Following Tabourier [28], Dorhout [10, 11] shows computational advantages in leaving a choice for the final column.

The following sections are devoted to the steps of shortest augmenting path LAP algorithms:

Step 1: *Initialization*

Step 2: *Termination*, if all rows are assigned.

Step 3: *Augmentation*
Construct the auxiliary network and determine from an unassigned row i to an unassigned column $j$ an alternating path of minimal total reduced cost, and use it to augment the solution.

Step 4: *Adjust the dual solution* to restore complementary slackness.
Go to step 2.

We discuss initialization strategies in the next section. How to modify shortest path algorithms for assignment augmentation is the subject of Section 5. Section 6 describes a simple way to adjust row and column prices after shortest path augmentation.

## 4. The Initialization Phase

A standard method of initialization in LAP algorithms is column and row reduction. For each column $j$ a row index $i^*$ is determined with minimum $c[i,j]$ $(i=1 \ldots n)$, $v[j]$ is set to $c[i^*,j]$ and, if $i^*$ is unassigned, $j$ is assigned to $i^*$. Next, one finds for every unassigned row $i$ the column $j^*$ with $c[i,j]-v[j]$ $(j=1 \ldots n)$ minimum and assigns $j^*$, if unassigned, to row $i$.

In our assignment algorithm the initialization is primarily aimed at reaching a high initial reduction of the costs matrix. It consists of three procedures, discussed below:

- reduction of columns,
- reduction transfer from unassigned to assigned rows,
- augmenting reduction of unassigned rows.

The first procedure is standard *column reduction*. An implementation detail is considering the columns in reverse order. So, the low indexed columns are most likely to remain unassigned. As a consequence, if minimum reduced costs in a row occur at an unassigned column, this column is automatically selected as the first in which the minimum occurs.

The second procedure is *reduction transfer*. Its objective is to further reduce assigned rows, but it has no direct net effect on the reduction sum. Afterwards a higher reduction sum may be obtained when unassigned rows are reduced.

Consider a row $i$ assigned to a column, say $k$. By sufficiently decreasing the price of column $k$, row $i$ can be reduced by the current second minimum of the reduced costs. This additional reduction of assigned rows leads to an increase of some reduced costs in unassigned rows, so that these may later be reduced further. The effect of the procedure is twofold, as assigned columns are made more expensive relative to the unassigned ones. Bertsekas [3] calls this "outpricing" of assigned columns. In general the shortest paths in the augmentation phase will now earlier reach some unassigned column.

The straightforward procedure for reduction transfer is given in Fig. 1. For clarity we have not taken into account that at the start of the procedure all $u[i]$ have value zero. Furthermore, one can keep track during the column reduction for which rows reduction transfer will certainly be useless.

```
procedure REDUCTION TRANSFER;
begin
   for each assigned row i do
   begin
      j1:=x[i]; μ:=min {c[i,j]−v[j] | j=1   n,j<>j1}
      v[j1]:=v[j1]−(μ−u[i]); u[i]:=μ
   end
end
```

Fig. 1. Procedure for reduction transfer

Clearly, reduction transfer may also be applied in the course of the augmentation phase. Computational experiments showed that this does not improve the performance of the algorithm LAPJV (Section 7).

*Augmenting row reduction* is the third initialization procedure. An attempt is made to find augmenting paths starting in unassigned rows, to which at the same time reduction is transferred. In the process assigned columns remain so, but rows may become assigned, unassigned or reassigned.

```
procedure AUGMENTING ROW REDUCTION;
begin
   LIST:={all unassigned rows};
   for all i∈LIST do
   repeat
      u1:=min {c[i,j]−v[j] | j=1 ... n};
      select j1 with c[i,j1]−v[j1]=u1;
      u2:=min {c[i,j]−v[j] | j=1 ... n, j<>j1};
      select j2 with c[i,j2]−v[j2]=u2 and j2<>j1;
      u[i]:=u2;
      if u1<u2 then v[j1]:=v[j1]−(u2−u1)
               else if j1 is assigned then j1:=j2;
      k:=y[j1]; if k>0 then x[k]:=0; x[i]:=j1; y[j1]:=i; i:=k
   until u1=u2 (* no reduction transfer *) or k=0 (* augmentation *
end.
```

Fig. 2. Procedure for augmenting row reduction

The procedure is given in Fig. 2. In each step of the for-loop an alternating path is started up from an unassigned row, say $i$. Consider the column $j$ where the minimum reduced costs in row $i$ occur. If $j$ is unassigned, the alternating path leads to augmentation of the solution. If not, the path is extended by reassigning column $j$ to