TREE MODIFICATION IN THE TRANSPORTATION PROBLEM:
A NEW METHOD

J. INTRATOR
DEPARTMENT OF ECONOMICS
BAR-ILAN UNIVERSITY, RAMAT-GAN, ISRAEL

G. EHRLICH
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
BAR-ILAN UNIVERSITY

SUMMARY

Most of the time consumed by the MODI method [2] for solving an m\times n transportation problem is spent on updating the m+n shadow-prices. It is shown that it is sufficient to update the shadow-prices for the m rows only. As a consequence we get an algorithm which is better than the usual MODI method for square problems and much better for long transportation problems. It is also an improvement to the methods especially developed for long problems [1, 3].

1. THE MODI METHOD FOR SOLVING A TRANSPORTATION PROBLEM

Let c_{ij} > 0, a_i > 0, b_j > 0, i = 1, 2, \ldots, m; j = m+1, m+2, \ldots, m+n be given. The classical transportation problem is to find a system x_{ij} \geq 0 that minimizes the object function.

\[ \sum_{i,j} c_{ij} x_{ij} \]

subject to

\[ \sum_j x_{ij} = a_i \]

and to

\[ \sum_i x_{ij} = b_j \]

where with no loss of generality it is supposed that \[ \sum a_i = \sum b_j. \]

It is also supposed that the problem is non-degenerate. i

The transportation problem may be represented as an undirected bipartite graph consisting of a set of origin nodes i = 1, 2, \ldots, m with supplies a_i and a set of destination nodes j = m+1, m+2, \ldots, m+n with demands b_j. Arcs (i,j) from origin nodes to destination nodes accommodate the transmission of flow and incur a cost if flow exists. If the bipartite graph defined in this way is a tree then the system of flows x_{ij} is called a basic solution. In Hadley [2] it can be seen how this "basic solution" is associated with the basic solution of the symplex method for linear programming. Suppose a basic solution is given. Out of the m+n nodes of the tree, Manuscript received May 1983, revised January 1984.
a node \( R \) is selected to serve as a root. A shadow price \( y_k \) is defined for every node \( k \) by:

\[
\begin{align*}
(i) & \quad y_R = 0 \\
(ii) & \quad \text{if } (i,j) \text{ is an edge in the tree then } y_i + y_j - c_{ij} = 0.
\end{align*}
\]

Since every node \( k \) is connected to \( R \) by a unique path, \( y_k \) is uniquely defined. The MODI method starts and works through basic solutions only. An edge \((i,j)\) not in the tree is a candidate for entering the tree if \( y_i + y_j - c_{ij} > 0 \). If none exist the current solution is optimal.

Each MODI iteration has the following steps:

a) Find an entering edge \( e_0 \).
b) Find the circle closed by \( e_0 \).
c) Find the outgoing edge.
d) Update the tree.
e) Update the shadow prices.

Steps (d) and especially (e) are time consuming; our aim is to save most of that time.

**Lemma:** Let the rows \( j \) and \( g \) be both neighbours of a column \( k \). Then

\[
y_j = y_g + c_{jk} - c_{gk}
\]

**Proof:**

\[
y_j = c_{jk} - y_k = c_{jk} - (c_{gk} - y_g) = y_g + c_{jk} - c_{gk}.
\]

It is sufficient to compute the shadow prices for rows only; and once we need a shadow price for a column we will find it by

\[
y_j = c_{p(j),j} - y_p(j), \quad \text{where } p(j), \text{ the father index of } j, \text{ is the node next to } j \text{ on the path from } j \text{ to the root } R.
\]

2. A BUSH

In our method we describe the tree by the father index \( P \) only. Besides the tree we build another tree called a bush. The nodes of the bush are just the \( m \) rows \( 1,2,\ldots,m \). We assume the root \( R \) of the tree to be a row. \( R \) also serves as the root of the bush. For each two rows \( i,j \) the edge \((i,j)\) belongs to the bush if and only if one of them is a grandfather of the other, namely if there is a column \( k \) s.t. \((i,k)\) and \((j,k)\) belong to the tree. The bush is uniquely defined by the tree, namely the root \( R \) and the father index \( P \); however to ease searching sub-bushes, we define the bush by the GrandSon and BRoger other indices.

\( GS(i) \) is one of the grandsons of \( i \); if none exists then \( GS(i) = 0 \), and the array \( BR(1), BR(2), \ldots, BR(m) \) is defined such that the list

\[
j + GS(i) \quad \text{Repeat } BR(j) \text{ while } j \neq 0
\]

is the list of all grandsons of \( i \). Let \( P \) and \( R \) describe a tree;
then a bush can be defined by Algorithm 1.

**Algorithm 1:**

Let BS and BR be m-elements arrays.

For i = 1 step 1 to m  GS(i) = 0, BR(i) = 0

For i = 1 step 1 to m

if i ≠ R then Begin j = P(P(i))

BR(i) = GS(j)

GS(j) = i

end

end Algorithm 1.

**Example:**

\[ m=5 \]
\[ n=6 \]

\[
\begin{align*}
R &= 2 \\
P &= (11,0,7,11,8,2,2,3,2,2) \\
GS &= (0,4,5,0,0) \\
BR &= (0,0,1,3,0)
\end{align*}
\]

We found that it is faster to compute the shadow prices for the bush or for some subtree of the bush by using the Width First Search [1] than to compute them by using the Depth First Search.

3. **UPDATING THE BUSH**

Suppose the entering edge to the tree, step a in the MODI method, is \((Q_1, Q_2)\). Suppose also that the leaving edge \((N_1, N_2)\) is lying on the path \(Q_1 - R\) and that \(N_2 = p(N_1)\). Let \(i\) be a row in the path \(Q_1 - N_1\). We have to check every grandson \(j\) of \(i\) and if \(P(i)\) belongs to the path \(Q_1 - N_1\) we have to transfer \(j\) from the list of grandsons of \(i\) to the list of the grandsons of the new \(P(P(j))\).
It is easily seen that we can update all the shadow prices that must be changed, by adding to them just the same constant.

It is faster to update the bush and the shadow prices than to rebuild the whole bush even for $m = 5$. We use the minimum column method [4] for finding the entering edge. Thus a shadow price for a column is computed just for a column that is checked for an entering edge.

In the MODI method, once we have to update the shadow price of a node we have to do the same for all its descendants. The expected number of sons of a row is $n/m$. In case of a long transportation problem $n >> m$, updating shadow prices requires much computation time; in our method this time is saved. The tree is defined by $P$ only and thus it can be updated very easily. Updating the bush and the shadow prices require no more than $O(m)$ operations.

The MODI method as it appears in [4] was coded; the same code was modified to use the bush. Both algorithms were tested on the Bar-Ilan IBM 370/168 using the PLI optimizer with a time optimization option. For several sizes of a problem $mxn$, four problems were generated. The values of $c, a, b$ were drawn out of a uniform distribution. The same four problems were solved by both methods and the average solving time is given in the table below. We measure only the net time required to solve the problems. The time to generate them was not included in the measure. The experimental results are presented in Table 1.

Table 1. Time comparison of the Jacobson implementation [4] and the Bush method. Time measured in milliseconds.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Jacobson</th>
<th>Bush</th>
<th>Bush/Jacobson</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 x 500</td>
<td>1436</td>
<td>360</td>
<td>.25</td>
</tr>
<tr>
<td>5 x 1000</td>
<td>6808</td>
<td>1218</td>
<td>.18</td>
</tr>
<tr>
<td>5 x 2000</td>
<td>31197</td>
<td>3581</td>
<td>.11</td>
</tr>
<tr>
<td>5 x 5000</td>
<td>210639</td>
<td>19198</td>
<td>.09</td>
</tr>
<tr>
<td>10 x 1000</td>
<td>9371</td>
<td>1698</td>
<td>.18</td>
</tr>
<tr>
<td>10 x 2000</td>
<td>40829</td>
<td>4500</td>
<td>.11</td>
</tr>
<tr>
<td>50 x 1000</td>
<td>12357</td>
<td>3796</td>
<td>.31</td>
</tr>
<tr>
<td>50 x 500</td>
<td>166</td>
<td>142</td>
<td>.86</td>
</tr>
<tr>
<td>100 x 100</td>
<td>818</td>
<td>622</td>
<td>.76</td>
</tr>
<tr>
<td>300 x 300</td>
<td>9231</td>
<td>7734</td>
<td>.83</td>
</tr>
</tbody>
</table>
Conclusion:

Adding the bush to the Jacobson code [4] proved itself a good time-saving device. It seems obvious that adding it to the new fast methods for long transportation problems will cause a meaningful improvement.

REFERENCES


