by $\sum_{j=1}^{n-2} j = \frac{1}{2}(n-1)(n-2)$ and this bound is sharp [4].

2. The new algorithm

We will solve a sequence of (perturbed) AP's over an increasing sequence of graphs G_0, $G_1, \ldots, G_n$. Each G_k defines an AP: AP_k. Let $V = \{v_1, v_2, \ldots, v_n\}$ be an arbitrary ordering of sink nodes, $r \equiv v_0$ be a dummy sink node, and let $G^{\#} \equiv (U, V + r, E + \{(u, r): u \in U\})$. When G' is a complete bipartite graph, then so is $G^{\#}$. We define G_k as $G_k \equiv G^{\#}[U + \{v_0, v_1, \ldots, v_k\}]$. Actually, AP_k is not strictly an assignment problem; since $b_u = -1$, $u \in U$, $b_{v_j} = 1$, $j = 1, \ldots, k$, $b_r = n - k$. For artificial edges we set, $w_{ur} = K$, for some large K, and set $y_r = K$, $y_u = 0$, $u \in U$ for AP_0. (It turns out that K can be set to 0.) Clearly G_0 is a feasible tree for AP_0. Hence it is optimal, and an optimal solution of AP_n will give the required solution.

Let T_k^* be an optimal tree for AP_k. Then $T_k^* - r$ will be a disjoint union of (primal) SFT's, together with $n-k$ isolated source nodes. Letting $v \equiv v_{k+1}$, in addition to G_k, G_{k+1} contains the node g, and the edges $\delta(U, v)$. Given T_k^* and n, the dual vector y is extended to node v and a new edge is added to T_k^* to obtain T, a DSFT for G_{k+1}:

$$y_v \equiv \min\{w_{uv} + y_u : (u, v) \in E\} = w_{\bar{u}v} + y_{\bar{u}} \qquad (4)$$

and $T = T_k^* + (\bar{u}, v)$. If $d(\bar{u}) = 2$ then T is optimal. Otherwise, $d(\bar{u}) = 3$, and all the reverse edges from r to $\bar{u}$ have flow value -1. Even though a dual simplex algorithm can choose any one of these as a cut-edge, there is a *unique* cut-edge which maintains DSFT, namely $g(\bar{u})$, the reverse edge whose tail is $\bar{u}$. Solving AP_{k+1} starting with the above T will be referred to *stage* $k+1$. Our algorithm for solving AP_{k+1} is the following:

Algorithm A2.
while $d(\bar{u}) = 3$ **do**
cut $f = g(\bar{u})$, and let $e \in T^{\perp}$ be the pivot edge via (3)
$T \leftarrow T + e - f$
$\bar{u} \leftarrow t(e)$
end {**while**}

Let $T_1 = T$, and let f_i, e_i be the cut-edge and pivot edge respectively at iteration i of the current stage, with $T_{i+1} = T_i + e_i - f_i$. Letting Y_i be the component of $T_i - f_i$ containing $t(f_i)$, since $e_i \in \delta^-(Y_i)$, and $e_i \in \gamma(Y_{i+1})$, it follows that $Y_{i+1} \supset Y_i$. Since each $u \in U - L_U$ can be the tail of a cut-edge during a stage, the number of pivots in a stage is bounded by $|U - L_U|$. Thus, since $|U - L_U| = k - 1$ at the beginning of stage k, we have the upper bound on the total number of pivots: $\sum_{k=1}^{n} k - 1 = \frac{1}{2}n(n-1)$. Notice that, the increase in the number of pivots is due to the dummy sink mode.

In some applications U and V may be of different sizes: a rectangular system. Clearly the Hungarian algorithm can handle this case quite easily. However, in primal and dual simplex algorithms, one needs to add enough dummy nodes (and artificial edges) in order to make the new problem feasible. Clearly, our algorithm does not need such a transformation.

3. Implementation and time complexity

In this section we will sketch the basic ideas for efficient implementation of the algorithm, i.e., in $O(n^2 \log n + nm)$ time. For this, it suffices to show that a *stage* can be implemented in $O(n \log n + m)$ time.

We assume that graph is represented by a pair of adjacency lists: for $u \in U$ we have the list $N^+(u)$ (of the edges that start at u) and for $v \in V$, the list $N^-(v)$. Clearly, both lists can be represented as one doubly linked list (see, e.g. [8]). To represent $\vec{T}$ we use 4 pointers: *parent, first* (child), *left* (sibling), *right* (sibling). Thus the children of a node are maintained as a doubly linked circular list [9].

As it is well known, the most costly part of the dual simplex algorithm is the selection of incoming edges. At every pivot, given the set Y_i, we need to determine

$$\min\{\bar{w}_j : j \in \delta^-(Y_i)\}. \qquad (3')$$

As pointed out by Balinski [3,4], Goldfarb [8] and Akgül [1], to achieve this bound one needs to capitalize on the nested structure of cutsets Y_i's, (similar to Hungarian and Dijkstra algorithms). Let us define

$$Z_i = \begin{cases} Y_1, & i = 1, \\ Y_i - Y_{i-1}, & i > 1, \end{cases} \qquad (5)$$

$$T_i^+ = T_i - Y_i = T - \bigcup_{j=1}^{i} Z_j. \qquad (6)$$

For $u \in U \cap Y_i^c$, let

$$s(u) = \min\{\bar{w}_{uv} : v \in Y_i\}, \qquad (7)$$

$$\mathrm{nb}(u) = v \quad \text{if } \bar{w}_{uv} = s(u). \qquad (8)$$

$s(u)$ measures the smallest reduced cost of edges in $\delta(u, Y_i)$, while $\mathrm{nb}(u)$ keeps the index of one such edge (hence a candidate for a pivot edge). Given $s(u)$, $u \in U \cap Y_i^c$, one can compute (3') by finding

$$\min\{s(u) : u \in U \cap Y_i^c\}. \qquad (9)$$

Instead of computing and storing $\bar{w}_e(y^{i-1})$, where y^i is the dual vector associated with tree T_i, we will compute and store $\bar{w}_e(y)$, the reduced cost at the beginning of a stage. Letting $\sigma = \sum_{j=1}^{i-1} \varepsilon_j$, at the i-th pivot, we need to compute $\bar{w}_e(y)$, $\forall e \in \delta(Y_i^c, Z_i)$. Since $\bar{w}_e(y^{i-1}) = \bar{w}_e(y) - \sigma$, we com-

pute and store $\bar{w}_e(y)$, in $s(u)$'s and set $\varepsilon_i = \min\{s(u): u \in U \cap Y_i^c\} - \sigma$. Working with T_i^+ now pays off, Z_i is the subtree of T_{i-1}^+ rooted at $t(f_i)$. In constant time we obtain T_i^+, and in time linear in $|Z_i|$ we traverse Z_i. We then examine edges in $\delta(Y_i^c, Z_i)$ (actually in $\delta^-(v)$, $v \in Z_i$), and update $s(u)$ and nb(u)'s. Note that any $s(u)$ for $u \in Z_i$ is discarded. Since each edge $e \in E$ will be examined at most once, only when $h(e) \in Z_i$ for some i, the total work for the computation of $s(u)$ and nb(u) is O(m). Evaluation of (9) will require O(n) time for each pivot. Hence, if one explicitly carries $s(u)$'s in an array; the total work for each stage will be O$(n^2 + m)$.

If one is willing to use binary heaps to carry $s(u)$'s, then the total work per stage will be O$(n \log n + m \log n)$. However, using Fibonacci heaps [7], one can perform all these operations in O$(n \log n + m)$ time per stage.

There remains the cost of reconstructing the new tree T_{k+1}^* at the end of a stage. If one is not careful, the construction of the new tree may require O(n^2) time because of the rerooting of the subtrees on Z_i's. The key to reduction of this bound to O(n) is to perform a graph search, e.g., breadth first search on the surface graph with nodes $Z_1, Z_2, \ldots, Z_i, T_i^+$, and edges $e_1, e_2, \ldots, e_i$, and then reconstruct T_{k+1}^* using that information. Once the tree T_{k+1}^* is constructed, one can easily compute the new dual variables from scratch in O(n) time.

4. Conclusion and remarks

We have presented yet another algorithm with O(n^2) pivots and O$(n^2 \log n + nm)$ time complexity: the same as with primal-dual [7], dual simplex [3,4,8], and primal-simplex [1]. The signature methods of Balinski and Goldfarb, strictly speaking, are not dual-simplex algorithms, for they may not recognize a feasible solution. Our algorithm provides a constant factor improvement over Balinski's [4], similar to Goldfarb's over the signature method. Assuming dual non-degeneracy, the behavior of our algorithm is completely determined by the ordering of the nodes in V, whereas in Balinski's [4] algorithm its behavior is determined by the order of the nodes input to Algorithm A1. It is important to note that the concept of a stage is essential, not only for proving the pivot bound, but also for obtaining the time bound. There are other pivot rules within the family DSFT which guarantee polynomial time. They are discussed in [2].

References

[1] M. Akgül, "A genuinenly polynomial primal simplex algorithm for the assignment problems", SERC Report IEOR 87-07, Bilkent University, 1987.

[2] M. Akgül, "Variations on a theme of Balinski: Signature methods for the linear assignment problem", SERC Report IEOR-8802, Bilkent University, 1988.

[3] M. Balinski, "Signature method for the assignment problem", *Operations Research* 33, 527–536 (1985). Presented at Mathematical Programming Symposium, Bonn, 1982.

[4] M. Balinski, "A competitive (dual) simplex method for the assignment problem", *Mathematical Programming* 34, 125–141 (1986).

[5] R. Barr, F. Glover and D. Klingman, "The alternating basis algorithm for assignment problem", *Mathematical Programming* 13, 1–3 (1977).

[6] W. Cunningham, "A network simplex method", *Mathematical Programming* 11, 105–116 (1976).

[7] M. Fredman and R. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms", *Journal of ACM* 34, 596–615 (1987). Also in *Proc. 25-th FOCS* (1984) 339–346.

[8] D. Goldfarb, "Efficient dual simplex algorithms for the assignment problem", *Mathematical Programming* 37, 187–203 (1985).

[9] R. Tarjan, *Data Structures and Network Algorithms*, SIAM, Philadelphia, PA, 1983.