Contents

Chapter 3

Primal-Dual and Dual Algorithms for the Assignment and Transportation Problems

The assignment and transportation problems are single commodity minimum cost flow problems on pure bipartite networks. **Primal-dual algorithms** are a class of methods for LPs with the following characteristic features:

1. They maintain a dual feasible solution, and a primal vector (this vector is primal infeasible until termination) that together satisfy all the complementary slackness optimality conditions for the original problem throughout the algorithm. This primal vector is usually feasible to a relaxation of the primal problem (typically this is obtained by changing the equality constraints in the problem to "$\leqq$" inequalities).

2. In each step, the algorithm either performs (a) below, or (b) if this is not possible.

   (a) Keeps the dual solution fixed, and tries to alter the primal vector to bring it closer to primal feasibility while continuing

to satisfy the complementary slackness conditions together with the present dual solution.

(b) Keeps the primal vector fixed, and changes the dual feasible solution. The aim of this is to get a new dual feasible solution satisfying two conditions. The first is that the new dual feasible solution satisfies the complementary slackness conditions together with the present primal vector. The second is that it makes it possible to get a new primal vector closer to primal feasibility when the algorithm continues.

3. As the algorithm progresses, the primal vector moves closer and closer to primal feasibility. In other words, there is a measure of primal infeasibility which improves monotonically during the algorithm.

There are two possible conclusions at termination. One occurs if the primal vector being maintained becomes primal feasible at some stage; then it is an optimum solution. The second occurs if a primal infeasibility criterion is satisfied at some stage.

For the assignment and transportation problems it is easy to obtain an initial dual feasible solution. And the task in (a) above is a maximum value flow problem on a subnetwork known as the **admissible** *or* **equality subnetwork** wrt the present dual feasible solution. These facts make the primal-dual methods particularly attractive to solve them. The blossom algorithms discussed in Chapter 10 for matching and edge covering problems are primal-dual algorithms that are generalizations of the Hungarian method of the next section to those problems. The primal-dual approach can be used to solve a general LP, however, for these general problems it seems to offer no particular advantage over the primal simplex algorithm.

3.1 The Hungarian Method for the Assignment Problem

The data in an assignment problem of order n is the cost matrix $c = (c_{ij})$ of order $n \times n$. Given c the problem is to find $x = (x_{ij})$ of order

$n \times n$ to

$$\begin{aligned} \text{Minimize } z(x) &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{Subject to } \sum_{j=1}^{n} x_{ij} &= 1 \quad \text{for } i = 1 \text{ to } n \qquad (3.1) \\ \sum_{i=1}^{n} x_{ij} &= 1 \quad \text{for } j = 1 \text{ to } n \\ x_{ij} &\geqq 0 \quad \text{for all } i, j \end{aligned}$$

$$\text{and } x_{ij} = 0 \text{ or } 1 \quad \text{for all } i, j \qquad (3.2)$$

Every feasible solution of (3.1) and (3.2) is an assignment of order n and vice versa. See (1.17) for an assignment of order 4. Every BFS of (3.1) satisfies (3.2). So, if (3.1) is solved by the simplex method ignoring (3.2), the optimum solution obtained will satisfy (3.2) automatically. The Hungarian method does not use basic vectors for (3.1), but it maintains (3.2) throughout.

In an assignment x, if a particular $x_{ij} = 1$, the cell (i, j) is said to have an **allocation** (in this case row i is said to be **allocated to**, or **matched with**, column j in x). A **partial assignment** of order n is a 0-1 square matrix of order n which contains at most one nonzero entry of 1 in each row and column. Here is a partial assignment of order 3.

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Clearly, a partial assignment is a feasible solution for a relaxed version of (3.1) and (3.2) in which the equality constraints in (3.1) are replaced by the corresponding "$\leqq$" inequalities. The Hungarian method moves among partial assignments in which the number of allocations keeps on increasing as the algorithm progresses.

Represent each row and each column of an $n \times n$ array by a node. Join each row node to each column node by an edge, leading to a complete bipartite network (see Figure 3.1). Make the cost of the edge joining row node i with column node j, c_{ij}. The set of allocated cells in any partial assignment (assignment) corresponds to a matching (perfect matching) in this bipartite network, and vice versa. For example, the dashed subnetwork in Figure 3.1 (a) is the perfect matching corresponding to the assignment in (1.17). Hence, the assignment problem (3.1) and (3.2) is equivalent to that of finding a minimum cost perfect matching in this bipartite network. That's why it is also known as the **bipartite minimum cost perfect matching problem**.

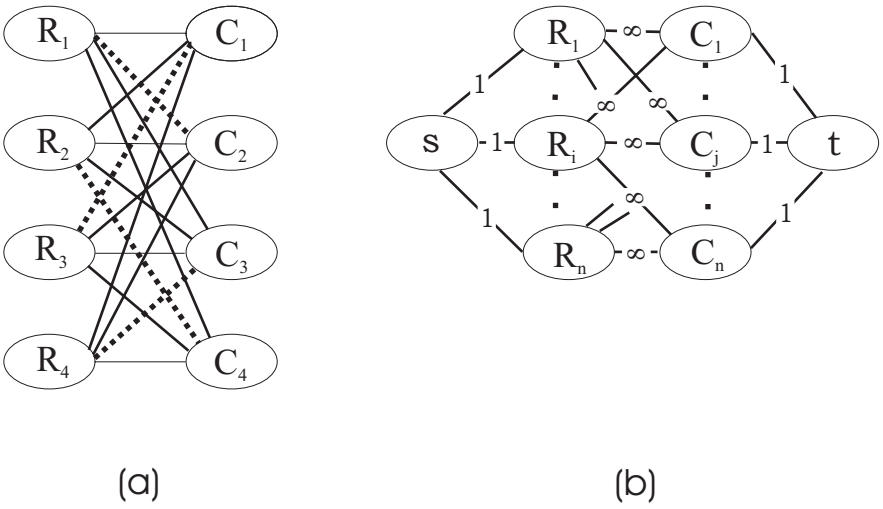

Figure 3.1:

Direct all the lines in this bipartite network from the row node to the column node. Treat all the row nodes as sources each with a supply of one unit, and the column nodes as sinks with a demand of one unit. Then (3.1) is the problem of finding a minimum cost feasible flow in this network. An allocation in cell (i, j) corresponds to a flow of one unit on the arc connecting row node i with column node j and vice versa. We can introduce a supersource s and supersink t and transform this into a minimum cost flow problem on the network in Figure 3.1 (b), in

which all the arcs incident at s, t are required to be saturated.

In large scale applications, for each i, a subset of $\{1, \ldots, n\}$ is usually specified, and x_{ij} can be equal to 1 only when j is in that subset; otherwise it has to be 0. This can be handled by defining c_{ij} to be $+\infty$ whenever x_{ij} is required to be 0. In the corresponding networks in Figures 3.1 (a) and (b), the arc (i, j) (i.e., $(\mathrm{R}_i, \mathrm{C}_j)$) is included only if x_{ij} can be equal to 1 in the problem. Thus the network is no longer the complete bipartite network. Let m denote the number of arcs in the network (i.e., the number of variables x_{ij} which can assume the value 1). The assignment problem is said to be **sparse** if m is small compared to $n^{\mathbf{2}}$, and **dense** if m is close to $n^{\mathbf{2}}$.

In this section we present an implementation of the primal-dual approach for the assignment problem known as the **Hungarian method**. It is described using arrays for ease of understanding, but computer implementations are usually based on the corresponding network. The arc joining row node i to column node j is omitted if x_{ij} is required to be 0 in the problem. The dual of (3.1) is

$$\begin{aligned} \text{Maximize } & \sum_{i=1}^{n} u_i \quad + \quad \sum_{j=1}^{n} v_j \\ \text{Subject to } & u_i + v_j \quad \leqq \quad c_{ij}, \ i, j = 1 \text{ to } n \end{aligned} \tag{3.3}$$

Denote the objective value of assignment x with c as the cost matrix by $z_c(x)$. Let c' be the matrix obtained by subtracting a real number α from every element in a row or a column of c. Since each assignment contains a single nonzero entry of 1 in each row and column, we have $z_c(x) = \alpha + z_{c'}(x)$. So, the set of optimum assignments that minimize $z_c(x)$ is the same as the set of optimum assignments that minimize $z_{c'}(x)$. Hence, for solving the assignment problem, we can replace c by c'. We use this idea repeatedly. Let $u = (u_1, \ldots, u_n), v = (v_1, \ldots, v_n)$ be such that $c^0_{ij} = c_{ij} - u_i - v_j \geqq 0, i, j = 1$ to n. This is the condition for u, v to be dual feasible, and in this case c^0_{ij} are the dual slacks, $c^0 = (c^0_{ij})$ is known as the **reduced cost matrix**, and $r_0 = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$ as the **total reduction** wrt u, v. The matrix c^0 is obtained by subtracting u_i from each entry in row i of c for $i = 1$ to n, and then subtracting v_j from each entry in column j of the resulting matrix, for $j = 1$ to n. By

the above argument, for each assignment x we have $z_c(x) = r_0 + z_{c^0}(x)$. Since $c^0 \geqq 0, z_{c^0}(x) \geqq 0$, and so r_0 is a lower bound for the minimum objective value in (3.1) whenever u, v is dual feasible. If we can find an assignment x which has allocations only in those cells in which the entries in c^0 are zero, then $z_{c^0}(x) = 0$, and hence x is an optimum assignment. To find such an assignment, we define the cell (i, j) or the corresponding arc (i, j) (we will use this notation to denote the arc joining row node i to column node j) in the network in Figure 3.2 to be **admissible** (or an **equality cell** or **equality arc** respectively) if $c^0_{ij} = c_{ij} - u_i - v_j = 0$. When all inadmissible arcs are removed from the network in Figure 3.1 (b), we get the **admissible subnetwork** or the **equality subnetwork** wrt the dual feasible solution u, v. A maximum value flow from s to t in the equality subnetwork corresponds to a partial assignment having the maximum number of allocations among admissible cells. If this is an assignment, it is clearly optimal to (3.1) and (3.2); otherwise we get a partial assignment x satisfying

$$x_{ij}(c_{ij} - u_i - v_j) = 0, \text{ for all } i, j \tag{3.4}$$

The complementary slackness optimality conditions for (3.1) and its dual (3.3) are (3.4). The Hungarian method maintains $x, (u, v)$ always satisfying (3.2), dual feasibility constraints in (3.3), and (3.4). When x satisfies (3.1), it is an optimum assignment and the method terminates. If the maximum value flow in the equality subnetwork does not saturate all the arcs incident at t, there exists no assignment which has allocations among admissible cells only. In this case the Hungarian method goes to a dual solution change routine. After this change the new reduced cost matrix will contain some new cells with zero entries in columns which have no allocations at present, and the procedure is repeated. During the method, each row and column of the array (i.e., each node in the network implementation) may be in three possible states: **unlabeled, labeled and unscanned, labeled and scanned**. The **list** is always the set of current labeled and unscanned rows and columns.

THE HUNGARIAN METHOD

Step 0 The initial dual feasible solution If some dual feasible

solution is available, use it as the initial one; otherwise define it to be $(u^1 = (u_i^1), v^1 = (v_j^1))$ where $u_i^1 = \min \{c_{ij} : j = 1 \text{ to } n\}$, $v_j^1 = \min \{c_{ij} - u_i^1 : i = 1 \text{ to } n\}$, for $i, j = 1$ to n. List $= \emptyset$. Go to Step 1 with any partial assignment containing allocations only among admissible cells in the reduced cost matrix wrt the initial dual solution (this could be 0, containing no allocations).

Step 1 Tree growth routine

Substep 1 Label each row without an allocation with $(s, +)$, and include it in the list.

Substep 2 If list $= \emptyset$, tree growth has terminated and there is a nonbreakthrough. The present set of allocations contains the maximum number possible among admissible cells; go to Step 3. Otherwise, select a row or column from the list for scanning and delete it from the list.

Forward labeling Scanning row i consists of labeling each unlabeled column j for which (i, j) is an admissible cell, with the label (row i, $+$).

Reverse labeling To scan column j, check whether it has an allocation. If the row in which that allocation occurs is unlabeled so far, label it with (column j, $-$).

If any column without an allocation has been labeled, there is a breakthrough; go to Step 2. Otherwise, include all newly labeled rows and columns in the list, and repeat this Substep 2.

Step 2 Allocation change routine Suppose column j, which does not have an allocation, has been labeled. Trace its predecessor path using the labels. Delete present allocations in cells corresponding to reverse arcs, and add allocations in cells corresponding to forward arcs of this path. If all the columns have allocations now, these allocations define an optimum assignment; terminate. Otherwise, chop down the present trees (i.e., erase the labels on all the rows and columns) and go back to Step 1.

Step 3 Dual solution change routine Compute δ, minimum value of reduced cost coefficient among cells in labeled rows and unlabeled columns. δ will be > 0. If $\delta = +\infty$; this can only happen if some x_{ij} are constrained to be 0 in the problem; there is no feasible assignment, terminate. If δ is finite, add it to the value of u_i in all labeled rows and subtract it from the value of v_j in all labeled columns. Compute the new reduced cost coefficient in each cell. Retain the present labels on all the labeled rows and columns, but include all the labeled rows in the list, and resume tree growth by going to Substep 2 in Step 1.

Discussion

When solving small problems by hand, a good initial partial assignment in Step 0 can be obtained by making an allocation in an admissible cell in the initial reduced cost matrix that is not yet struck off, in a row or column containing only one such cell if possible, or in any admissible cell not yet struck off otherwise; striking off all other admissible cells in the row and column of the allocated cell; and repeating this process with the remaining admissible cells.

Also, to find δ in Step 3, draw a straight line in the present reduced cost matrix through each unlabeled row and each labeled column, then these straight lines cover all the admissible cells (if there is an admissible cell without a line through it, its column would have been labeled when its row was scanned, a contradiction, see Array 3.1). Hence every reduced cost coefficient not covered by a straight line is > 0, and δ is the minimum of these entries. So, δ will always be > 0. The number of allocations at this stage can be verified to be equal to the number of straight lines drawn. To get the new reduced cost coefficients, subtract δ from the entry in the present reduced cost matrix in each cell in a labeled row and unlabeled column (i.e., those cells without a straight line through them), and add δ to the entry in each cell in an unlabeled row and labeled column (i.e., those cells at the intersection of two straight lines). From the definition of δ this implies that all the new reduced cost coefficients are $\geqq 0$, i.e., the new dual solution is dual feasible. Let α be (the number of labeled rows $-$ the number of labeled columns) at present. Add $\delta\alpha$ to the total reduction; this updates it.

Array 3.1 Summary of Position When Hungarian Method Reaches Dual Solution Change Routine. $(\tilde{u}, \tilde{v}, \tilde{\tilde{c}} = (\tilde{\tilde{c}}_{ij} = c_{ij} - \tilde{u}_i - \tilde{v}_j))$, are the dual solution and reduced cost matrix before change. $(\hat{u}, \hat{v}, \hat{\tilde{c}} = (\hat{\tilde{c}}_{ij} = c_{ij} - \hat{u}_i - \hat{v}_j))$ are the corresponding things after change.

	Block of labeled cols.	Block of unlabeled cols.	Allocations	St. lines	Dual change
Block of labeled rows	Each col. here has an allocation among labeled rows (there is breakthrough otherwise). $\hat{\tilde{c}}_{ij} = \tilde{\tilde{c}}_{ij}$ here, so, admissibility pattern remains unchanged.	No admissible cells here (one col. would be labeled otherwise). $\tilde{\tilde{c}}_{ij} > 0$, here. $\delta = \text{Min.}\{\tilde{\tilde{c}}_{ij}: (i,j)$ here$\} > 0$. $\hat{\tilde{c}}_{ij} = \tilde{\tilde{c}}_{ij} - \delta$ here. New admissible cells created here, this allows tree growth.	Some rows have no allocation.		$\hat{u}_i = \tilde{u}_i + \delta$ for i here.
Block of unlabeled rows	No allocation here (otherwise a row here could be labeled). $\hat{\tilde{c}}_{ij} = \tilde{\tilde{c}}_{ij} + \delta$ here. All cells here become inadmissible next.	Each row here contains an allocation in these cols. $\hat{\tilde{c}}_{ij} = \tilde{\tilde{c}}_{ij}$ here, so admissibility pattern remains unchanged here.	Each row has allocation.	Draw through each row.	$\hat{u}_i = \tilde{u}_i$ for i here.
Allocations	Each col. has allocation.	Some cols. have no allocation.			
St. lines	Draw through each col.				
Dual change	$\hat{v}_j = \tilde{v}_j - \delta$ for j here	$\hat{v}_j = \tilde{v}_j$ for j here.			

Cells that have allocations at present remain admissible in the new reduced cost matrix, and hence (3.4) continues to hold. New admissible cells are created among labeled rows and unlabeled columns, so, when the list is made equal to the set of all labeled rows, and tree growth resumed, at least one new column will be labeled. See Array 3.1

Suppose δ came out to be $+\infty$ in Step 3 at some stage. Let $\mathbf{J}$ be the set of all cells (i, j) with row i labeled, and column j unlabeled at this stage. So, the present reduced cost coefficient $\bar{c}_{ij} = +\infty$ for all cells $(i, j) \in \mathbf{J}$ (i.e., x_{ij} is required to be 0 for every $(i, j) \in \mathbf{J}$). Even if all cells not in $\mathbf{J}$ are made admissible, no more labeling can be carried out, and the current nonbreakthrough continues to hold. This implies that the present partial assignment contains the maximum number of allocations possible under the constraint that x_{ij} must be 0 for all $(i, j) \in \mathbf{J}$, hence there is no feasible assignment in the problem.

Consider the Hungarian method applied to solve an assignment problem of order n. Whenever Step 2 is carried out, the number of allocations increases by 1. Thus Step 2 is carried out at most n times in the algorithm. From Array 3.1 we see that at least one new column gets labeled when tree growth is resumed after a dual solution change step. Thus Step 3 can occur at most n times between two consecutive occurrences of Step 2. The effort needed to carry out Step 3 (updating the reduced cost coefficients) and the following tree growth, before going to Step 2 or 3 again is at most $O(n^{\mathbf{2}})$. Thus the effort between two consecutive occurrences of Step 2 is $O(n^{\mathbf{3}})$, and therefore the entire method takes at most $O(n^{\mathbf{4}})$. Later on we show that the method can be implemented so that its worst case computational complexity is at most $O(n^{\mathbf{3}})$. If the infeasibility criterion is never satisfied, at termination we will have an assignment x and a dual feasible solution (u, v) which together satisfy the complementary slackness conditions (3.4), so x is an optimum assignment, and (u, v) is an optimum dual solution.

EXAMPLE 3.1 Illustration of the allocation change routine

In this example $n = 6$ and Array 3.2 contains all the relevant information. Admissible cells are those with a zero in the upper left corner.

Allocations are marked with a □ in the cell. All other information is omitted.

When column 6 without an allocation is labeled (Row 6, +) we had a breakthrough, so we put a new allocation in the cell (6, 6). Now look at the label on row 6, which is (Col 1, −). Thus we delete the allocation in cell (6, 1). Continuing this way using the labels, we put a new allocation in (3, 1), delete the one in (3, 4), add on allocation in (4, 4), and reach row 4 labeled $(s, +)$, implying that the allocation change routine is complete. The allocation change path indicated by the labels on Array 3.2 is shown in Figure 3.2. The wavy edges in Figure 3.2 correspond to allocated cells in Array 3.2, on this allocation change path. This path is clearly an alternating path (nodes in it correspond alternatively to unallocated cells, allocated cells in Array 3.2). It is called the alternating predecessor path of column 6 traced by the labels.

Array 3.2

$j =$	1	2	3	4	5	6	Row labels
$i = 1$		0 □					
2			0 □				Col. 3,−
3	0			0 □			Col. 4,−
4			0	0			$(s, +)$
5			0	0			$(s, +)$
6	0 □				0	0	Col 1,−
Column Labels	Row 3 +		Row 4 +	Row 4 +		Row 6 +	

The allocation change routine reverses the roles of unallocated and allocated cells along this path. It has the effect of increasing the number

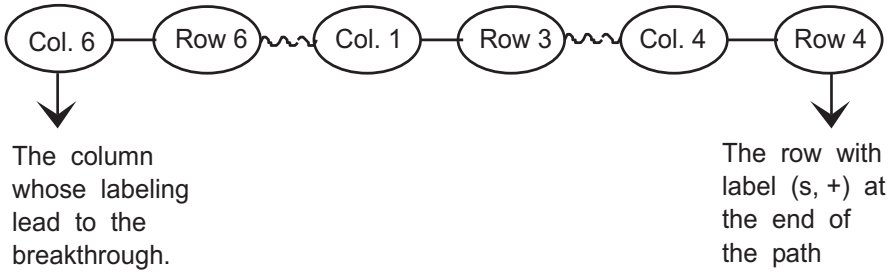

Figure 3.2: Allocation change path.

Array 3.3

$j =$	1	2	3	4	5	6
$i = 1$		0 □				
2			0 □			
3	0 □			0		
4			0	0 □		
5			0	0		
6	0				0	0 □

of allocations by 1. Hence a path like this is called an **augmenting path** in the admissible network. An augmenting path in the admissible network wrt the present allocations is an alternating path of unallocated and allocated arcs, joining a column node and a row node both of which have no allocated arcs incident at them. The tree growth routine discussed above is an efficient scheme to look for such an augmenting path. When a breakthrough occurs, it is an indication that an augmenting path has been identified. In this case the tree is said to have become an **augmenting tree**. The augmenting path is the predecessor path of the column node without an allocated arc

incident at it, whose labeling lead to the breakthrough. If a nonbreakthrough occurs, it is an indication that no augmenting path exists in the admissible network wrt the present allocations.

The new allocations after the allocation change are shown in Array 3.3. The new partial assignment has five allocations, one more than the previous. Now the labels on all the rows and columns are erased, and the algorithm begins another labeling cycle to see whether yet another allocation can be squeezed in among the present admissible cells.

EXAMPLE 3.2

Here we solve the assignment problem of order 4 with the original cost matrix $c = (c_{ij})$ given below.

	c_{ij}				Initial	$c_{ij} - u_i$			
$j =$	1	2	3	4	u_i	$j = 1$	2	3	4
$i = 1$	15	22	13	4	4	11	18	9	0
2	12	21	15	7	7	5	14	8	0
3	16	20	22	6	6	10	14	16	0
4	6	11	8	5	5	1	6	3	0
					$v_j \rightarrow$	1	6	3	0

Total Reduction = 32

1st reduced cost matrix with initial allocations

$j =$	1	2	3	4	u_i	Row labels
$i = 1$	$10 = \bar{c}_{11}$	12	6	0	4	$s, +$
2	4	8	5	0 □	7	Col. 4, $-$
3	9	8	13	0	6	$s, +$
4	0	0	0 □	0	5	
v_j	1	6	3	0	Nonbreakthrough	
Col. labels				Row $3, +$	$\delta = 4$. Resume labeling	

2nd reduced cost matrix

$j=$	1	2	3	4	u_i	Row labels
$i=1$	6	8	2	0	8	$s,+$
2	0	4	1	0 □	11	Col. 4, $-$
3	5	4	9	0	10	$s,+$
4	0	0	0 □	4	5	
v_j	1	6	3	-4	Breakthrough, col. 1 labeled	
Col. labels	Row 2 $+$			Row 3 $3,+$	Total reduction 40	

2nd reduced cost matrix, new allocations, labels

$j=$	1	2	3	4	u_i	Row labels
$i=1$	6	8	2	0	8	$s,+$
2	0 □	4	1	0	11	
3	5	4	9	0 □	10	Col. 4, $-$
4	0	0	0 □	4	5	
v_j	1	6	3	-4	Nonbreakthrough	
Col. labels				Row 1 $1,+$	$\delta=4$. Resume labeling	

Third reduced cost matrix

$j=$	1	2	3	4	u_i	Row labels
$i=1$	4	6	0	0	10	$s,+$
2	0 □	4	1	2	11	
3	3	2	7	0 □	12	Col. 4, −
4	0	0	0 □	6	5	Col. 3, −
v_j	1	6	3	−6	Breakthrough, col. 2 labeled	
Col. labels		Row 4 +	Row 1 +	Row 1, +	Total reduction 42	

3rd reduced cost matrix, new allocations

$j=$	1	2	3	4	u_i
$i=1$	4	6	0 □	0	10
2	0 □	4	1	2	11
3	3	2	7	0 □	12
4	0	0 □	0	6	5
v_j	1	6	3	−6	

Now we have a full assignment, $\{(1, 3), (2, 1), (3, 4), (4, 2)\}$ among the admissible cells, which is an optimum assignment for the problem. Its cost (wrt the original cost matrix) can be verified to be 42, which is also the total reduction at this stage. An optimum dual solution is the (u, v) from the final reduced cost matrix.

An $O(n^3)$ Implementation of the Hungarian Method

An $O(n^3)$ implementation of the Hungarian method is obtained by not updating the entire reduced cost matrix but only portions of it

that are needed, after each dual solution change step. In this implementation an **index** is maintained on each unlabeled column. It is an ordered pair of the form $[t_j, p_j]$ for column j, where p_j is the minimum current reduced cost coefficient in this column among labeled rows, and t_j is the number of a labeled row in which this minimum occurs. The index is maintained in each column as long as it remains unlabeled. The moment the column gets labeled in the method its index is erased. So, at each visit to Step 3 in the method, $\delta = \min \{p_j$: over columns j unlabeled at that time$\}$.

Define a **stage** in the Hungarian method to begin either after getting the initial partial assignment or after an allocation change has been carried out, and to end when the next allocation change has been completed. So there are at most n stages in the method, and Step 3 occurs at most n times in each stage. All reduced cost coefficients are computed once at the beginning of each stage using the dual solution at that time. The reduced cost coefficient is not computed again during this stage in any cell in this implementation.

We will now describe how the work in a stage is organized in this implementation. At the beginning of the stage, compute the entire reduced cost matrix $\bar{c} = (\bar{c}_{ij} = c_{ij} - u_i - v_j)$ wrt the present (u, v). Then carry out Step 1 as usual, and if this leads to Step 2, this stage is completed after it. On the other hand, if this leads to Step 3, before going there, compute for each unlabeled column j, $p_j = \min \{\bar{c}_{ij}$: row i labeled$\}$, and let t_j be an i that ties for this minimum (break ties arbitrarily), and index this column with $[t_j, p_j]$. Then enter Step 3. Each time you have to carry Step 3 in this algorithm, compute $\delta = \min \{p_j$: over columns j unlabeled at that time$\}$. Subtract δ from the p_j index of each unlabeled column j. Let $\mathbf{L}$ be the set of all unlabeled columns j for which p_j became 0 as a result of this operation. For each $j \in \mathbf{L}$, (t_j, j) is a new admissible cell. Put all labeled rows in the list, and resume tree growth (scanning the present labeled rows first) treating $\{(t_j, j) : j \in \mathbf{L}\}$ as the set of new admissible cells among the unlabeled columns. In this tree growth step, each column $j \in \mathbf{L}$ will get labeled. Whenever a column gets labeled, erase its index. Whenever a new row, say row i, is labeled do the following. Because of the way the dual variables are changed in Steps 3 of the algorithm, for each unlabeled

column j the present v_j is the same as at the beginning of this stage, and the same thing is true for u_i since row i remained unlabeled so far and got labeled just now. So, the present reduced cost coefficient in cell (i, j) in unlabeled columns j in this row i is the same as at the beginning of this stage, $\bar{c}_{ij}$. For each unlabeled column j at this time let $[t_j, p_j]$ be the present index on it. If $\bar{c}_{ij} = 0$ label column j with the label (row i, $+$). If $p_j > \bar{c}_{ij} > 0$ change the index on column j to $[i, \bar{c}_{ij}]$; if $p_j \leqq \bar{c}_{ij} > 0$ leave the index on column j unchanged. Repeat this same process each time Step 3 has to be carried out in this stage.

The efficiency of this implementation stems from the fact that it updates the indices on unlabeled columns during a stage without recomputing the reduced cost coefficients after each dual solution change. Since the reduced cost coefficient in any cell is computed once only in each stage, the computational effort per stage can be verified to be at most $O(n^2)$. Since there are at most n stages, the overall computational complexity of the Hungarian method with this implementation is at most $O(n^3)$.

Algorithm to Obtain a Maximum Cardinality Independent Set of Admissible Cells

Suppose we are given a subset of cells of the $p \times n$ transportation array called **admissible cells**. A subset of admissible cells is said to be **independent** if no two cells in it lie in the same row or column. Let a line refer to either a row or column of the array. A subset of lines is said to **cover** all the admissible cells (or to **form a covering set of lines**) if each admissible cell is contained in a row or column (or both) from the subset. Every subset of an independent set is obviously independent, but an independent set may loose its independence when a new cell is introduced into it. So, the problem of finding a maximum cardinality independent set is mathematically interesting. Similarly, a covering set of lines continues to possess this property when new lines are introduced into it, but may no longer remain a covering set if lines are deleted from it. Hence the problem of finding a minimum cardinality covering set of lines is a mathematically interesting problem. The famous König-Egerváry Theorem in bipartite network theory (see

Exercise 3.13) states that the cardinalities of maximum cardinality independent sets of cells and minimum cardinality covering sets of lines are always equal.

When an allocation is made in each cell of an independent set, we get a partial assignment. So, a maximum cardinality independent set can be obtained by finding a partial assignment with the maximum number of allocations among admissible cells. For this we can start with 0 allocations in the $p \times n$ array, and carry out Steps 1 and 2 of the Hungarian method until the tree growth routine ends in a nonbreakthrough. The set of cells with allocations at termination is a maximum cardinality independent set, and the set consisting of unlabeled rows and labeled columns is a minimum cardinality covering set of lines.

Exercises

3.1 Solve the assignment problems with the following cost matrices, to minimize cost.

$$\begin{pmatrix} 10 & 11 & 10 & 5 & 6 & 4 & 3 \\ 5 & 26 & 14 & 18 & 15 & 10 & 10 \\ 6 & 22 & 18 & 17 & 15 & 8 & 8 \\ 2 & 14 & 16 & 16 & 24 & 25 & 12 \\ 4 & 15 & 19 & 10 & 8 & 14 & 11 \\ 10 & 22 & 22 & 15 & 28 & 24 & 12 \\ 8 & 18 & 21 & 18 & 18 & 18 & 14 \end{pmatrix}, \quad \begin{pmatrix} 121 & 6 & 17 & 9 & 8 & 10 \\ 8 & 33 & 45 & 15 & 20 & 31 \\ 5 & 19 & 30 & 16 & 14 & 22 \\ 7 & 22 & 35 & 25 & 27 & 26 \\ 2 & 10 & 24 & 18 & 31 & 14 \\ 4 & 12 & 31 & 17 & 18 & 18 \end{pmatrix}$$

3.2 The coach of a swim team needs to assign swimmers to a 200-yard medley relay team. The "best times" (in seconds for 50 yards) achieved by his five swimmers in each of the strokes are given below. Which swimmer should the coach assign to each of the four strokes?

Stroke	Carl	Chris	Ram	Tony	Ken
Backstroke	37.7	32.9	33.8	37.0	35.4
Breast Stroke	43.4	33.1	42.2	34.7	41.8
Butterfly	33.3	28.5	38.9	30.4	33.6
Freestyle	29.2	26.4	29.6	28.5	31.1

3.3 There are n boys and n girls. Friendship between boys and girls is a mutual relationship (i.e., a boy and a girl are either friends of each other or not). b_j is the number of boys among the n who are friends of girl j, and a_i is the number of girls among the n who are friends of boy i,$i, j = 1$ to n. If all the a_i and b_j are equal to a positive number γ, prove that it is possible to form n boy-girl friendly couples.

3.4 A colonel has five positions to fill and five eligible candidates to fill them. The number of years of experience of each candidate in each field is given in the following table. How should the candidates be assigned to positions to give the greatest total years of experience for all jobs?

	Position				
Candidate	Adjutant	Intelli.	Operations	Supply	Training
1	3	5	6	2	2
2	2	3	5	3	2
3	3	0	4	2	2
4	3	0	3	2	2
5	0	3	0	1	0

3.5 Find a minimum cost assignment wrt the cost matrix given below. No allocations are allowed in cells with a dot in them.

$$\begin{pmatrix} . & 12 & . & 6 & 9 & 6 & . & . \\ . & 8 & . & 4 & 3 & 8 & . & . \\ . & 3 & . & 18 & 3 & 19 & . & . \\ . & 1 & . & 6 & 5 & 11 & . & . \\ 5 & 1 & 13 & 4 & 5 & 6 & 1 & 2 \\ . & 13 & . & 12 & 3 & 1 & . & . \\ 3 & 12 & 3 & 7 & 13 & 6 & 8 & 3 \\ 13 & 4 & 1 & 5 & 5 & 5 & 4 & 9 \end{pmatrix}$$

3.6 Let $c = (c_{ij})$ where $c_{ij} = (i-1)(j-1)$, for $i, j = 1$ to n; $\bar{c} = (\bar{c}_{ij})$, where $\bar{c}_{ij} = (i+j-n)(i+j-n-1)/2$, for $i, j = 1$ to n; $\bar{x} = (\bar{x}_{ij})$ where $\bar{x}_{ij} = 1$ if $i + j = n + 1$, 0 otherwise, for $i, j = 1$ to n.

For $n = 5$, solve the assignment problem with c as the cost matrix by the Hungarian method. Show that $\bar{c}$ is the final reduced cost matrix

at termination, and that $\bar{x}$ is the optimum assignment. Show that these results are true for any $n \geq 2$, and that the Hungarian method goes through $(n-1)(n-2)/2$ breakthroughs and nonbreakthroughs put together before solving this problem. (Silver [1960], Machol and Wien [1977]).

3.7 Prove that every feasible solution x of (3.1) can be expressed as a convex combination of assignments of order n.

3.8 The HQ of a large company has n gates, each of which is manned by a night watchman every night. There are n night watchmen on the payroll, and they are rotated among the gates for security reasons. A planning period consists of r nights, and $x = (x_{ij})$ is a nonnegative integral matrix satisfying $\sum_{j=1}^{n} x_{ij} = r, \sum_{i=1}^{n} x_{ij} = r$, for all $i, j = 1$ to n. It is required to assign night watchman to gates over the period (one watchman per gate per night) so that the ith watchman is assigned to watch the jth gate for exactly x_{ij} nights. Prove that an assignment like that exists, and develop an efficient algorithm to find it. Apply your algorithm on the problem in which $n = 5, r = 30$ and x is the following matrix, and generate an assignment of the 5 night watchman to the 5 gates over the 30 nights of the planning period, satisfying the conditions mentioned above.

$$x = \begin{pmatrix} 3 & 10 & 9 & 5 & 3 \\ 8 & 2 & 5 & 10 & 5 \\ 11 & 4 & 10 & 5 & 0 \\ 0 & 7 & 3 & 4 & 16 \\ 8 & 7 & 3 & 6 & 6 \end{pmatrix}$$

(D. Gale)

3.9 An assignment problem of order 9 is being solved by the $O(n^3)$ implementation of the Hungarian method discussed above. Relevant information on the array when a nonbreakthrough has just occurred is given in the following array. $\square$ indicates an allocation in that cell in the present partial assignment. In some of the cells the original cost coefficient is given at the bottom right corner. The present labels on all the labeled rows and labeled columns, and the indices on the unlabeled

columns are given. The present dual feasible solution is also given. Continue the application of the method until the next breakthrough occurs.

$j=$	1	2	3	4	5	6	7	8	9	label	u_i
$i=1$										$s,+$	200
2										$s,+$	200
3	□									1, −	200
4			□							3, −	200
5		□								2, −	200
6				□	400	400	400	400	400		200
7				400	□	400	400	400	350		200
8				400	400	□	400	400	400		200
9				400	400	400	□	400	400		200
label	1, +	1, +	2, +								
index				[2,40]	[3,40]	[2,80]	[4,100]	[5,110]	[5,110]		
v_j	100	100	100	100	100	100	100	100	100		

3.10 Prove that the number of labeled columns plus the number of unlabeled rows is equal to the number of allocated cells, at the occurrence of a nonbreakthrough in the Hungarian method. At the same point in the method, prove that the number of labeled rows minus the number of labeled columns is $\geq$ the number of rows without allocations. Using this prove that the total reduction strictly increases whenever the dual solution changes in the method.

Give a proof of the primal infeasibility criterion in the Hungarian method (that there is no feasible assignment if $\delta = +\infty$ in some dual solution change step) using the duality theorem of LP.

3.11 Consider an assignment problem of order n in which x_{ij} is required to be 0 whenever $j \notin \mathbf{S}_i$ for each $i = 1$ to n, where $\mathbf{S}_1, \ldots, \mathbf{S}_n$ are

all subsets of $\{1, \ldots, n\}$ which are given. Determine the necessary and sufficient conditions that these sets have to satisfy, for the existence of a feasible assignment to the problem.

3.12 Evaluate the worst case computational complexity of the Hungarian method to solve a sparse assignment problem of order n as a function of order n and m ($n^2 - m$ is the number of variables required to be 0 in the problem).

3.13 Let $\mathbf{Q}$ be a specified subset of admissible cells in the $p \times n$ array. Set up a directed network $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N} = \{\check{s}, R_1, \ldots, R_p, C_1, \ldots, C_n, \check{t}\}$ ($\check{s}, \check{t}$, are the supersource, supersink respectively, and R_i, C_j correspond to row i, col j of the array, for $i = 1$ to p, $j = 1$ to n), $\mathcal{A} = \{(\check{s}, R_i) : i = 1 \text{ to } p\} \cup \{(R_i, C_j) : i, j \text{ s. t. cell } (i, j) \in \mathbf{Q}\} \cup \{(C_j, \check{t}) : j = 1 \text{ to } n\}$, as in Figure 3.2. Make the lower bound for flow on all the arcs in $\mathcal{A}$ zero, and the capacities for flow on all the arcs $(\check{s}, R_i)$ and $(C_j, \check{t})$ equal to 1, and ∞ for all the arcs of the form $(R_i, C_j) \in \mathcal{A}$.

(i) Prove that the maximum cardinality among independent sets of admissible cells in the array is equal to the maximum value of flow from $\check{s}$ to $\check{t}$ in G. Given an integral maximum value flow vector in G, discuss how to construct a maximum cardinality independent set of admissible cells in the array from it, and vice versa.

(ii) Let $[\mathbf{X}, \bar{\mathbf{X}}]$ be a cut separating $\check{s}$ and $\check{t}$ in G. Prove that the capacity of this cut is finite iff there are no arcs of the form $(R_i, C_j) \in \mathcal{A}$ with $R_i \in \mathbf{X}$, $C_j \in \bar{\mathbf{X}}$ and $(i, j) \in \mathbf{Q}$. Using this prove that in this case, the set of lines $\{\text{Row } i : i \in \bar{\mathbf{X}}\} \cup \{\text{Column } j : j \in \mathbf{X}\}$ is a covering set of lines, and that the cardinality of this covering set is equal to the capacity of this cut $[\mathbf{X}, \bar{\mathbf{X}}]$ in G. Conversely, given a covering set of lines in the array, prove that $[\mathbf{X}, \bar{\mathbf{X}}]$, where $\mathbf{X} = \{\check{s}\} \cup \{\text{Row } i\text{: row } i \text{ is not in the covering set}\} \cup \{\text{Column } j\text{: column } j \text{ is in the covering set}\}$, and $\bar{\mathbf{X}} = \mathcal{N} \setminus \mathbf{X}$, is a cut separating $\check{s}$ and $\check{t}$ in G whose capacity is equal to the cardinality of this covering set of lines. Using these prove that the problem of finding a minimum cardinality cover for $\mathbf{Q}$ in the array is equivalent to that of finding a minimum capacity cut separating $\check{s}$ and $\check{t}$ in G.

(iii) Using these prove the **König-Egerváry Theorem** which states that the maximum cardinality among independent sets is equal to the minimum cardinality among covering sets of lines, by applying the maximum flow minimum cut theorem on G.

(iv) Prove that the covering set of lines obtained at termination of the algorithm discussed above is a minimum cardinality covering set of lines.

3.14 $c = (c_{ij})$ is the original cost matrix for an assignment problem of order n. For any cell (i, j) in the $n \times n$ array, let $a(i, j)$ denote a minimum cost assignment among those containing an allocation in cell (i, j). For a given r, the following method finds $a(r, 1), \ldots, a(r, n)$.

Step 1 Find a minimum cost assignment with c as the cost matrix by the Hungarian method. Let $a_1 = \{(1, j_1), \ldots, (n, j_n)\}$ be the optimum assignment obtained, and $\bar{c}$ the final reduced cost matrix.

Step 2 Let α be a positive number $>$ every entry in $\bar{c}$. Add α to all the elements in row r of $\bar{c}$. In the resulting matrix, subtract α from each entry in column j_r.

Step 3 Find a most negative entry in the present matrix. Suppose it appears in row i and has absolute value β. Add β to every entry in row i of the present matrix. In the resulting matrix subtract β from every entry in column j_i.

Step 4 Repeat Step 3 as often as necessary, until the present matrix becomes $\geqq 0$. Then go to Step 5.

Step 5 Let $\tilde{c} = (\tilde{c}_{ij})$ be the matrix obtained at the end. For $1 \leqq q \leqq n$, there exists an assignment with an allocation in cell (r, q) and all the other allocations in cells (i, j) with $\tilde{c}_{ij} = 0$. Any such assignment is $a(r, q)$.

(i) Prove that Step 3 has to be repeated exactly $n - 1$ times before going to Step 5 in this method.

(ii) Prove the statement in Step 5, that for each $q = 1$ to n, there exists an assignment with an allocation in cell (r, q), and all other allocations in cells with 0 entries in $\tilde{c}$, and any such assignment is indeed a minimum cost assignment among those with an allocation in (r, q).

(iii) Derive the worst case computational complexity of this method. Apply the method on the problem with $n = 5$, $r = 1$ and

$$c = \begin{pmatrix} 0 & 3 & 9 & 4 & 0 \\ 2 & 0 & 7 & 0 & 11 \\ 5 & 15 & 0 & 16 & 12 \\ 4 & 0 & 18 & 0 & 17 \\ 0 & 20 & 21 & 13 & 0 \end{pmatrix}$$

(Kreuzberger and Weiterstadt, [1971], "Eine Methode zur Bestimmung mehrerer Losungen Furdas Zuordnungsproblem," *Angewandte Informatik*, 13, no. 9 (407-414), in German)

Comment 3.1 The Hungarian method for the assignment problem is due to Kuhn [1955]. The name for the method recognizes the work of the Hungarian mathematicians J. Egerváry and D. König (the König-Egerváry Theorem, see Exercise 3.13) which is the basis for the method. Each maximum value flow problem encountered in the method is of the König-Egerváry type (i.e., that of finding a maximum cardinality set of independent admissible cells). The $O(n^3)$ implementation of the Hungarian method is due to Lawler [1976 of Chapter 1].

****

3.1.1 Minimal Chain Decompositions in Partially Ordered Sets

Suppose we are given a finite set of n elements, $\mathbf{P}$. We will number the elements serially and represent each element by its number, thus

$\mathbf{P} = \{1, \ldots, n\}$. A strict partial order on $\mathbf{P}$ is an order relation between some pairs of elements of $\mathbf{P}$, denoted by $\succ$, satisfying : $i \not\succ i$ for all i; $i \succ j$ implies $j \not\succ i$; and the following property called **transitivity** : for i, j, h, $i \succ j$, $j \succ h$ implies $i \succ h$. This relationship makes $\mathbf{P}$ a **partially ordered set** *or* **poset**, which we also denote by the same symbol $\mathbf{P}$. The partial order can be represented by a directed network with the elements in $\mathbf{P}$ as its nodes and arcs (i, j) if $i \succ j$. Normally if $i \succ j$ and $j \succ h$, we have $i \succ h$ by transitivity, but we do not include the arc (i, h) in this network, even though it is quite harmless to do so. Thus whenever $i \succ j$, either there is an arc (i, j), or there exists a chain from node i to node j in the network; and conversely. Thus $i \succ j$ iff there exists a chain from i to j in this network representation. The network thus constructed has no directed circuits by the properties of the partial order, see Figure 3.3 for an example.

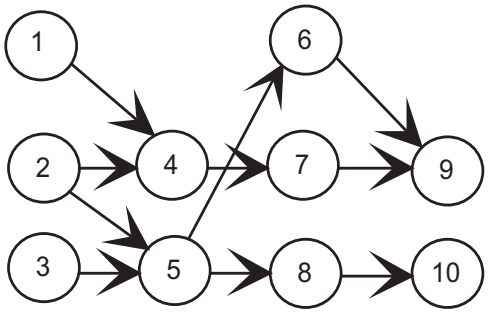

Figure 3.3:

We define a **chain** in this poset to be a set of one or more elements $i_1, i_2, \ldots, i_t$ of $\mathbf{P}$ satisfying $i_1 \succ i_2 \succ \ldots \succ i_t$; thus it corresponds to a chain in the network when there are two or more elements in it. However, a single element by itself (that is, a single node in the network) is also considered a chain of the poset.

A **decomposition** of this poset is a partition of the set $\mathbf{P}$ into chains which are mutually disjoint. The trivial decomposition of $\mathbf{P}$ into n one-element chains is an example of a decomposition. A decomposition with the smallest number of chains in it is said to be a **minimal decomposition**. We will use the symbol $\triangle$ to denote decompositions

and $|\triangle|$ to denote the number of disjoint chains in $\triangle$.

A subset $\mathbf{N} \subset \mathbf{P}$ is said to be **a set of unrelated elements** in the poset if for every pair $i, j \in \mathbf{N}$, neither $i \succ j$, nor $j \succ i$.

As an example, consider the poset on $\mathbf{P} = \{1 \text{ to } 10\}$ represented by the network in Figure 3.3. In this poset, the set of all j satisfying $2 \succ j$ is {4, 5, 6, 7, 8, 9, 10}. The sets {1, 2, 3}, {6, 7, 8} are both sets of unrelated elements, but the set {5, 6, 7, 8} is not since $5 \succ 6$. $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ is a chain decomposition of this poset where

$$\begin{aligned} \mathcal{C}_1 &= 1, (1,4), 4, (4,7), 7, (7,9), 9 \\ \mathcal{C}_2 &= 2, (2,5), 5, (5,8), 8, (8,10), 10 \\ \mathcal{C}_3 &= 3; \quad \mathcal{C}_4 = 6 \end{aligned} \tag{3.5}$$

In any decomposition, each node in a set of unrelated elements $\mathbf{N}$ has to appear on a different chain (as otherwise there will be two nodes i, j in $\mathbf{N}$ such that either $i \succ j$ or $j \succ i$, a contradiction). Hence the number of disjoint chains in any decomposition of $\mathbf{P}$ is $\geq$ the cardinality of any set of unrelated elements in $\mathbf{P}$. Hence we have the result.

$$\left.\begin{array}{c}\text{no. of chains in minimal} \\ \text{decomposition of } \mathbf{P}\end{array}\right\} \geq \left\{\begin{array}{l}\text{maximum no. of mutually} \\ \text{unrelated elements of } \mathbf{P}\end{array}\right. \tag{3.6}$$

The finite case of the well-known Dilworth's chain decomposition theorem for posets asserts that equality holds in (3.6). Here we show how to prove this result as a corollary of the König-Egerváry theorem, through a bipartite network formulation. Also, we show that a minimal decomposition of $\mathbf{P}$ and a maximum cardinality set of unrelated elements in $\mathbf{P}$ can both be obtained using the algorithms discussed earlier. These results are due to Fulkerson [1956].

Construct the bipartite network G = ($\mathcal{N}_1$, $\mathcal{N}_2$; $\mathcal{A}$) where $\mathcal{N}_1 = \{R_1, \ldots, R_n\}$, $\mathcal{N}_2, = \{C_1, \ldots, C_n\}$, and $\mathcal{A} = \{(R_i, C_j) : i \succ j\}$. Notice that G contains arcs corresponding to all order relations implied by transitivity, that is, $(R_j, C_j) \in \mathcal{A}$ whenever $i, j \in \mathbf{P}$ are such that $i \succ j$. Corresponding to G, set up an $n \times n$ array with R_i, C_j associated with row i, column j of the array, and the arc (R_i, C_j) in G associated with

the cell (i, j) in the array. Make the cell (i, j) in the array admissible iff (R_i, C_j) is an arc in G. Then an independent set of admissible cells in the array corresponds to a matching in G and vice versa. If $\mathbf{M} = \{(R_{i_1}, C_{i_2}), (R_{i_3}, C_{i_4}), \ldots, (R_{i_{2l-1}}, C_{i_{2l}})\}$ is a matching in G with $|\mathbf{M}| = l$, $\{(i_1, i_2), (i_3, i_4), \ldots, (i_{2l-1}, i_{2l})\}$ is the corresponding independent set of admissible cells in the associated array, and vice versa. We can group the distinct elements in the set $\{i_1, i_2, \ldots, i_{2l}\}$ into chains, each one passing through two or more elements. Since $\mathbf{M}$ is a matching in G, these chains will be disjoint. Let r be the number of these chains, and let $p_1, p_2, \ldots, p_r$ be the numbers of elements on them. So, the numbers of arcs on these chains are $p_1 - 1, \ldots, p_r - 1$ respectively, and hence $l = |\mathbf{M}| = (p_1 - 1) + \ldots + (p_r - 1)$. The total numbers of elements of $\mathbf{P}$ which do not appear on any of these chains is $n - (p_1 + \ldots + p_r) = n_1$. Add a one-element chain at each of these elements to the set of r chains obtained above. This leads to a decomposition $\triangle$ of $\mathbf{P}$, with $|\triangle| = n_1 + r$. Now, $n = (n - (p_1 + \ldots + p_r)) + p_1 + \ldots + p_r = n_1 + r + ((p_1 - 1) + \ldots + (p_r - 1)) = |\triangle| + |\mathbf{M}|$. So, corresponding to any matching $\mathbf{M}$ in G, we can construct a decomposition $\triangle$ of $\mathbf{P}$ such that $|\triangle| = n - |\mathbf{M}|$.

As an example, consider the poset represented by the acyclic network in Figure 3.3. Here $n = 10$, and the array corresponding to this partially ordered set is given below, with admissible cells marked by a x.

Array 3.4

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
R_1				x			x		x	
R_2				x	x	x	x	x	x	x
R_3					x	x		x	x	x
R_4							x		x	
R_5						x		x	x	x
R_6									x	
R_7									x	
R_8										x
R_9										
R_{10}										

An independent set of admissible cells in Array 3.4 is $\mathbf{M} = \{(1, 4), (2, 5), (4, 7), (5, 8), (7, 9), (8, 10)\}$. This independent set of admissible cells corresponds to the chain decomposition $\triangle = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ of the partially ordered set, where $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ are given in (3.5). It can be verified that $|\mathbf{M}| + |\triangle| = n = 10$ here.

Let $\mathbf{X} = \{R_{t_1}, \ldots, R_{t_w}, : C_{j_1}, \ldots, C_{j_q}\}$ be a set of lines (rows and columns) in the array corresponding to the poset $\mathbf{P}$, which covers all the admissible cells. Suppose $\mathbf{X}$ is a proper cover, that is, no subset of $\mathbf{X}$ covers all the admissible cells in the array. Then we claim that $t_1, \ldots, t_w, j_1, \ldots, j_q$ are all distinct. To see this, suppose $t_1 = j_1$. Since $\mathbf{X}$ is a proper cover, there must exist an r such that $R_r \notin \mathbf{X}$ and (R_r, C_{j_1}) is an admissible cell. Similarly there must exist an s such that $C_s \notin \mathbf{X}$ and (R_{t_1}, C_s) is admissible. By transitivity and the assumption that $t_1 = j_1$, it follows that (R_r, C_s) is an admissible cell, and since neither R_r, nor C_s is in $\mathbf{X}$, this contradicts the assumption that $\mathbf{X}$ covers all admissible cells. So, elements in $\{t_1, \ldots, t_w;\ j_1, \ldots, j_q\}$ are all distinct. Let $\mathbf{U} = \{1, \ldots, n\} \backslash \{t_1, \ldots, t_w;\ j_1, \ldots, j_q\}$. Since $\mathbf{X}$ covers all admissible cells, the elements in $\mathbf{U}$ are mutually unrelated, and by the definition of $\mathbf{U}$ we have $\{t_1, \ldots, t_w, j_1, \ldots, j_q\} \cup \mathbf{U} = \mathbf{P}$, and hence $|\mathbf{X}| + |\mathbf{U}| = n$. So, corresponding to any covering set $\mathbf{X}$ of lines in the array, we can construct a mutually unrelated set of elements $\mathbf{U}$ of $\mathbf{P}$ such that $|\mathbf{U}| = n - |\mathbf{X}|$.

As an example, consider the poset represented by the acyclic network in Figure 3.3. Array 3.4 corresponds to it. In Array 3.4, all admissible cells are covered by the set of lines $\{R_1, R_2, R_3, R_4, R_5, R_8, C_9\}$. This covering set of lines leads to the set of unrelated elements $\mathbf{U}$=$\{6, 7, 10\}$ in $\mathbf{P}$.

THEOREM 3.1 *(Dilworth's Theorem) The number of chains in a minimal decomposition of a finite poset* $\mathbf{P}$ *is equal to the maximum number of mutually unrelated elements in* $\mathbf{P}$.

Proof Construct the array corresponding to $\mathbf{P}$ as described above. In this array find a maximum cardinality set of independent admissible cells $\hat{\mathbf{M}}$, and a minimum cardinality set of covering lines $\hat{\mathbf{L}}$ covering all the admissible cells. Obtain the chain decomposition $\hat{\triangle}$ of $\mathbf{P}$ corresponding to $\hat{\mathbf{M}}$, and a set $\hat{\mathbf{U}}$ of mutually unrelated elements

of $\mathbf{P}$ corresponding to $\hat{\mathbf{L}}$ by the procedures described above. We have $|\hat{\triangle}| = |\mathbf{P}| - |\hat{\mathbf{M}}|$ and $|\hat{\mathbf{U}}| = |\mathbf{P}| - |\hat{\mathbf{L}}|$, and by König-Egerváry theorem $|\hat{\mathbf{M}}| = |\hat{\mathbf{L}}|$. So $|\hat{\triangle}| = |\hat{\mathbf{U}}|$. But by (3.6) we have $|\triangle| \geqq |\mathbf{U}|$ for every chain decomposition $\triangle$ and set of unrelated elements $\hat{\mathbf{U}}$ in $\mathbf{P}$ and $|\hat{\triangle}| = |\hat{\mathbf{U}}|$. This implies that $\hat{\triangle}$ is a minimal decomposition of $\mathbf{P}$ and $\hat{\mathbf{U}}$ is a maximum cardinaltiy set of mutually unrelated elements in $\mathbf{P}$, and since $|\hat{\triangle}| = |\hat{\mathbf{U}}|$, the theorem is proved. ∎

Hence to find a minimal chain decomposition of a finite poset $\mathbf{P}$, we construct the array corresponding to it as described above, and then find a maximum cardinality set of independent admissible cells, $\mathbf{M}$, in it, using the algorithm discussed earlier. A minimal chain decomposition of $\mathbf{P}$ is obtained directly using $\mathbf{M}$ as described above.

3.1.2 The Bottleneck Assignment Problem

In the assignment problem (3.1), the objective function is the sum of the costs of all the allocations. This objective function may not be appropriate in some practical applications. As an example, consider the application discussed in Exercise 3.15 at the end of this section. Minimizing the sum of the inconvenience of all the salesmen may not please a particular salesman if his own individual inconvenience turns out to be large in the optimum assignment. A better objective in this case, is to minimize the maximum inconvenience experienced by any salesman. This leads to a problem known as the **bottleneck assignment problem** or the **min-max assignment problem**. If $c = (c_{ij})$ is the square cost matrix of order n, in this problem our aim is to find an assignment $x = (x_{ij})$ that minimizes the function $f(x) =$ maximum $\{c_{ij} : i, j$ such that $x_{ij} = 1\}$ over the set of all assignments of order n. We discuss an algorithm for this problem now. It uses the tree growth steps in the Hungarian method. To avoid confusion we refer to steps in this method as items. In this method the value of z^r is like a threshold. At each stage, all cells with cost coefficient $\leqq$ threshold are admissible. If there is no assignment among the set of admissible cells, the threshold is increased to the next higher level. The method terminates as soon as a full assignment is located among the set of admissible cells. Hence, this method is known as the **threshold method**

for the bottleneck assignment problem. It is due to Gross [1959]. The assignment at termination is an optimum assignment, and the terminal value of z^r is the optimum objective value in the bottleneck problem.

THE THRESHOLD METHOD

Item 0 Define $u_i = \min.\ \{c_{i1}, \ldots, c_{in}\}$, $i = 1$ to n, $v_j = \min.\ \{c_{1j}, \ldots, c_{nj}\}$, $j = 1$ to n, $z^1 = \max.\ \{u_1, \ldots, u_n; v_1, \ldots, v_n\}$. Define the set of **admissible cells** to be $\{(i,j) : i,j \text{ s. t. } c_{ij} \leqq z^1\}$. With this set of admissible cells and any partial assignment with allocations only among admissible cells (for example, the 0 partial assignment), enter the tree growth routine (Step 1) of the Hungarian method and make allocation changes as breakthroughs occur, until either a full assignment is obtained or a nonbreakthrough occurs. If a full assignment is obtained, it is an optimum assignment; terminate. If a nonbreakthrough occurs, go to Item 1.

Item 1 Let r be the number of the present visit to this iteration. Define $z^{r+1} = \min.\{c_{ij} : (i,j) \text{ is inadmissible at this stage}\}$. Let the new set of admissible cells be $\{(i,j) : c_{ij} \leqq z^{r+1}\}$. Make the list = set of all labeled rows, and resume tree growth by going to Substep 2 in Step 1 of the Hungarian method, and continuing as in Item 0.

Exercises

3.15 There are n salesmen to be assigned to n markets on a one to one basis. c_{ij} measures the inconvenience experienced by the ith salesman if he is assigned to market j (for example, c_{ij} may be the daily commuting distance for him under this assignment). It is required to find an assignment that minimizes the maximum inconvenience experienced by any salesman. Solve this problem when $c = (c_{ij})$ is the matrix given below.

$$c = \begin{pmatrix} 3 & 9 & 15 & 20 & 5 \\ 13 & 16 & 7 & 8 & 9 \\ 19 & 12 & 13 & 14 & 15 \\ 7 & 17 & 8 & 9 & 13 \\ 9 & 15 & 16 & 12 & 11 \end{pmatrix}$$

3.16 Each visit to Item 1 in the the threshold method increases the number of admissible cells in the array by at least one. Hence Item 1 occurs at most n^2 times in the method. Using this, derive the worst case computational complexity of this method.

3.17 An assembly line consists of workstations $1, \ldots, n$, each staffed by an operator. Each unit is processed on each workstation as it passes along, it cannot move past a workstation until its processing there is completed. There are n laborers, and c_{ij} is the number of seconds that the jth laborer takes to process a unit at workstation i. It is required to determine how the laborers should be assigned to the workstations on a one to one basis, so as to maximize the productivity of the line. Formulate this problem. Obtain an optimum solution when $c = (c_{ij})$ is the following matrix.

$$c = \begin{pmatrix} 9 & 7 & 18 & 13 & 14 & 16 \\ 16 & 12 & 11 & 23 & 5 & 19 \\ 13 & 9 & 25 & 19 & 17 & 18 \\ 18 & 16 & 10 & 9 & 13 & 11 \\ 12 & 12 & 8 & 18 & 8 & 9 \\ 6 & 11 & 7 & 19 & 9 & 12 \end{pmatrix}$$

3.2 The Primal-Dual Method for the Uncapacitated Balanced Transportation Problem

A commodity has to be shipped from p sources (ith source has a_i units available to ship, $i = 1$ to p) to n markets (jth market needs b_j units,

$j = 1$ to n). c_{ij} is the unit transportation cost (\$/unit) on the route from source i to market j, $i = 1$ to p, $j = 1$ to n. The data satisfies

$$a_i, b_j > 0, \text{ for all } i, j; \text{ and } \sum a_i = \sum b_j$$

x_{ij} denotes the amount of material shipped from source i to market j. These x_{ij} are the decision variables in the problem. It is

$$\begin{aligned} \text{Minimize } z(x) &= \sum_{i=1}^{p}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{Subject to } \sum_{j=1}^{n} x_{ij} &= a_i,\ i = 1 \text{ to } p \\ \sum_{i=1}^{p} x_{ij} &= b_j,\ j = 1 \text{ to } n \\ x_{ij} &\geqq 0, \text{ for all } i, j \end{aligned} \tag{3.7}$$

It is the problem of finding a minimum cost flow vector saturating all the arcs leading to the super sink in the bipartite network in Figure 3.4. All lower bounds are 0. Data on each arc is capacity; unit cost coefficient in that order. x_{ij} is the flow on the arc joining source i and market j. By identifying the cell (i, j) in the $p \times n$ transportation array with the arc joining source i to market j in the network, all the computations can actually be carried out on the array itself.

In practical applications, each source may not be able to ship to all the sinks. If sink j is too far away from source i, it is realistic to specify that source i cannot ship to sink j (i.e., that $x_{ij} = 0$). There may also be other practical reasons why a source cannot ship to some sinks. Thus, for each source, a subset of sinks to which it can ship is specified, and all flows from that source to sinks outside that set are required to be zero. This can be handled by defining c_{ij} to be $+\infty$ whenever x_{ij} is required to be zero in the formulation (3.7). In the corresponding bipartite network in Figure 3.4, the arc (i, j) is not included if x_{ij} is required to be 0; thus it is no longer the complete bipartite network. We denote by m the number of source to market arcs in this network. Clearly $m \leqq pn$, the transportation problem is

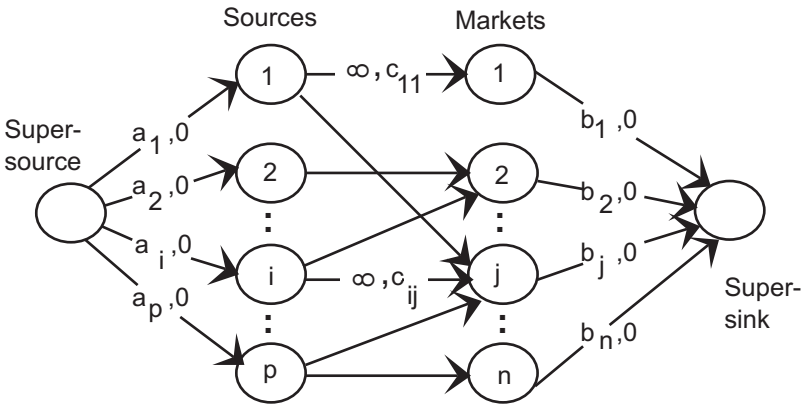

Figure 3.4:

said to be **sparse** if m is small compared to pn, and **dense** if m is close to pn.

The primal-dual algorithm is described using arrays for ease of understanding, but computer implementations are usually based on the network formulation as described above, this saves in memory requirements and running time for solving practical problems which are almost always sparse. The dual problem is

$$\begin{aligned} \text{Maximize } & \sum a_i u_i \quad + \quad \sum b_j v_j \\ \text{Subject to } & u_i + v_j \quad \leqq \quad c_{ij} \quad \text{for all } i, j \end{aligned} \tag{3.8}$$

Let $\bar{c}_{ij} = c_{ij} - u_i - v_j$, for $i = 1$ to p, $j = 1$ to n. These are the **reduced cost coefficients** wrt (u, v), and (u, v) is dual feasible iff they are all $\geqq 0$. The complementary slackness conditions for optimality in these problems are

$$x_{ij}\bar{c}_{ij} = 0 \text{ for all } i, j. \tag{3.9}$$

The cell (i, j) in the array (the corresponding arc (i, j) in the bipartite network in Figure 3.4) is said to be an **admissible** or **equality cell** (**admissible** or **equality arc**) wrt (u, v) if $\bar{c}_{ij} = 0$; otherwise it is **inadmissible**. The network obtained by deleting all the inadmissible

arcs from Figure 3.4 is known as the **admissible** *or* **equality subnetwork** wrt (u, v). The complementary slackness conditions require that the flow amounts should be 0 on all the inadmissible arcs. The flow problem, known as the **restricted primal** at this stage, is to find a maximum value flow from the super source to the super sink in the equality subnetwork. It is equivalent to

$$\begin{aligned}
\text{Maximize } \sum(x_{ij} \quad &: \quad \text{over } (i,j) \text{ admissible }) \\
\text{Subject to } \sum_{j=1}^{n} x_{ij} \quad &\leqq \quad a_i,\ i = 1 \text{ to } p \\
\sum_{i=1}^{p} x_{ij} \quad &\leqq \quad b_j,\ j = 1 \text{ to } n \qquad (3.10) \\
x_{ij} \quad &\geqq \quad 0, \quad \text{if } (i,j) \text{ admissible, } 0 \text{ otherwise}
\end{aligned}$$

The primal-dual algorithm maintains vectors $x, (u, v)$ which always satisfy the constraints in (3.10), (3.8), (3.9). When the x vector satisfies the equality constraints in (3.7), it is an optimum solution and the method terminates.

THE CLASSICAL PRIMAL-DUAL METHOD FOR THE UNCAPACITATED BALANCED TRANSPORTATION PROBLEM

Step 0 Initialization If some dual feasible solution is available, use it as the initial one, otherwise define it to be $(u^1 = (u_i^1), v^1 = (v_j^1))$ where $u_i^1 = \min \{ c_{ij} : j = 1 \text{ to } n \}$, $v_j^1 = \min \{ c_{ij} - u_i^1 : i = 1 \text{ to } p \}$, for $i = 1$ to p, $j = 1$ to n. List $= \emptyset$. Define $x^1 = 0$. Go to Step 1.

Step 1 Tree growth routine Let $\tilde{x} = (\tilde{x}_{ij})$ be the present flow.

Substep 1 Label each row i satisfying $\sum_j \tilde{x}_{ij} < a_i$ with $(s, +)$, and include it in the list.

Substep 2 If list $= \emptyset$, tree growth has terminated and there is a nonbreakthrough. The present flow is a maximum value flow in the admissible subnetwork, go to Step 3. Otherwise,

select a row or column from the list for scanning and delete it from the list.

Forward labeling Scanning row i consists of labeling each unlabeled column j such that (i, j) is an admissible cell, with the label (row i, $+$).

Reverse labeling To scan column j, label all unlabeled rows i
satisfying $\tilde{x}_{ij} > 0$ with (column j, $-$).

If any column without an allocation has been labeled, there is a breakthrough, go to Step 2. Otherwise, include all newly labeled rows and columns in the list, and repeat this Substep 2.

Step 2 Flow change routine Suppose column j satisfying $\alpha = b_j - \sum_i \tilde{x}_{ij} > 0$ has been labeled. Trace its predecessor path using the labels, suppose it ends with row i with the label of $(s, +)$. This path is an FAP from row i to column j in the equality subnetwork. Call it $\mathcal{P}$. Let $\beta = a_i - \sum_j \tilde{x}_{ij}$, and ϵ the residual capacity of $\mathcal{P}$. Define $\gamma = \min\{\ \alpha, \beta, \epsilon\ \}$. Carry out flow augmentation by the amount γ on the FAP $\mathcal{P}$, and get the new flow $\hat{x}$. If $\hat{x}$ is feasible to (3.7), it is an optimum solution; terminate. Otherwise, chop down the present trees (i.e., erase the labels on all the rows and columns) and go back to Step 1.

Step 3 Dual solution change routine Same as Step 3 of the Hungarian Method.

EXAMPLE 3.3

Consider the balanced transportation problem with data given in the following array. An initial dual feasible solution $(u = (u_i), v = (v_j))$ is also given in the array.

$j=$	1	2	3	4	5	6	a_i	u_i
$i=1$	$c_{11}=5$	3	7	3	8	5	4	3
2	5	6	12	5	7	11	3	5
3	2	8	3	4	8	2	3	2
4	9	6	10	5	10	9	7	5
b_j	3	3	6	2	1	2		
v_j	0	0	1	0	2	0		

The reduced cost coefficients, $\bar{c}_{ij} = c_{ij} - u_i - v_j$ are entered in the upper left corners of the cells in the following array. All cells in which $\bar{c}_{ij} = 0$ are admissible cells, and they are marked with a little box in the middle. An initial flow among the admissible cells is obtained by inspection, and the flow amounts, when nonzero, are entered in the little boxes.

$j=$	1	2	3	4	5	6	a_i	u_i	Row label
$i=1$	$2=\bar{c}_{11}$	0 $\boxed{3}$	3	0 $\boxed{1}$	3	2	4	3	Col. 4, -
2	0 $\square+\theta$	1	6	0 $\square$	0 $\boxed{1}$	6	3	5	s, +
3	0 $\boxed{3}-\theta$	6	0 $\square+\theta$	2	4	0 $\square$	3	2	Col. 1, -
4	4	1	4	0 $\boxed{1}$	3	4	7	5	s, +
b_j	3	3	6	2	1	2			
v_j	0	0	1	0	2	0			
Col. label	Row 2 +		Row 3 +	Row 2 +	Row 2 +				

$j=$	1	2	3	4	5	6	a_i	u_i	Row label
$i=1$	$2=\bar{c}_{11}$	0 [3]	3	0 [1]	3	2	4	3	Col. 4, -
2	0 [2]	1	6	0 □	0 [1]	6	3	5	
3	0 [1]	6	0 [2]	2	4	0 □	3	2	
4	4	1	4	0 [1]	3	4	7	5	s, +
b_j	3	3	6	2	1	2			
v_j	0	0	1	0	2	0			
Col. label		Row 1 +		Row 4 +					

In the first array at the top, rows 2 and 4 have additional material to be shipped, so they are labeled with $(s,+)$. Columns 3 and 6 have unfulfilled requirements. The labeling routine is applied, and it ends in a breakthrough with column 3 labeled. We move to the flow change routine. Entries of $+\theta$ and $-\theta$ are made in admissible cells as indicated by the labels. The value of θ should be min.$\{6 =$ unfulfilled requirement in column 3; $2 =$ additional material available at row 2; $3 =$ flow amount in cell (3, 1) with a $-\theta$ entry$\} = 2$. The new flow vector is recorded in the next array together with the present reduced cost coefficients. The old labels are erased and the labeling routine is applied again. The row and column labels obtained are recorded on the array.

We have a nonbreakthrough. So, we move to the dual solution change routine. $\delta = 2$. The new dual feasible solution is $\hat{u}, \hat{v}$ marked in the following array. In this array we show the new reduced cost coefficients in each cell. The new admissible cells are marked with a little box in the middle. Notice that all the cells with positive flow amounts in the present flow vector remain admissible. The present

flow amounts in cells with positive flow are entered inside the box in them.

$j=$	1	2	3	4	5	6	a_i	$\hat{u}_i$
$i=1$	0 □	0 [3]	1	0 [1]	1	0 □	4	5
2	0 [2]	3	6	2	0 [1]	6	3	5
3	0 [1]	8	0 [2]	4	4	0 □	3	2
4	2	1	2	0 [1]	1	2	7	7
b_j	3	3	6	2	1	2		
$\hat{v}_j$	0	-2	1	-2	2	0		

The algorithm now resumes labeling using the new admissible cells. It can be continued in the same manner until an optimum solution is obtained.

Discussion

If δ turned out to be $+\infty$ in Step 3 at some stage, the flow at that stage is a maximum value flow, not only in the equality subnetwork at that stage but in the entire original bipartite network. This case can only occur if some of the x_{ij} were required to be 0 in the original problem. Since this flow leaves the demand at some markets unfulfilled, it implies that there is no feasible solution to the problem.

Array 3.5 Summary of Position at Occurrence of Nonbreakthrough. $(\tilde{u}, \tilde{v}, \tilde{\bar{c}} = (\tilde{\bar{c}}_{ij} = c_{ij} - \tilde{u}_i - \tilde{v}_j)$, are the dual solution and reduced cost matrix before the change. $(\hat{u}, \hat{v}, \hat{\bar{c}} = (\hat{\bar{c}}_{ij} = c_{ij} - \hat{u}_i - \hat{v}_j)$ are the corresponding things after the change. $\tilde{x} = (\tilde{x}_{ij})$ is the present flow.

	Block of labeled cols.	Block of unlabeled cols.	Supply position	Dual change
Block of labeled rows	$\hat{\bar{c}}_{ij} = \tilde{\bar{c}}_{ij}$ here, so admissibility pattern remains unchanged	No admissible cells here (a col. would be labeled otherwise). So $\tilde{\bar{c}}_{ij} > 0$ here. $\delta =$ Min$\{\tilde{\bar{c}}_{ij}$: (i,j) here$\} > 0$. $\hat{\bar{c}}_{ij} = \tilde{\bar{c}}_{ij} - \delta$ here. New admissible cells created here, and their cols. get labeled next.	Material available at some rows here.	$\hat{u}_i = \tilde{u}_i + \delta$ here.
Block of unlabeled rows	$\tilde{x}_{ij} = 0$ here (otherwise a row could be labeled). $\hat{\bar{c}}_{ij} = \tilde{\bar{c}}_{ij} + \delta$ here. All cells here become inadmissible next.	$\hat{\bar{c}}_{ij} = \tilde{\bar{c}}_{ij}$ here, so admissibility pattern remains unchanged.	$\sum_j \tilde{x}_{ij} = a_i$ here; otherwise a row could be labeled.	$\hat{u}_i = \tilde{u}_i$ here.
Requirement position	$\sum_i \tilde{x}_{ij} = b_j$ here; otherwise it would be a breakthrough.	Some cols. with unfulfilled requirements.		
Dual change	$\hat{v}_j = \tilde{v}_j - \delta$ here	$\hat{v}_j = \tilde{v}_j$ here.		

For arbitrary c and arbitrary (i.e., not necessarily rational) positive a_i, b_j sat- isfying $\sum a_i = \sum b_j$, the method terminates in a finite number of steps if the list is maintained as a queue and the node for scanning in Substep 2 of Step 1 is selected by the FIFO (first in first out) rule. To see this, we notice from the facts in Array 3.5 that when labeling is resumed after each occurrence of Step 3, at least one new column gets labeled. So, in this algorithm, after at most n consecutive occurrences of Step 3, Step 2 must occur, and then the total flow $\sum \sum x_{ij}$ strictly increases.

Each equality subnetwork appearing in the algorithm consists of all the nodes (source nodes with their availabilities, sink nodes with their requirements) and equality arcs corresponding to the subset of equality cells wrt the dual feasible solution at that stage. We can associate any equality subnetwork with the subset of cells corresponding to the equality arcs in it. Denote by $\vartheta(\mathbf{E})$ the value of the maximum value flow in the equality subnetwork associated with the subset of cells $\mathbf{E}$.

The equality subnetwork changes in the algorithm after each occurrence of Step 3. The first time that Step 3 occurs after an equality subnetwork associated with a subset $\mathbf{E}$ of cells appears for the first time in the algorithm, the total flow $\sum \sum x_{ij}$ becomes equal to $\vartheta(\mathbf{E})$, and this happens after at most $O(m(p+n))$ consecutive occurrences of Steps 1 and 2, by the results in Section 2.3.3. If the algorithm did not terminate then, Step 3 will occur, and it could repeat at most n times consecutively, followed by another occurrence of Steps 1 and 2, at which time the total flow strictly exceeds $\vartheta(\mathbf{E})$.

So, after the equality subnetwork defined by the subset of cells $\mathbf{E}$ first appears in the algorithm, there is a consecutive run of at most $O(m(p+n))$ Steps 1 and 2 ; followed by a consecutive run of at most n Steps 3, and then another occurrence of Steps 1 and 2. After these are over, the equality subnetwork defined by the subset of cells $\mathbf{E}$ can never reappear in the algorithm. Since there are only a finite number of subsets of cells, this implies that the algorithm is a finite algorithm.

This primal-dual method for the transportation problem is practically efficient and is useful for doing sensitivity analysis when the a_i and b_j change. Experience indicates that in solving large problems, computer implementations of the primal algorithm using tree labels,

discussed in Chapter 5, are superior to those of the primal-dual method discussed here. Newer variants of the primal-dual method are discussed in Chapter 5, they are competitive with other algorithms for solving large scale minimum cost flow problems.

To study the worst case computational complexity of this algorithm, we will assume that all the a_i and b_j are positive integers and let $\sum_i a_i = \sum_j b_j = \gamma$. In this case, every breakthrough leads to an increase in the total flow value, $\sum_i \sum_j x_{ij}$, by at least 1. When tree growth is resumed after each dual solution change, at least one new column gets labeled. Hence between two consecutive breakthroughs, there are at most n nonbreakthroughs. It can be verified that the overall computational effort in the primal-dual method in this case is bounded above by $O(\gamma(n+p)^{\mathbf{2}})$. This grows exponentially with the size of the problem, since γ grows exponentially with the number of digits needed to store the a_i, b_j. However there is a **scaling** *or* **digit-by-digit implementation** of the primal-dual method which is polynomially bounded, which we discuss next.

Polynomially Bounded Scaling Implementation

Consider the case where the a_i, b_j are all rational numbers. By selecting the unit for measuring the commodity appropriately, the problem can then be modified into one in which all the a_i, b_j are positive integers. We assume that this has been done. Let $\sum_i a_i = \sum_j b_j = \gamma$. No assumptions are made about c_{ij}.

Let q be the smallest positive integer such that a_i and $b_j \leqq 2^{\mathbf{q}}$ for all i, j (i.e., each a_i, b_j have at most q digits in its binary expansion, and q is the smallest integer with this property). The scaling implementation deals with a sequence of $q+1$ problems called subproblems all of which are transportation problems with the same cost matrix (and hence the same set of nodes and arcs) as (3.7), but with scaled down availabilities and requirements which approximate those of (3.7) to successively more digits of precision. Initially availabilities and requirements, and hence flow augmentations are on a coarser scale than in the original problem, but by the end of the sequence all the data converts to the original data, and the terminal solution, if one is obtained, will be an optimum

solution of the original problem. The final solution of each approximate problem leads to a good initial flow for the next approximate problem in the sequence. The worst case computational complexity of the scaling implementation is proportional to the number of digits in the binary encoding of the a_i, b_j.

We consider the general problem in which some of the x_{ij} may be required to be 0. For $i = 1$ to p, let $\mathbf{\Gamma}_i = \{j\colon\ x_{ij}$ can have a positive value in the solution$\}$; and for $j = 1$ to n, let $\mathbf{\Omega}_j = \{i\colon\ x_{ij}$ can have a positive value in the solution$\}$. So, source i is allowed to ship only to sinks in $\mathbf{\Gamma}_i$ for all i, and sink j can receive shipments only from sources in $\mathbf{\Omega}_j$ for all j. Let $\mathbf{F} = \{(i,j) : x_{ij}$ can have a positive value in the solution$\} = \cup_{i=1}^{p}\{(i,j) : j \in \mathbf{\Gamma}_i\} = \cup_{j=1}^{n}\{(i,j) : i \in \mathbf{\Omega}_j\}$. So, the cost coefficients c_{ij} are only defined for $(i,j) \in \mathbf{F}$ in the problem, for $(i,j) \notin \mathbf{F}$ we treat c_{ij} to be $+\infty$. Hence the original problem to be solved is

$$\begin{aligned} \text{Minimize} \sum_{(i,j)\in\mathbf{F}} c_{ij}\ \ x_{ij} & \\ \text{Subject to} \sum_{j\in\mathbf{\Gamma}_i} x_{ij} &= a_i, i = 1 \text{ to } p \\ \sum_{i\in\mathbf{\Omega}_j} x_{ij} &= b_j, j = 1 \text{ to } n \\ x_{ij} &\geqq 0, \text{ for } (i,j) \in \mathbf{F} \end{aligned} \tag{3.11}$$

Let $d_i, g_j, i = 1$ to p, $j = 1$ to n, be non-negative integers. In the scaling implementation, the subproblems are all transportation models in the form (3.12), where Δ is a specified positive integer. In the algorithm, d_i will be of the form $\lfloor \frac{a_i}{2^{\mathbf{s}}} \rfloor$ and g_j will be of the form $\lfloor \frac{b_j}{2^{\mathbf{s}}} \rfloor$, where $\mathbf{s}$ ranges from q to 0 in stages. When some of the x_{ij} are required to be zero even if the original problem (3.11) is feasible, the subproblems in some stages may be infeasible for some Δ. In any feasible solution of (3.12), Δ is known as the **total flow** in that solution. It is the total amount of material reaching the sinks from the sources in that solution.

$$\text{Minimize} \sum_{(i,j)\in\mathbf{F}} c_{ij}\ \ x_{ij}$$

$$\begin{aligned} \text{Subject to } \sum_{j \in \mathbf{\Gamma}_i} x_{ij} &\leqq d_i, i = 1 \text{ to } p \\ \sum_{i \in \mathbf{\Omega}_j} x_{ij} &\leqq g_j, j = 1 \text{ to } n \\ \sum_i \sum_j x_{ij} &= \Delta \\ x_{ij} &\geqq 0, \text{ for } (i,j) \in \mathbf{F} \end{aligned} \tag{3.12}$$

The dual of the original problem (3.11) is

$$\begin{aligned} \text{Maximize } \sum a_i u_i &+ \sum b_j v_j \\ \text{Subject to } u_i + v_j &\leqq c_{ij} \text{ for all } (i,j) \in \mathbf{F} \end{aligned} \tag{3.13}$$

Denoting the dual variables corresponding to the constraints in (3.12) by π_i, μ_j, δ in that order, the dual of (3.12) is

$$\begin{aligned} \text{Maximize } \delta\Delta - \sum d_i \pi_i &- \sum g_j \mu j \\ \text{Subject to } -\pi_i - \mu_j + \delta &\leqq c_{ij}, \text{ for all } (i,j) \in \mathbf{F} \\ \pi_i, \mu_j &\geqq 0, \text{ for all } i, j. \end{aligned} \tag{3.14}$$

The complementary slackness optimality conditions in the primal dual pair (3.12), (3.14) are

$$(c_{ij} + \pi_i + \mu_j - \delta) x_{ij} = 0, \text{ for all } (i,j) \in \mathbf{F} \tag{3.15}$$

$$\pi_i (d_i - \sum_{j=1}^{n} x_{ij}) = 0, i = 1 \text{ to } p \tag{3.16}$$

$$\mu_j (g_j - \sum_{i=1}^{p} x_{ij}) = 0, j = 1 \text{ to } n \tag{3.17}$$

We call the pair $(x, (\pi, \mu, \delta))$ feasible to (3.12), (3.14) an **extreme pair** if they together satisfy (3.15), whether they satisfy (3.16), (3.17) or not. An extreme pair for (3.12),(3.14) is said to be a **maximum extreme pair** if the total flow $\sum(x_{ij} : \text{over } (i,j) \in \mathbf{F})$ is the maximum

value that can be attained with the supply at source (row) node i limited to at most d_i, and the amount that can be shipped to sink (column) node j limited to at most g_j, $i = 1$ to p, $j = 1$ to n.

If $(\overline{x}, (\overline{\pi}, \overline{\mu}, \overline{\delta}))$ is a maximum extreme pair when $d_i = a_i, g_j = b_j$ for all i, j, and the total flow in $\overline{x}$ is $\gamma = \sum a_i$, then (3.16), (3.17) hold automatically in this pair. In this case, let $\overline{u}_i = \overline{\delta} - \overline{\pi}_i, \overline{v}_j = -\overline{\mu}_j$ for all i, j, then $(\overline{x}, (\overline{u}, \overline{v}))$ is an optimum pair for the original problem. This fact is used in this implementation.

For $r = 0$ to q, define

$$d_i^r = \lfloor \frac{a_i}{2^{\mathbf{q-r}}} \rfloor, g_j^r = \lfloor \frac{b_j}{2^{\mathbf{q-r}}} \rfloor, i = 1 \text{ to } p, j = 1 \text{ to } n.$$

The scaling implementation solves a sequence of $(q + 1)$ subproblems. For $r = 0$ to q, the rth subproblem is (3.12) with $d_i = d_i^r, g_j = g_j^r$ for all i, j. Each subproblem is solved by the primal dual algorithm, walking along extreme pairs only, until a maximum extreme pair is obtained for it. This will be recognized in the algorithm when, either the capacity at all the source (row) nodes is used up, or all the capacity at the sink (column) nodes is used up, or if $\delta = \infty$ in a dual solution change step. It then moves to the next subproblem, beginning with an initial extreme pair for it constructed from the last pair for the present subproblem. In the qth (i.e., final) subproblem, we have $d_i = a_i, g_j = b_j$ for all i, j, and since $\sum a_i = \sum b_j$, a maximum extreme pair for it leads to a solution for the original problem (3.11) if it is feasible, as shown later.

For $r < q$ we may not have $\sum_i d_i^r = \sum_j g_j^r$, but we apply the primal dual algorithm to find a maximum extreme pair for the rth subproblem. While solving this problem the algorithm maintains the pair $(x, (u, v))$ where x satisfies (3.12) with $d_i = d_i^r, g_j = g_j^r$ for all i, j; u, v satisfy (3.13) always, and these vectors together satisfy (3.9). There is a vector (π, μ, δ) feasible to (3.14) and satisfying $-\pi_i - \mu_j + \delta = u_i + v_j$ for all i, j; that can be obtained from (u, v) by the formulas $\pi_i = \nu - u_i, \mu_j = \nu - v_j, \delta = 2\nu$ for all i, j, where $\nu = \max.\{u_1, \ldots, u_p; v_1, \ldots, v_n\}$, and the pair $(x, (\pi, \mu, \delta))$ will then be an extreme pair for (3.12), (3.14).

Let G denote the bipartite network for the original problem (3.11) as in Figure 3.5, in which the arc joining source node i to market node

j exists iff $(i,j) \in \mathbf{F}$. For $r = 0$ to q, let G_r denote the same network as G, but with the capacities a_i replaced by d_i^r, and capacities b_j replaced by g_j^r, for all i, j.

Subproblem 0 begins with $x^0 = 0, u^0 = (u_i^0 = \text{min.}\{c_{ij} : j = 1 \text{ to } n\})$, $v^0 = (v_j^0 = \text{min.}\{c_{ij} - u_i^0 : i = 1 \text{ to } p\})$. The rth subproblem is terminated when flow value in the network G_r from the supersource to the supersink (which is the total flow in the solution x at that stage) reaches the maximum value possible in G_r recognized as described above. If the original problem is on the complete bipartite network (i.e., all variables x_{ij} are allowed to take positive values), this will happen when either the supply d_i^r at each source i is used up or the demand g_j^r at each market j is met. If some variables are required to be zero in the original problem, another signal for the termination of the rth subproblem is δ becoming ∞ in a dual solution change step. Let $\tilde{x}^r, (\tilde{u}^r, \tilde{v}^r)$ be the pair at the termination of subproblem r. The total flow in $\tilde{x}^r$ is

$$\sum_i \sum_j \tilde{x}^r = \gamma_r \tag{3.18}$$

Define, for $r = 0$ to q

$$\Delta_r = \text{ min. } \{\sum_i d_i^r, \sum_j g_j^r\} \tag{3.19}$$

So, $\Delta_q = \gamma$, and $\gamma_r \leq \Delta_r$ for all r. For $r = 0$ to $q-1$, if $\gamma_r < \Delta_r - (n+p)$, terminate the algorithm. In this case the original problem (3.11) has no feasible solution (see Lemma 3.2 given below). Otherwise, initiate subproblem $r+1$ with the pair $x^{r+1} = 2\tilde{x}^r, (u^{r+1} = \tilde{u}^r, v^{r+1} = \tilde{v}^r)$, this pair satisfies (3.13), (3.9) since $\tilde{x}^r, (\tilde{u}^r, \tilde{v}^r)$ do. Thus all the pairs obtained during the implementation will be extreme pairs. At the end of subproblem q, we have the maximum extreme pair $\tilde{x}^q, (\tilde{u}^q, \tilde{v}^q)$ in G, with total flow $= \gamma_q$. If $\gamma_q = \gamma$, $\tilde{x}^q, (\tilde{u}^q, \tilde{v}^q)$ together satisfy (3.11), (3.13), (3.9), so they form an optimum primal dual pair for the original problem (3.11). If $\gamma_q < \gamma$, the original problem (3.11) is infeasible.

Flow changes occur at most Δ_0 times in subproblem 0. If the algorithm is continued after subproblem r, $2\tilde{x}^r$ is the initial primal vector for solving the $(r+1)$th subproblem. By Lemma 3.2 given below, there will be at most $\Delta_{r+1} - 2\Delta_r + 2(n+p)$ flow changes while solving the

$(r+1)$th subproblem. So, the total number of flow augmentations during the entire scaling implementation is at most $\Delta_0+\sum_{r=0}^{q-1}(\Delta_{r+1}-2\Delta_r)$ $+ 2q(n+p)$.

i	a	d^0	d^1	d^2	d^3	d^4	d^5	d^6
1	38	0	1	2	4	9	19	38
2	41	0	1	2	5	10	20	41
3	23	0	0	1	2	5	11	23
4	15	0	0	0	1	3	7	15
Total	117	0	2	5	12	27	57	117

j	1	2	3	4	Total
b	25	29	30	33	117
g^0	0	0	0	0	0
g^1	0	0	0	1	1
g^2	1	1	1	2	5
g^3	3	3	3	4	13
g^4	6	7	7	8	28
g^5	12	14	15	16	57
g^6	25	29	30	33	117

$\triangle_r$ = 0,1,5,12,27,57,117 respectively for $r = 0$ to 6.

As an example we consider a problem with $p = n = 4$. The cost matrix is not shown, but the vectors a, b, and d^r, g^r are given for all r in the table above. $\gamma = \sum a_i = \sum b_j = 117$. Max.$\{a_i, b_j : \text{all } i, j\} = 41$, and the smallest integer q satisfying the property that all $a_i, b_j \leqq 2^{\mathbf{q}}$ is 6. So, we have to solve seven subproblems, these correspond to $r = 0$ to 6.

In this example $\Delta^0 + \sum_{r=0}^{q-1}(\Delta_{r+1} - 2\Delta_r) = 15$. So, to solve this problem, the scaling implementation needs at most $15 + 96 = 111$ flow augmentations, this compares with the maximum of 117 that may be needed in the direct implementation of the primal dual method.

LEMMA 3.1 $0 \leqq \Delta_0 \leqq$ *max.*$\{p, n\}$*;* $0 \leqq \Delta_{r+1} - 2\Delta_r \leqq$ *max.*$\{\ p, n\ \}$*, for all* $r \geqq 0$*.*

Proof From the definition of q, all d_i^0, g_j^0 are 0 or 1. So, $0 \leqq \Delta_0 \leqq$ max.$\{p, n\}$.

Let ξ, η be any positive integers. We have

$$2\lfloor\frac{\eta}{2^\xi}\rfloor \leqq \lfloor\frac{\eta}{2^{\xi-1}}\rfloor \leqq 2\lfloor\frac{\eta}{2^\xi}\rfloor + 1$$

Applying this we conclude that

$$\begin{aligned} 0 \leqq \sum_{i=1}^{p} \lfloor \frac{a_i}{2^{\mathbf{q}-r-1}} \rfloor - 2\sum_{i=1}^{p} \lfloor \frac{a_i}{2^{\mathbf{q-r}}} \rfloor &\leqq p \\ 0 \leqq \sum_{j=1}^{n} \lfloor \frac{b_j}{2^{\mathbf{q}-r-1}} \rfloor - 2\sum_{j=1}^{n} \lfloor \frac{b_j}{2^{\mathbf{q}-r}} \rfloor &\leqq n \end{aligned}$$

These inequalities, and the definitions of Δ_r, Δ_{r+1}, imply this lemma. ∎

LEMMA 3.2 *If the original problem (3.11) has a feasible solution, then* $\gamma_r \geqq \Delta_r - (p+n)$, *for all* r.

Proof Let $\hat{x} = (\hat{x}_{ij})$ be a feasible solution for (3.11). Then, for all i, j

$$\begin{aligned} \sum_{j \in \mathbf{\Gamma}_i} (\hat{x}_{ij}/2^{\mathbf{r}}) &= a_i/2^{\mathbf{q}-r} \\ \sum_{i \in \mathbf{\Omega}_j} (\hat{x}_{ij}/2^{\mathbf{r}}) &= b_j/2^{\mathbf{q}-r} \end{aligned}$$

So, if the amount of material that can leave source (row) node i is $a_i/2^{\mathbf{q}-r}$, and that reaching market (column) node j is $b_j/2^{\mathbf{q}-r}$, for all i, j; then all the arcs incident at the supersource and the supersink will be exactly saturated in a maximum value flow ($\hat{x}/2^{\mathbf{r}}$ is such a flow). However, the maximum amount of material that can leave source (row) node i is limited to at most $d_i^r = \lfloor a_i/2^{\mathbf{q}-r} \rfloor \geqq (a_i/2^{\mathbf{q}-r}) - 1$, and that reaching market (column) node j is limited to at most $g_j^r = \lfloor b_j/2^{\mathbf{q}-r} \rfloor \geqq (b_j/2^{\mathbf{q}-r}) - 1$ in G_r. So the maximum flow value in G_r is $\geqq \Delta_r - (p+n)$, i.e., $\gamma_r \geqq \Delta_r - (p+n)$. ∎

THEOREM 3.2 *Assuming that all the* a_i, b_j *are positive integers, the overall computational effort for solving (3.11) by the scaling implementation of the primal dual algorithm is at most* $O(L(n+p)^{\mathbf{2}}(max.\{p,n\}))$, *where* L *is the sum of the binary digits in all the* a_i *and* b_j.

Proof q defined earlier is $< L$. Lemmas 3.1 and 3.2 imply that the total number of flow changes in any subproblem is at most 4 max.$\{ p, n \}$. So, the total number of flow augmentations in the scaling implementation is at most $4(q+1)(\text{max.}\{ p, n \}) \leqq 4L(\text{max.}\{ p, n \})$. We have already seen that the computational effort between two consecutive occurrences of flow augmentation in any subproblem is at most $O(p+n)^2$. Hence the result follows. ∎

So, the scaling implementation of the primal dual algorithm is a polynomially bounded algorithm for solving the transportation problem (3.7) with some specified subset of variables x_{ij} set equal to 0 if desired, in which the overall computational effort is bounded above by a low degree polynomial in the size of the problem.

Comment 3.2 This scaling technique was introduced by Edmonds and Karp [1972 of Chapter 2]. It led to the first polynomial time algorithm for minimum cost flow problems.

Exercises

3.18 Complete the solution of the problem in Example 3.3 by the primal-dual algorithm

3.19 Develope a primal-dual method to solve the following capacitated transportation problem.

$$\begin{aligned} \text{Minimize } z(x) &= \sum_{i=1}^{p}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{Subject to } \sum_{j=1}^{n} x_{ij} &\leqq a_i, i = 1 \text{ to } p \\ \sum_{i=1}^{p} x_{ij} &= b_j, j = 1 \text{ to } n \\ 0 \leqq x_{ij} &\leqq k_{ij}, \text{ for all } i, j \end{aligned}$$

3.20 Whenever a nonbreakthrough occurs in the primal-dual algorithm for the balanced transportation problem (3.7), prove that the quantity $\sum (a_i : \text{over labeled rows } i) - \sum(b_j : \text{over labeled cols. } j) > 0$. Using this prove that the dual objective function $\sum_i a_i u_i + \sum_j b_j v_j$ strictly increases whenever the dual solution changes in this algorithm. Also prove the following using the duality theorem of LP: While solving (3.7) with some variables x_{ij} constrained to be 0, by the primal-dual algorithm, if δ turns out to be $+\infty$ in some dual solution change step, the problem is infeasible.

3.3 Transformation of Single Commodity Minimum Cost Flow Problem into Sparse Balanced Transportation Problem

The balanced transportation problem is a special case of the single commodity minimum cost flow problem on a bipartite network. We will now show that every single commodity minimum cost flow problem, even on a nonbipartite network, can be transformed into an uncapacitated balanced transportation problem which is sparse.

Consider the single commodity minimum cost flow problem on the directed network $G = (\mathcal{N}, \mathcal{A}, 0, k, c, \breve{s}, \breve{t}, \bar{v})$ for shipping $\bar{v}$ units from $\breve{s}$ to $\breve{t}$ at minimum cost. Let $|\mathcal{N}| = n, |\mathcal{A}| = m$. Construct a bipartite network $H = (\mathcal{N}_1, \mathcal{N}_2; \mathbf{A})$ as follows. For each $(i, j) \in \mathcal{A}$ put a node corresponding to it in $\mathcal{N}_1$ which we conveniently denote by the symbol "ij." So, $|\mathcal{N}_1| = m$. Make $\mathcal{N}_2 = \mathcal{N}$. $\mathcal{N}_1, \mathcal{N}_2$ are respectively the sets of source and sink nodes in H. Define $\mathbf{A} = \{(ij, i), (ij, j) : \text{ for each } (i, j) \in \mathcal{A}\}$, so $|\mathbf{A}| = 2m$. The lower bounds and capacities for arcs in $\mathbf{A}$ are 0, $+\infty$ respectively. For each $(i, j) \in \mathcal{A}$, in H the unit cost coefficient on the arc (ij, i) is 0, and that on the arc (ij, j) is c_{ij}; and the availability of material at the source node ij in $\mathcal{N}_1$ is k_{ij}. For each $i \in \mathcal{N}$, the requirement at the sink node $i \in \mathcal{N}_2$ in H is b_i where

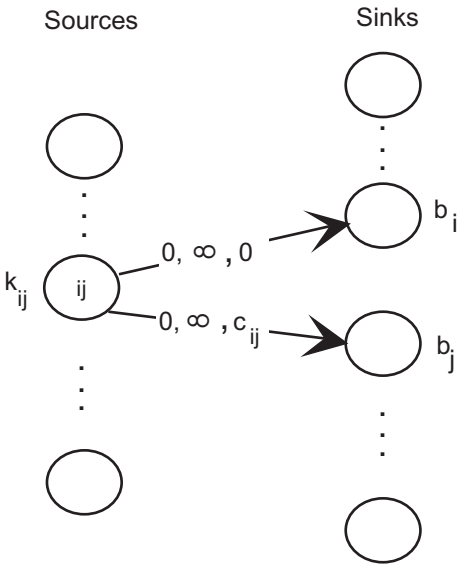

Figure 3.5: Bipartite network H. Data on arcs is lower bound, capacity, unit cost coefficient, in that order. Data by the side of the nodes is availability (for sources), requirement (for sinks).

$$b_i = \begin{cases} k(i,\mathcal{N}) - \bar{v} & \text{if } i = \breve{s} \\ k(i,\mathcal{N}) + \bar{v} & \text{if } i = \breve{t} \\ k(i,\mathcal{N}) & \text{if } i \neq \breve{s} \text{ or } \breve{t} \end{cases}$$

Unless $k(\breve{s},\mathcal{N}) \geqq \bar{v}$, the flow problem in G is infeasible. So, we assume that this condition holds. This implies that all the availabilities at the sources, and the requirements at the sinks, in H, are $\geqq 0$. In H, each source node is joined to exactly two sink nodes. See Figure 3.5. The sum of all the availabilities at the source nodes in H, as well as the sum of all the requirements at the sink nodes, are both equal to $k(\mathcal{N},\mathcal{N})$. Thus the minimum cost flow problem in H satisfying all the availability, requirement constraints at the nodes, is a sparse uncapacitated balanced transportation problem.

We will now prove the equivalence of the minimum cost flow problems in G and H. Let $f = (f_{ij})$ be a feasible flow vector in G. Define the

corresponding flow vector in H to be x where for each $(i,j) \in \mathcal{A}, x_{ij,j} = f_{ij}$, and $x_{ij,i} = k_{ij} - f_{ij}$. Then it can be verified that x is a feasible flow vector in H, having the same cost as f in G. Conversely, if x is a feasible flow vector in H, define the flow vector $f = (f_{ij})$ in G by: for each $(i,j) \in \mathcal{A}$, $f_{ij} = x_{ij,j}$. It can be verified that f is a feasible flow vector in G which has the same cost as x in H. So, the minimum cost flow problem in G is equivalent to the minimum cost flow problem in H, which is a sparse uncapacitated balanced transportation problem. Using this transformation, and the scaling implementation to solve the resulting transportation problem, we get a polynomially bounded algorithm for the minimum cost flow problem in G. Primal-dual algorithms that solve the minimum cost flow problem in G directly will be discussed later in Chapter 5.

3.4 Dual Simplex Signature Methods for the Assignment Problem

Consider the LP

$$\begin{array}{rcl} \text{Minimize } c_B x_B & + & c_D x_D \\ \text{Subject to } Bx_B & + & Dx_D = b \\ x_B & , & x_D \geqq 0 \end{array}$$

in which the coefficient matrix $A = (B \vdots D)$ is of order $m \times n$ and rank m. The LP is written with the variables partitioned into basic, nonbasic parts, x_B, x_D. The primal basic solution associated with the basic vector x_B is $\bar{x} = (\bar{x}_B = B^{-1}b, \bar{x}_D = 0)$. It is **primal feasible** if $\bar{x} \geqq 0$. The dual basic solution associated with this basic vector is $\bar{\pi} = c_B B^{-1}$. This basic vector is **dual feasible** if the associated reduced cost vector $\bar{c} = (\bar{c}_B, \bar{c}_D) = (0, c_D - \bar{\pi}D) \geqq 0$. The dual simplex algorithm for solving this LP is always initiated with a dual feasible basic vector x_B. If x_B is also primal feasible, it is an optimum basic vector, and the algorithm terminates. Otherwise, the algorithm checks to see if a condition for primal infeasibility is satisfied, and if so, it again terminates. If neither

of these two events has occurred, it selects a primal variable x_j whose value $\bar{x}_j$ in the primal basic solution $\bar{x}$ associated with x_B satisfies $\bar{x}_j < 0$, as the **dropping variable** from the present basic vector x_B, for a dual simplex pivot step. The nonbasic variable to replace it, called the **entering variable** is selected by the dual simplex minimum ratio test, whose function is to guarantee that the next basic vector will also be dual feasible. Then the whole process is repeated with the new basic vector. In the dual simplex algorithm, the dual objective function πb is monotone nondecreasing. See Murty [1983 of Chapter 1] for a detailed description of the dual simplex algorithm. In this section we will discuss a new class of algorithms for the assignment problem called **signature methods**, which are related to the dual simplex algorithm in spirit.

We consider the assignment problem (3.1) of order n associated with the cost matrix $c = (c_{ij})$, a minimum cost flow problem on the bipartite network $G = (\mathcal{N}_R, \mathcal{N}_C, \mathcal{A})$, where $\mathcal{N}_R = \{R_1, \ldots, R_n\}$ = set of row nodes, $\mathcal{N}_C = \{C_1, \ldots, C_n\}$ = set of column nodes, and $\mathcal{A} = \{ (R_i, C_j)$: i, j s.t. an allocation is allowed in cell $(i, j)\}$. The arc (R_i, C_j) in G is associated with the variable x_{ij} in the problem. The dual variables are u_i, v_j associated with the nodes R_i, C_j respectively, $i, j = 1$ to n. Each basic vector for (3.1) corresponds to a spanning tree in G and vice versa, in-tree arcs are associated with basic variables, and out-of-tree arcs with nonbasic variables. A spanning tree $\mathbb{T}$ in G has exactly $2n - 1$ arcs, and the dual basic solution corresponding to it, denoted by $u(\mathbb{T}) = (u_i(\mathbb{T})), v(\mathbb{T}) = (v_j(\mathbb{T}))$, can be obtained by arbitrarily fixing the value of one dual variable (e.g. $u_1(\mathbb{T}) = 0$) and then solving the equations

$$u_p(\mathbb{T}) + v_q(\mathbb{T}) = c_{pq}, \text{ for each in-tree arc } (R_p, C_q)$$

by back substitution. The matrix $\bar{c}(\mathbb{T}) = (\bar{c}_{ij}(\mathbb{T}))$, where $\bar{c}_{ij}(\mathbb{T}) = c_{ij} - u_i(\mathbb{T}) - v_j(\mathbb{T})$ is the matrix of reduced or relative cost coefficients wrt $\mathbb{T}$. $\mathbb{T}$ is dual feasible if $\bar{c} \geqq 0$, dual infeasible otherwise.

The primal basic solution of (3.1) corresponding to $\mathbb{T}$, $x(\mathbb{T}) = (x_{ij}(\mathbb{T}))$ is obtained by setting $x_{ij}(\mathbb{T}) = 0$ whenever (R_i, C_j) is an out-of-tree arc, in the system of equality constraints in (3.1), and then solving the remaining system by back substitution. $x(\mathbb{T})$ is integer,

but it may not be ≥ 0. $\mathbb{T}$ is primal feasible if $x(\mathbb{T}) \geq 0$ (in this case $x(\mathbb{T})$ will be an assignment), primal infeasible otherwise.

We denote by $\mathcal{P}(p, q, \mathbb{T})$, the unique path between the nodes p, q in $\mathbb{T}$.

The **row signature vector (column signature vector)** of a spanning tree in G is the vector of its row node degrees (column node degrees). If $(d_1, \ldots, d_n)$ is the row or column signature vector of a spanning tree in G, clearly, $d_i \geq 1$ for all $i = 1$ to n, and $\sum_{i=1}^{n} d_i = 2n - 1$. Hence, in every signature vector there is at least one entry equal to 1. It corresponds to a terminal node. For example, the row, column signature vectors of the spanning tree in Figure 3.9 are (2, 1, 2, 2, 2, 2), (1, 3, 4, 1, 1, 1) respectively. Both these vectors have at least one entry of 1.

THEOREM 3.3 *A spanning tree $\mathbb{T}$ in* G *which contains exactly one entry of 1 in either its row or column signature vectors is a primal feasible spanning tree.*

Proof Let $d = (d_1, \ldots, d_n)$ be the row signature vector of $\mathbb{T}$, and suppose it contains a unique 1 entry. A similar proof holds if the column signature vector contains a single 1 entry.

Let i^* be the unique number such that $d_{i^*} = 1$. Since $d_i \geq 1$ for all i and $\sum_{i=1}^{n} d_i = 2n - 1$, the hypothesis implies that $d_i = 2$ for all $i \neq i^*$. For each $i \neq i^*$, the path $\mathcal{P}(R_{i^*}, R_i, \mathbb{T})$ contains exactly one edge incident at R_i, and since $d_i = 2$, there must be exactly one edge in $\mathbb{T}$ incident at R_i which is not on the path $\mathcal{P}(R_{i^*}, R_i, \mathbb{T})$. Let $\sigma_{i^*} = j^*$, where (R_{i^*}, C_{j^*}) is the unique in-tree arc incident at R_{i^*}, and for $i \neq i^*$, let σ_i be j, where j is such that (R_i, C_j) is the unique in-tree arc incident at R_i not on $\mathcal{P}(R_{i^*}, R_i, \mathbb{T})$. Clearly, $\sigma_i \neq \sigma_h$ for any $i \neq h$, as otherwise there will be a cycle in $\mathbb{T}$, a contradiction. So, $\{(i, \sigma_i) : i = 1 \text{ to } n\}$ is an assignment, and since all these cells correspond to in-tree edges, this assignment is the basic solution of (3.1) corresponding to $\mathbb{T}$, hence $\mathbb{T}$ is a primal feasible tree. ∎

As an example consider the spanning tree in Figure 3.9 with its row signature vector (2, 1, 2, 2, 2, 2). R_2 is the unique terminal row node. The primal basic solution corresponding to this tree is the assignment

{(1, 1), (2, 3), (3, 2), (4, 5), (5, 4), (6, 6)}, which corresponds to the set of solid lines in Figure 3.9.

Signature methods are always initiated with a dual feasible spanning tree in G and dual feasibility is maintained throughout. They go through a sequence of pivot steps; each pivot step moves from a tree to an adjacent tree that differs from it in a single arc. The dropping arc from the present tree is always selected by a special dropping arc selection rule described in the algorithm. The entering arc to replace the dropping arc is determined so as to maintain dual feasibility. Termination occurs when either primal infeasibility is established, or when a spanning tree with exactly one terminal row node, or one terminal column node is obtained. In the latter case the primal basic solution corresponding to the final tree is an optimum assignment. The method tries to reduce the number of terminal row nodes (or terminal column nodes) to one. This number is monotone nonincreasing during the method.

A **stage** in a signature method begins with a dual feasible spanning tree in G with more than one terminal row node (or column node, if the method works with column signature vectors) and is completed when this number decreases by 1. Each stage may consist of several pivot steps. Since G is bipartite and $\mathbb{T}$ has $2n-1$ arcs, the total number of terminal row nodes, or column nodes in any spanning tree in G, is at most $n-1$. So, the algorithm has at most $n-2$ stages.

The dropping arc choice is defined by the method, but once the dropping arc is selected, the choice of the entering arc to replace it is carried out by the same procedure in all the signature methods. We describe this next.

The Entering Arc Choice Rule

Let $\mathbb{T}$ be the present dual feasible spanning tree in G, and let (w, l) denote the in-tree arc that has been selected as the dropping arc, w, l are the row and column nodes on it. Deletion of the arc (w, l) from $\mathbb{T}$ leaves two distinct subtrees, a $\mathbb{T}^w$ containing node w, and a $\mathbb{T}^l$ containing node l. If $\mathbb{T}$ is drawn as a rooted tree with w as the root node, the family of node l is exactly the set of nodes in $\mathbb{T}^l$, and the set of nodes in $\mathbb{T}^w$ is the complement of this set. Let $\mathbf{X} = \mathbf{R}^w$, $\bar{\mathbf{X}} = \mathbf{R}^l$ ($\mathbf{Y} =$

$\mathbf{C}^w$, $\bar{\mathbf{Y}} = \mathbf{C}^l$) be the set of row (column) nodes in $\mathbb{T}^w, \mathbb{T}^l$ respectively. ($\mathbf{X}$, $\bar{\mathbf{X}}$), ($\mathbf{Y}$, $\bar{\mathbf{Y}}$), are the partitions of row, column nodes in this pivot step. They are uniquely determined by $\mathbb{T}$ and (w, l).

Since (w, l) is an in-tree arc, $\tilde{\bar{c}}_{wl}$, its present relative cost coefficient, is 0, but it becomes nonnegative in the next tree. Suppose its new value is δ. Let $\tilde{u}_i, \tilde{v}_j, \tilde{\bar{c}}_{ij}$ denote the present quantities in the tree $\mathbb{T}$. It can be verified that the unique solution of the system: $u_1 = 0, c_{pq} - u_p - v_q = 0$ for each arc $(R_p, C_q) \in \mathbb{T}$ except the arc (w, l), and $= \delta$ for arc (w, l), is $(\hat{u}, \hat{v})$ marked in the following Array 3.6, and $\hat{\bar{c}}_{ij}$ are the reduced cost coefficients wrt it.

Array 3.6

Block of rows ↓	Block of cols. $\mathbf{Y} = \mathbf{C}^w$	$\bar{\mathbf{Y}} = \mathbf{C}^l$	New dual solution $\hat{u}$
$\mathbf{X} = \mathbf{R}^w$	$\hat{\bar{c}}_{ij} = \tilde{\bar{c}}_{ij}$ in this block	$\hat{\bar{c}}_{ij} = \tilde{\bar{c}}_{ij} + \delta$ in this block	$\hat{u}_i = \tilde{u}_i$ in this block
$\bar{\mathbf{X}} = \mathbf{R}^l$	$\hat{\bar{c}}_{ij} = \tilde{\bar{c}}_{ij} - \delta$ in this block	$\hat{\bar{c}}_{ij} = \tilde{\bar{c}}_{ij}$ in this block	$\hat{u}_i = \tilde{u}_i + \delta$ in this block
New dual solution $\hat{v}$	$\hat{v}_j = \tilde{v}_j$ in this block	$\hat{v}_j = \tilde{v}_j - \delta$ in this block	

Dual feasibility is maintained in the new tree if $\hat{\bar{c}}_{ij} \geqq 0$ for all i, j. For this we must choose

$$\delta = \min.\{\tilde{\bar{c}}_{ij} = c_{ij} - \tilde{u}_i - \tilde{v}_j : R_i \in \bar{\mathbf{X}}, C_j \in \mathbf{Y}\}$$

Since the present tree $\mathbb{T}$ is dual feasible, this δ will be $\geqq 0$. Consequently, the entering arc in this pivot step is selected to be an arc (g, h) with $g \in \bar{\mathbf{X}}$ and $h \in \mathbf{Y}$, which attains the minimum for δ in the above equation. The new tree is obtained by replacing (w, l) in $\mathbb{T}$ with (g, h).

In the above equation, δ will be ∞ only if all the cells (i, j) with $i \in \bar{\mathbf{X}}$, and $j \in \mathbf{Y}$ are constrained to have no allocation in them. As before, this is an indication that there exists no feasible assignment, and the method terminates if this happens.

Selection of the Initial Dual Feasible Spanning Tree

If the assignment problem being solved is on the complete bipartite network (i.e., every variable x_{ij} is eligible to have value 1), an initial dual feasible spanning tree can be taken to be the tree $\mathbb{T}^0$ consisting of the following arcs: all arcs $(R_p, C_j), j = 1$ to n, for some p, and arcs (R_i, C_{q^i}) for $i \neq p$, where q^i is an index attaining the minimum for u_i^0 = min. $\{ c_{ij} - c_{pj} : j = 1 \text{ to } n \}$. Let $u_p^0 = 0$, and $v_j^0 = c_{pj}$, for $j = 1$ to n. It can be verified that (u^0, v^0) is the dual basic solution associated with $\mathbb{T}^0$, and that it is dual feasible. The row signature vector of $\mathbb{T}^0$ is $(n, 1, \ldots, 1)$. Hence it has $n - 1$ terminal row nodes.

In dense or sparse problems, other methods for obtaining an initial dual feasible spanning tree are discussed in Murty and Witzgall [1977], and Section 13.6, Chapter 13 in Murty [1983 of Chapter 1]. In highly sparse problems, it may be necessary to include some artificial arcs (among those which correspond to cells that are constrained not to have any allocations) in the network, associated with very large positive cost coefficients (equal to $\beta = 1 + n(\text{max. } \{|c_{ij}| : (i, j) \text{ can have an allocation}\})$), in order to get an initial dual feasible spanning tree. If any of these artificial cells have an allocation in the final optimum assignment, it is an indication that the original problem has no feasible assignment.

The only remaining thing needed to describe a signature method is the dropping arc selection strategy to be used. We discuss several signature methods next.

3.4.1 Signature Method 1

This method is initiated with a dual feasible spanning tree. It is entirely guided by the row signature vector, the column signature vector and the primal basic solution are never explicitly used in carrying out the algorithm.

The method seeks a tree whose row signature vector contains a single 1 and otherwise 2s. Given a dual feasible spanning tree $\mathbb{T}$ with more than one terminal row node, it selects one of these nodes, say t, and designates it as the **target row node**. There must be at least one row node in $\mathbb{T}$ of degree $\geqq 3$. It selects one of these, say s, as the **source row node**. Once the target row node t and the source row node s in the tree $\mathbb{T}$ are selected, the dropping arc in this method is always the arc incident at s on the path $\mathcal{P}(s, t, \mathbb{T})$. If it is the arc (s, l), l is a column node with degree $\geqq 2$ in $\mathbb{T}$, as the path $\mathcal{P}(s, t, \mathbb{T})$ itself contains two arcs incident at l. Let (g, h) be the entering arc to replace (s, l), determined as described above, and $\mathbb{T}^1$ the new tree obtained after this pivot step. Let $d(s), d(g), d^1(s), d^1(g)$ be the degrees of s, g in $\mathbb{T}, \mathbb{T}^1$ respectively. Then $d^1(s) = d(s) - 1, d^1(g) = d(g) + 1$. So, g which is the row node on the entering arc, is not a terminal row node in $\mathbb{T}^1$. If g was a terminal row node in $\mathbb{T}$, the number of terminal row nodes has decreased by one in this pivot step, this completes a stage in the method. If the number of terminal row nodes in $\mathbb{T}^1$ is one, it is primal feasible, and hence optimal, and the method terminates. Otherwise, it goes to the next stage with $\mathbb{T}^1$.

If g was not a terminal row node in $\mathbb{T}$, continue the stage with $\mathbb{T}^1$. Keep the same target row node t, but make g the next source row node and carry out the next pivot step. In the next pivot step, the set $\bar{\mathbf{X}}$ in the row partition becomes smaller (it loses node g) and the set $\mathbf{Y}$ in the column partition becomes larger.

So, the target row node remains the same in a stage, but the source row node keeps changing. The set $\bar{\mathbf{X}}$ gets smaller, and the set $\mathbf{Y}$ gets larger, until in the last pivot step of the stage the row node in the entering arc is a terminal row node before that arc enters. The row nodes that drop off from $\bar{\mathbf{X}}$ during this stage are all nonterminal row nodes, so, if the initial tree in this stage had r terminal row nodes, the number of pivot steps in it will be $\leqq n - r$. If carried out directly, the computational effort in each pivot step (to compute δ and update all the reduced cost coefficients) is at most $O(n^2)$, and hence the total effort in this stage may be $O((n-r)n^2)$, or $O(n^3)$. However, using the technique described under the $O(n^3)$ implementation of the Hungarian method, this stage can be implemented in such a way that the computational

effort in it is at most $O(n^2)$. In this implementation, all the reduced cost coefficients are computed in the first pivot step of the stage, and for each row node R_i in the set $\bar{\mathbf{X}}$ at that time an index of the form $[w_i, p_i]$ is defined, where p_i = minimum current reduced cost coefficient in R_i among columns in the present set $\mathbf{Y}$, and w_i is a column in $\mathbf{Y}$ where this minimum occurs. In each pivot step of the stage, we do the following work.

(i) δ = minimum of p_i in the indices of rows in the set $\bar{\mathbf{X}}$ in that step. Computing this δ therefore takes only $O(n)$ effort using these indices and if the row $R_q \in \bar{\mathbf{X}}$ attains this minimum (break ties arbitrarily) and the present index on R_q is $[w_q, p_q]$, then the arc (R_q, w_q) is the entering arc in this pivot step.

(ii) Update the dual solution as indicated in Array 3.6

(iii) Get the new sets $\bar{\mathbf{X}}$, $\mathbf{Y}$. Eliminate the indices on all the rows no longer in $\bar{\mathbf{X}}$.

(iv) Subtract δ from the p_i entry in the index of each row node in the new $\bar{\mathbf{X}}$.

(v) For each column C_j that just joined the set $\mathbf{Y}$, compute the correct reduced cost coefficient $\bar{c}'_{ij} = c_{ij} - u'_i - v'_j$, where (u', v') is the new dual solution, for rows R_i in the new set $\bar{\mathbf{X}}$ only, and if the present p_i on this row satisfies $p_i > \bar{c}'_{ij}$ change the index for R_i to $[C_j, \bar{c}'_{ij}]$; otherwise leave this index unchanged.

Under this implementation, in each cell, the reduced cost coefficient is computed once at the beginning of the stage, and at most once more during the entire stage. It can be verified that the overall computational effort during a stage is at most $O(n^2)$ under this implementation. Each stage reduces the number of terminal row nodes by one, hence there can be at most n stages in the method. So, the overall computational effort in this method is at most $O(n^3)$, which is of the same order as that of the Hungarian method.

As an example consider the problem in which $n = 6$, and the dual feasible spanning tree $\mathbb{T}$ at the beginning of a stage is the one in Figure

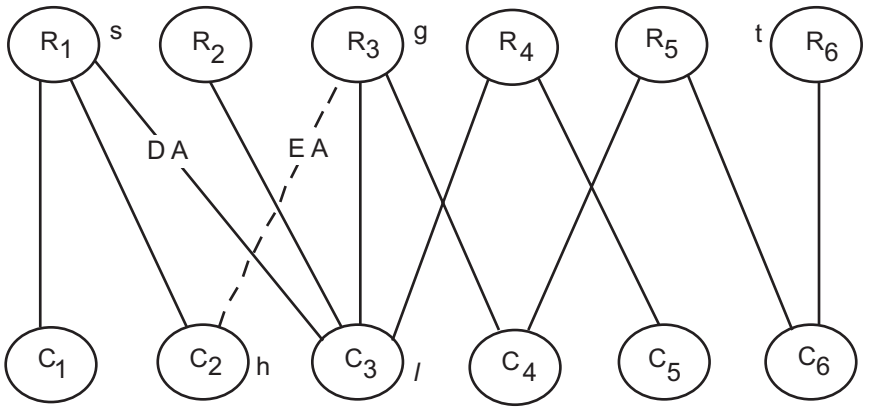

Figure 3.6: DA, EA denote the dropping arc, entering arc respectively.

3.6. Orientations of the arcs are not shown in the figure, each line is directed from the row node to the column node on it. To keep the presentation simple, we do not include the arc cost coefficients in this illustration, but we give the entering arc in each step. $\mathbb{T}$ has two terminal row nodes, R_2 and R_6, of which R_6 has been selected as the target row node t. R_1 of degree 3 in $\mathbb{T}$ has been chosen as the source row node s. (R_1, C_3), the first arc on the path $\mathcal{P}(R_1, R_6, \mathbb{T})$ is the dropping arc in this pivot step. We have $\mathbf{R}^s = \mathbf{X} = \{R_1\}$, $\mathbf{C}^s = \mathbf{Y} = \{ C_1, C_2\}$, $\mathbf{R}^l = \bar{\mathbf{X}} = \{R_2, R_3, R_4, R_5, R_6\ \}$, $\mathbf{C}^l = \bar{\mathbf{Y}} = \{\ C_3, C_4, C_5, C_6\}$. Suppose the arc (R_3, C_2) marked with a dashed line in Figure 3.7 is the entering arc. So, in the notation used above for describing the pivot step, $g = R_3, h = C_2$. Since $g = R_3$ is not a terminal row node in Figure 3.6, the stage continues. The next tree $\mathbb{T}^1$ is drawn in Figure 3.7.

R_6 continues to be the target row node t. The new source row node is R_3. The dropping arc is (R_3, C_4). $\bar{\mathbf{X}} = \{R_5, R_6\}$. $\mathbf{Y} = \{C_1, C_2, C_3, C_5\}$. It can be verified that the set $\bar{\mathbf{X}}$ became smaller, and $\mathbf{Y}$ became larger. In $\mathbb{T}^1$ the entering arc is (R_5, C_2), so $g = R_5$. This is not a terminal row node in $\mathbb{T}^1$, so the stage continues. The next tree $\mathbb{T}^2$ is drawn in Figure 3.8. In $\mathbb{T}^2$, $\bar{\mathbf{X}} = \{R_6\}$, $\mathbf{Y} = \{C_1, C_2, C_3, C_4, C_5\}$, and (R_6, C_3) is the entering edge. Since $g = R_6$ is a terminal row node

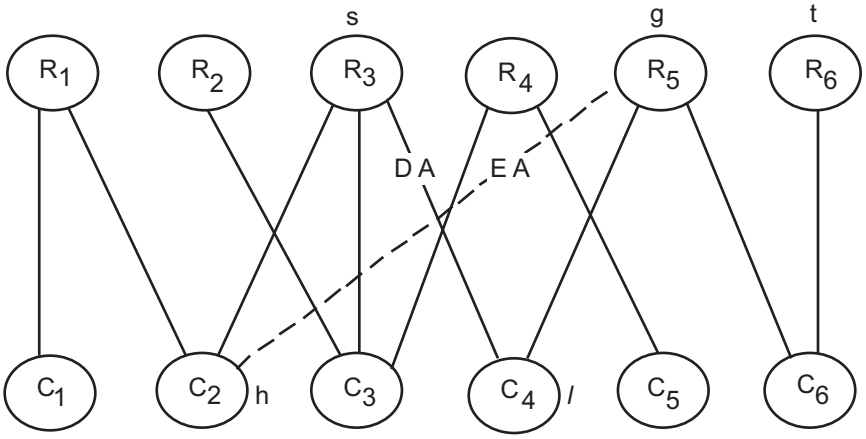

Figure 3.7: DA, EA denote the dropping arc, entering arc respectively.

in $\mathbb{T}^2$ (it is actually the target row node), with the pivot step in it, the stage is completed. The next tree $\mathbb{T}^3$ is drawn in Figure 3.9. The row signature vector in this tree is (2, 1, 2, 2, 2, 2), so this is an optimum tree. The solid arcs in Figure 3.9 define an optimum assignment.

The method is executed without ever computing the primal basic solution corresponding to any tree, except for the very last tree to get the optimum assignment. Let $\tilde{\mathbb{T}}, \hat{\mathbb{T}}$ be two consecutive trees obtained in the method in that order, with $x(\tilde{\mathbb{T}}) = (x_{ij}(\tilde{\mathbb{T}}))$, $x(\hat{\mathbb{T}}) = (x_{ij}(\hat{\mathbb{T}}))$, the associated primal basic solutions. Let (s, l) be the dropping arc from $\tilde{\mathbb{T}}$, which is replaced by the entering arc (g, h) in the pivot step that led to $\hat{\mathbb{T}}$. Let $\mathbb{C}$ be the fundamental cycle of (g, h) wrt $\tilde{\mathbb{T}}$. It contains the dropping arc (s, l). Orient $\mathbb{C}$ so that the dropping arc (s, l) is a reverse arc. Let θ be the flow amount on the arc (s, l) in $x(\tilde{\mathbb{T}})$. In $x(\tilde{\mathbb{T}})$, add θ to the flow amounts on all the forward arcs of $\mathbb{C}$, and subtract θ from the flow amounts on all the reverse arcs of $\mathbb{C}$, this leads to $x(\hat{\mathbb{T}})$. In the same way, the primal basic solution corresponding to each tree obtained in the method can be computed by updating the previous basic solution along the fundamental cycle of the entering arc.

We will now show that if this method is initiated with the dual feasible spanning tree $\mathbb{T}^0$ discussed above, then it is very similar to

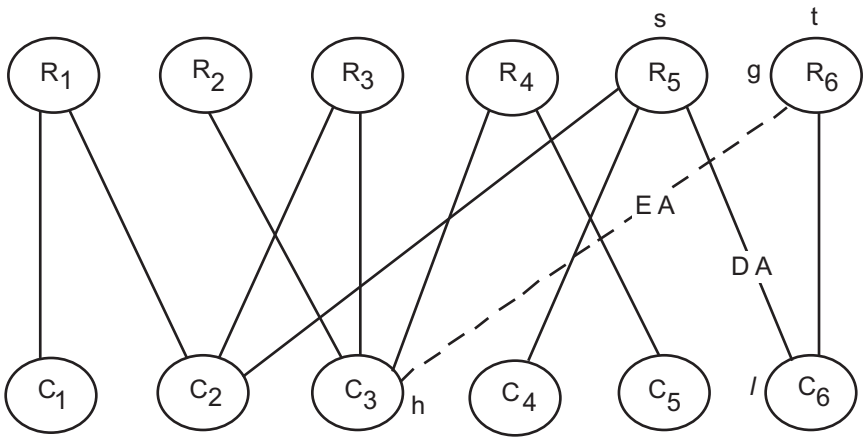

Figure 3.8: DA, EA denote the dropping arc, entering arc respectively.

the dual simplex algorithm in its choice of the dropping arc in each pivot step. This follows from the following results. We assume that the initial spanning tree is $\mathbb{T}^0$ constructed with $p = 1$.

1. Let $\mathbb{T}$ be a tree obtained during the method, and $(\mathbf{X}, \bar{\mathbf{X}})$ the partition of row nodes in the pivot step carried in it. Then every row node in $\bar{\mathbf{X}}$ has degree 1 or 2 in $\mathbb{T}$. The reason for this is the following. From the initial spanning tree $\mathbb{T}^0$ and the manner in which the method is executed, it is clear that R_1 is the only row node whose degree can be > 2 in any tree obtained during the method. If $\mathbb{T}$ is the first tree in any stage, $\mathbf{X}$ contains the source row node s which is R_1, as this is the only row node with degree > 2 in $\mathbb{T}$, the result holds. If $\mathbb{T}$ is not the first tree in a stage, the result follows from this and the fact that the set $\bar{\mathbf{X}}$ gets smaller as we move from one pivot step to the next in a stage.

2. Let (s, l) be the dropping arc in a pivot step carried out on a spanning tree $\mathbb{T}$ in this method. Then $x_{sl}(\mathbb{T}) \leq 0$. To see this, let $(\mathbf{X}, \bar{\mathbf{X}})$, $(\mathbf{Y}, \bar{\mathbf{Y}})$ be the row node and column node partition in the pivot step carried out on $\mathbb{T}$. $\bar{\mathbf{X}}, \bar{\mathbf{Y}}$ are the sets of row, column nodes in the component $\mathbb{T}^l$ containing column node l when (s, l) is deleted from $\mathbb{T}$. By 1, each node in $\bar{\mathbf{X}}$ has degree 1

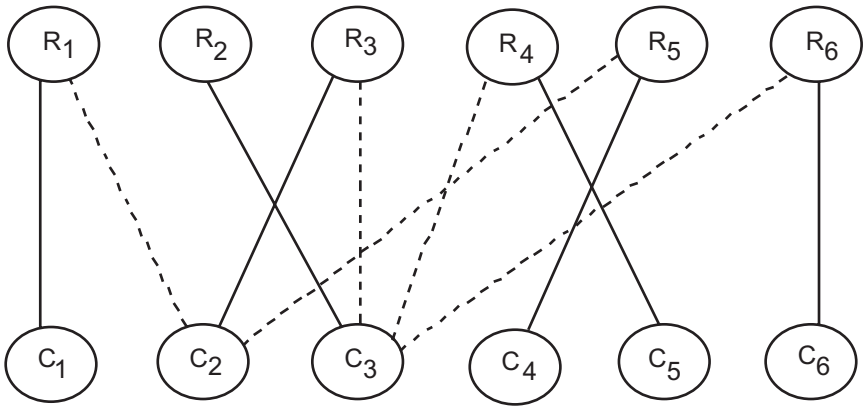

Figure 3.9:

or 2 in $\mathbb{T}$, let n_1, n_2 be the numbers of these nodes with degrees 1, 2 respectively. This implies that $\mathbb{T}^l$ has n_1+n_2 row nodes, and n_1+2n_2 arcs. Since $\mathbb{T}^l$ is a tree, these facts imply that the number of column nodes in it, $|\bar{\mathbf{Y}}| = n_1 + 2n_2 + 1 - (n_1 + n_2) = n_2 + 1$.

$x(\mathbb{T})$ is obtained by substituting $x_{ij} = 0$ for all (i, j) corresponding to arcs not in $\mathbb{T}$ in the system of equality constraints in (3.1) and then solving the remaining system for the values of x_{ij} for (i, j) corresponding to arcs in $\mathbb{T}$. From this, and the fact that $|\bar{\mathbf{X}}|$ $= n_1 + n_2$, $|\bar{\mathbf{Y}}| = 1 + n_2$, and that node l is the column node on the dropping arc (s, l), we get the following relations by summing the equality constraints in (3.1) over rows in $\mathbb{T}^l$, and columns in $\mathbb{T}^l$ separately.

$$\begin{aligned} \sum(x_{ij}(\mathbb{T}) : \text{ over } (i,j) \in \mathbb{T}^l) &= n_1 + n_2 \\ \sum(x_{ij}(\mathbb{T}) : \text{ over } (i,j) \in \mathbb{T}^l) &= n_2 + 1 - x_{sl}(\mathbb{T}) \end{aligned}$$

So, $n_2+1-x_{sl}(\mathbb{T}) = n_1+n_2$, which implies that $x_{sl}(\mathbb{T}) = 1-n_1 \leqq 0$, since $\mathbb{T}^l$ contains the target node which has degree 1 in $\mathbb{T}$.

3. The dual objective value is nondecreasing during the method. This follows from the result in 2.

These results imply that if Signature method 1 is initiated with the spanning tree $\mathbb{T}^0$, then it can be viewed as a dual simplex method, even though it does not always choose pivots in the usual dual simplex way, since the primal basic value associated with the dropping arc may sometimes be 0 instead of being < 0.

Signature method 1 can of course be initiated with any dual feasible spanning tree in G. However, in this case the primal basic value associated with the dropping arc may not be $\stackrel{<}{=} 0$ in some pivot steps (the proof of this property given above is based on the fact that all the trees obtained under the method have at most one row node of degree > 2 which may not hold in this general case). So, the dual objective value may increase, decrease, or remain unchanged in a pivot step in this general version, and hence this general version cannot always be interpreted as a dual simplex method, even though it moves from a dual feasible tree to an adjacent dual feasible tree in each pivot step. This general version also takes no more than n stages, and the computational effort in it is bounded above by $O(n^3)$.

Exercises

3.21 Let $\mathbb{T}$ be a spanning tree in G whose row signature vector $d = (d_i)$ contains a unique entry equal to 1, which is d_p. For each $j = 1$ to n, define μ_j to be the i where i is such that (R_i, C_j) is the arc incident at C_j on the path $\mathcal{P}(R_p, C_j, \mathbb{T})$. Prove that the primal basic solution associated with $\mathbb{T}$ is the assignment $\{(\mu_j, j)$: $j = 1$ to $n\}$.

3.22 Prove that if two spanning trees of G correspond to different assignments, they cannot have the same signature vectors.

3.23 Let $\mathbb{T}$ be a spanning tree in G whose row signature vector contains a single 1 entry, and $\bar{\mathbb{T}}$ the corresponding cotree. Prove that $(\mathbb{T}, \bar{\mathbb{T}}, \emptyset)$ is a strongly feasible partition (see Section 5.5.1 for definition) for (3.1) when the root node is the unique terminal row node in $\mathbb{T}$.

3.24 Solve the assignment problem of order 5 with the following cost matrix, to find a minimum cost assignment.

$$\begin{pmatrix} 14 & 18 & 15 & 10 & 10 \\ 18 & 17 & 15 & 8 & 8 \\ 16 & 16 & 24 & 25 & 12 \\ 19 & 10 & 8 & 14 & 11 \\ 22 & 15 & 28 & 24 & 12 \end{pmatrix}$$

3.25 Consider a spanning tree $\mathbb{T}$ in G, in which all row nodes R_2 to R_n have degree $\leqq 2$, and only R_1 has degree > 2. Prove that the associated primal basic solution $(x_{ij}(\mathbb{T}))$ is primal feasible iff $x_{1j}(\mathbb{T}) \geqq 0$ for all j. (Balinski [1985])

3.26 A Worst Case Example Consider the assignment problem of order n with the cost matrix $c = (c_{ij})$ where $c_{ij} = (n-i)(j-1)$ for each i, j. When initiated with the dual feasible spanning tree $\mathbb{T}^0$ discussed above, corresponding to $p = 1$, show that Signature method 1 requires $(n-1)(n-2)/2$, or $O(n^2)$ pivot steps to solve this problem. (Balinski [1985, 1986])

3.4.2 An Inductive Signature Method

Consider an assignment problem of order n with $c = (c_{ij} : i, j = 1 \text{ to } n)$ as the cost matrix. For $r = 1$ to n, define $c^r = (c_{ij} : i, j = 1 \text{ to } r)$, and the assignment problem of order r to be the one with row nodes $R_1, \ldots, R_r$, column nodes $C_1, \ldots, C_r$, and cost matrix c^r.

Suppose we have an optimum tree $\mathbb{T}(r)$ for problem r. Let $u^r = (u_1^r, \ldots, u_r^r), v^r = (v_1^r, \ldots, v_r^r)$ be the dual basic solution associated with $\mathbb{T}(r)$. Define $\hat{u}_{r+1} = \min \{ c_{r+1,j} - v_j^r : j = 1 \text{ to } r\}$, and let q be a j which attains this minimum, break ties arbitrarily. Define $\hat{u}^{r+1} = (u_1^r, \ldots, u_r^r, \hat{u}_{r+1})$. Define $\hat{v}_{r+1} = \min \{ c_{i,r+1} - \hat{u}_i^{r+1} : i = 1 \text{ to } r+1 \}$, and let w be $r+1$ if it attains this minimum, otherwise w is any i which attains this minimum. Define $\hat{v}^{r+1} = (v_1^r, \ldots, v_r^r, \hat{v}_{r+1})$. In the bipartite network for problem $r+1$ define the spanning tree $\mathbb{T}(r+1)$ to be the one consisting of all the arcs in $\mathbb{T}(r)$ and the two arcs (R_{r+1}, C_q)

and (R_w, C_{r+1}). The dual basic solution corresponding to $\mathbb{T}(r+1)$ in problem $r+1$ is $(\hat{u}^{r+1}, \hat{v}^{r+1})$. This can be verified to be dual feasible. If $w = r+1$, $\mathbb{T}(r+1)$ has exactly one terminal row node (the one which is a terminal row node in $\mathbb{T}(r)$, since R_{r+1} has degree 2 in $\mathbb{T}(r+1)$ in this case), and hence an optimum tree for problem $r+1$. If $w \neq r+1$, $\mathbb{T}(r+1)$ has exactly two terminal row nodes (these are R_{r+1} and the terminal row node in $\mathbb{T}(r)$). So, problem $r+1$ can be solved in this case by Signature method 1 initiated with $\mathbb{T}(r+1)$, in just one stage, this takes a computational effort of at most $O(r^2)$.

This method proceeds inductively on r. It is initiated with the problem of order 2 for which the solution is trivially obtained. For $r = 2$ to $n-1$, it solves problem $r+1$ beginning with an optimum tree obtained for problem r. The final problem is the original problem. The overall computational effort is $\sum O(r^2) = O(n^3)$.

Suppose we are solving a sparse assignment problem using this method. After solving problem r, it may be necessary to rearrange the remaining rows (and columns in the same order) and select row $r+1$ among them appropriately, in order to guarantee that $\mathbb{T}(r)$ can be extended to an initial dual feasible spanning tree for problem $r+1$, using only cells in which allocations are permitted; or artificial cells can be introduced as mentioned earlier.

The worst case upper bound on the computational effort of this inductive version can be shown to be achieved on the problem with cost matrix $c = (c_{ij})$ where $c_{ij} = ij$ for all i, j. The Hungarian method and Signature method 1 initiated with $\mathbb{T}^0$ also require the worst case upper bound computational effort on this problem.

3.4.3 Signature Method 2 : A Dual Simplex Method

We now discuss signature methods which are strictly dual simplex methods on all counts. In these methods, the initial dual feasible spanning tree in G is $\mathbb{T}^0$ discussed above constructed with $p = 1$. Here all the trees will be treated as rooted trees with R_1 permanently designated as the root node (because in the initial tree, R_1 has degree n). These methods use both the row and column signature vectors.

An arc e in a spanning tree in G is said to be an **odd arc** if the

son(e) is the column node on it, **even arc** otherwise. Also given two nodes p, q we say that p **is higher (lower) than** q if p is an ancestor (descendent) of q. Likewise, given two in-tree arcs e, e', we say that e is **higher (lower) than** e' if e is on the predecessor path of parent(e') (e' is on the predecessor path of parent(e)).

We discuss two versions of this method based on the following dropping arc choice rules.

DROPPING ARC CHOICE RULE 1 If a stage has just been completed and the present spanning tree is $\mathbb{T}$, terminate if it has a unique terminal column node, since $\mathbb{T}$ is optimal then. Otherwise, select any odd arc (R_i, C_j) in $\mathbb{T}$ where C_j has degree $\geqq 3$ in $\mathbb{T}$, as the dropping arc in the initial pivot step for the next stage. If you are not at the beginning of a stage, the preceding pivot step must have caused the degree of the column node, say l, on the entering arc in that step to increase to 3 or more. Select the dropping arc to be the unique odd arc incident at l in $\mathbb{T}$.

DROPPING ARC CHOICE RULE 2 Let $\mathbb{T}$ be the present dual feasible but primal infeasible spanning tree in G. Define PC($\mathbb{T}$) to be the set of all in-tree arcs that tie for the most negative primal basic value among all in-tree arcs on which the column node has degree $\geqq 3$. If you are beginning a new stage, select the dropping arc to be the highest arc in PC($\mathbb{T}$). If you are not at the beginning of a stage, the preceding pivot step must have caused the degree of the column node, say h, of the entering arc in that step to increase to 3 or more. Choose the dropping arc to be the unique odd arc on which the column node has degree $\geqq 3$ that is highest above h in $\mathbb{T}$.

Dropping arc choice rule 2 helps to increase the sets **X, Y** very rapidly, and thus gain efficiency. See Balinski [1986] for a proof that in both these versions, the primal basic value associated with the dropping arc is always $\leqq -1$. Hence these versions are strictly dual simplex methods. Balinski [1985, 1986] initiated the signature methods, and the Signature methods 1,2 are both due to him. The inductive signature method is due to Goldfarb [1985].

3.5 Other Methods for the Assignment Problem

There are several other methods for the assignment problem. The algorithms of Chapter 5 can be specialized to the assignment problem. For example the specialization of the algorithms of Section 5.6 leads to relaxation methods for the assignment problem (Bertsekas [1981]). The method of Section 5.8.1 specialized to the assignment problem leads to the shortest augmenting path algorithm for it.

3.6 Algorithm for Ranking Assignments in Nondecreasing Order of Cost

Consider the assignment problem of order n with $c = (c_{ij})$ as the cost matrix. Here we discuss an efficient method that actually ranks the assignments in nondecreasing order of cost starting with a minimum cost one. In each step it obtains one new assignment in the ranked sequence with a computational effort of at most $O(n^3)$, and can be continued as long as necessary, and terminated whenever a sufficient number of assignments in the ranked sequence have been obtained.

We will denote the assignment $x = (x_{ij})$ by the set $\{(i,j) : x_{ij} = 1\}$. Correspondingly, we write $(i,j) \in x, or \notin x$ to indicate that $x_{ij} = 1$ or 0 respectively. Let $a(1)$ denote a minimum cost assignment, and $a(1), a(2), \ldots, a(r), \ldots$ the ranked sequence of assignments satisfying for all $r \geq 2$

$$a(r) = \text{ a min. cost assignment excluding } a(1), \ldots, a(r-1)$$

In the algorithm we will use subsets of assignments called **nodes**. A node is a nonempty subset of assignments a of the form

$$\begin{aligned} \mathbf{N} &= \{(i_1,j_1), \ldots, (i_r,j_r); \overline{(m_1,p_1)}, \ldots, \overline{(m_u,p_u)}\} \\ &= \{a : (i_1,j_1) \in a, \ldots, (i_r,j_r) \in a; \\ &\qquad (m_1,p_1) \notin a, \ldots, (m_u,p_u) \notin a\} \end{aligned} \tag{3.20}$$

The cells $(i_1, j_1), \ldots, (i_r, j_r)$ are **specified to be contained in**, and the cells $(m_1, p_1), \ldots, (m_u, p_u)$ are **specified to be excluded from** each assignment in $\mathbf{N}$. In the definition of the node $\mathbf{N}$, $i_1, \ldots, i_r$ will all be distinct, and the same property will hold for $j_1, \ldots, j_r$. Also, in all nodes generated in the algorithm, all the specified to be excluded cells will belong to the same row of the array, i.e., $m_1 = m_2 = \ldots = m_u$ in $\mathbf{N}$. The matrix obtained by striking off rows $i_1, \ldots, i_r$ and columns $j_1, \ldots, j_r$ from c and replacing the entries in positions $(m_1, p_1), \ldots, (m_u, p_u)$ by infinity or a very large positive number, is known as *the remaining cost matrix corresponding to node* $\mathbf{N}$ and is denoted by $c_{\mathbf{N}}$. A minimum cost assignment in $\mathbf{N}$ can be found by solving the assignment problem of order $n - r$ with $c_{\mathbf{N}}$ as the cost matrix. Let $x_{\mathbf{N}}, z_{\mathbf{N}}$ denote a minimum cost assignment in $\mathbf{N}$ and its objective value.

One of the operations performed in the algorithm is that of **partitioning a node using a minimum cost assignment in it**. Let $\mathbf{N}$ be the node in (3.20) and $x_{\mathbf{N}} = \{(i_1, j_1), \ldots, (i_r, j_r), (s_1, t_1), \ldots, (s_{n-r}, t_{n-r})\}$ be an optimum assignment in it. Each of $(s_1, t_1), \ldots, (s_{n-r}, t_{n-r})$ should be distinct from $(m_1, p_1), \ldots, (m_u, p_u)$ from the definition of $\mathbf{N}$. Let

$$\begin{aligned}
\mathbf{N}_1 &= \{(i_1, j_1), \ldots, (i_r, j_r); \overline{(m_1, p_1)}, \ldots, \overline{(m_u, p_u)}, \overline{(s_1, t_1)}\} \\
\mathbf{N}_2 &= \{(i_1, j_1), \ldots, (i_r, j_r); (s_1, t_1); \overline{(m_1, p_1)}, \ldots, \overline{(m_u, p_u)}, \overline{(s_2, t_2)}\} \\
&\vdots \\
\mathbf{N}_{n-r-1} &= \{(i_1, j_1), \ldots, (i_r, j_r); (s_1, t_1), \ldots, (s_{n-r-2}, t_{n-r-2}); \\
&\qquad \overline{(m_1, p_1)}, \ldots, \overline{(m_u, p_u)}, \overline{(s_{n-r-1}, t_{n-r-1})}\}
\end{aligned}$$

The partitioning of $\mathbf{N}$ using $x_{\mathbf{N}}$ generates the mutually disjoint subnodes $\mathbf{N}_1, \ldots, \mathbf{N}_{n-r-1}$, and the partition itself is

$$\mathbf{N} = \{x_{\mathbf{N}}\} \cup \cup_{v=1}^{n-r-1} \mathbf{N}_v \tag{3.21}$$

The algorithm maintains a **list** which is a set of nodes. Each node in the list is stored together with a minimum cost assignment in it and its objective value.

THE ASSIGNMENT RANKING ALGORITHM

Initial step Find a minimum cost assignment $a(1) = \{(1, j_1), \ldots, (n, j_n)\}$, say. Let the list at this stage be $\{\overline{(1, j_1)}\}$, $\{(1, j_1); \overline{(2, j_2)}\}, \ldots, \{(1, j_1), \ldots, (n-2, j_{n-2}); \overline{(n-1, j_{n-1})}\}$. A minimum cost assignment in each of these nodes is found, and it is stored together with the node in the list. Go to the next step.

General step Suppose $a(1), \ldots, a(r)$ in the ranked sequence have already been obtained, and the list of nodes at this stage is $\mathbf{M}_1, \ldots, \mathbf{M}_\ell$. From the manner in which these are generated, $\mathbf{M}_1, \ldots, \mathbf{M}_\ell$ will be mutually disjoint, and their union will be the set of all assignments excluding $a(1), \ldots, a(r)$. Let $x_{\mathbf{M}_d}$ be an optimum assignment in the node $\mathbf{M}_d$ and $z_{\mathbf{M}_d}$ its objective value, for $d = 1$ to ℓ. So, the next assignment in the ranked sequence is $a(r+1) = x_{\mathbf{M}_d}$ for a d satisfying $z_{\mathbf{M}_d} = \text{min. } \{z_{\mathbf{M}_1}, \ldots, z_{\mathbf{M}_\ell}\}$. If $a(r+1)$ is the last assignment in the ranked sequence that is required, the algorithm terminates here. Otherwise, delete $\mathbf{M}_d$ from the list, and partition it using $x_{\mathbf{M}_d}$. Let $\mathbf{M}_{d,1}, \ldots, \mathbf{M}_{d,f}$ be the subnodes generated. Find a minimum cost assignment in each of them, add each of these subnodes to the list and go to the next step.

Discussion

If a predetermined number, h, of assignments in the ranked sequence are required, only the h nodes that are associated with the least objective values are stored in the list and the rest are pruned. If it is desired to obtain all the assignments whose cost is $\leqq$ some predetermined number, α, only those nodes in which the minimum objective value is $\leqq \alpha$ are stored in the list, and the rest are pruned.

As an example consider the assignment problem of order 10 with the following cost matrix.

$$c = \begin{pmatrix} 7 & 51 & 52 & 87 & 38 & 60 & 74 & 66 & 0 & 20 \\ 50 & 12 & 0 & 64 & 8 & 53 & 0 & 46 & 76 & 42 \\ 27 & 77 & 0 & 18 & 22 & 48 & 44 & 13 & 0 & 57 \\ 62 & 0 & 3 & 8 & 5 & 6 & 14 & 0 & 26 & 39 \\ 0 & 97 & 0 & 5 & 13 & 0 & 41 & 31 & 62 & 48 \\ 79 & 68 & 0 & 0 & 15 & 12 & 17 & 47 & 35 & 43 \\ 76 & 99 & 48 & 27 & 34 & 0 & 0 & 0 & 28 & 0 \\ 0 & 20 & 9 & 27 & 46 & 15 & 84 & 19 & 3 & 24 \\ 56 & 10 & 45 & 39 & 0 & 93 & 67 & 79 & 19 & 38 \\ 27 & 0 & 39 & 53 & 46 & 24 & 69 & 46 & 23 & 1 \end{pmatrix} \tag{3.22}$$

The minimum cost assignment in this example is $a(1) = \{(1, 9), (2, 7), (3, 3), (4, 8), (5, 6), (6, 4), (7, 10), (8, 1), (9, 5), (10, 2)\}$, with an objective value of 0. The list at the end of the initial step consists of the following nodes. The minimum objective value in each node is also recorded.

$$\begin{aligned}
\mathbf{M}_1 &= \{\overline{(1,9)}\}, z_{\mathbf{M}_1} = 10 \\
\mathbf{M}_2 &= \{(1,9), \overline{(2,7)}\}, z_{\mathbf{M}_2} = 14 \\
\mathbf{M}_3 &= \{(1,9), (2,7), \overline{(3,3)}\}, z_{\mathbf{M}_3} = 14 \\
\mathbf{M}_4 &= \{(1,9), (2,7), (3,3), \overline{(4,8)}\}, z_{\mathbf{M}_4} = 1 \\
\mathbf{M}_5 &= \{(1,9), (2,7), (3,3), (4,8), \overline{(5,6)}\}, z_{\mathbf{M}_5} = 15 \\
\mathbf{M}_6 &= \{(1,9), (2,7), (3,3), (4,8), (5,6), \overline{(6,4)}\}, z_{\mathbf{M}_6} = 53 \\
\mathbf{M}_7 &= \{(1,9), (2,7), (3,3), (4,8), (5,6), (6,4), \overline{(7,10)}\}, z_{\mathbf{M}_7} = 45 \\
\mathbf{M}_8 &= \{(1,9), (2,7), (3,3), (4,8), (5,6), (6,4), (7,10), \overline{(8,1)}\}, z_{\mathbf{M}_8} = 47 \\
\mathbf{M}_9 &= \{(1,9), (2,7), (3,3), (4,8), (5,6), (6,4), (7,10), (8,1), \overline{(9,5)}\}, z_{\mathbf{M}_9} = 56
\end{aligned}$$

Comparing the values of $z_{\mathbf{M}_1}$ to $z_{\mathbf{M}_9}$, we find that $a(2)$ is a minimum cost assignment in $\mathbf{M}_4$. It is $a(2) = \{(1, 9), (2, 7), (3, 3), (4, 2), (5, 6), (6, 4), (7, 8), (8, 1), (9, 5), (10, 10)\}$ with an objective value of 1. If it is required to find $a(3)$, then $\mathbf{M}_4$ should be partitioned using $a(2)$ and the algorithm continued.

Comment 3.3 This ranking algorithm has been taken from Murty [1968]. The same approach has been extended to rank the chains between a pair of nodes in a directed network (Section 4.7); to rank the spanning trees in an undirected network(Section 9.3); to rank the cuts in a capacitated network (Hamacher [1982], Hamacher, Picard and Queyranne [1984]); and in general to rank the solutions of any discrete optimization problem(Lawler [1972]).

.

3.7 Exercises

3.27 Consider the assignment ranking example given above for the assignment problem of order 10 with the cost matrix c given in (3.22). Find $a(3)$ to $a(6)$ in this example.

3.28 There are n jobs with given processing durations $t_i, i = 1$ to n; and job starting times, $s_i, i = 1$ to n. Each job must be processed without interruption on any one of the unlimited set of identical machines. Each machine can process any job, but no more than one job at a time. Formulate the problem of determining the smallest number of machines to process all the jobs, as one of finding a minimal chain decomposition of a poset. Solve the problem with the following data. (Gertsbakh and Stern [1978]).

i	1	2	3	4	5	6	7	8	9	10
t_i	30	25	10	18	65	7	9	10	3	18
s_i	4	30	50	68	7	19	8	110	150	88

3.29 An Application in Job Scheduling There are n jobs. For $i = 1$ to n, the processing of the ith job has to start at specified time a_i and must be finished at time $b_i (> a_i), t_i = b_i - a_i$ being the processing time for this job. All the jobs can be processed by a type of machine, of which several copies are available. The set-up time required for a machine to process job j after processing job i is $r_{ij} \geq 0$, where the (r_{ij}) satisfy the triangle inequality : $r_{ij} \leq r_{ip} + r_{pj}$, for all i, j, p. Define

a partial order $\not\prec$ on the set of jobs by $i \not\prec j$ iff $b_i + r_{ij} \leqq a_j$ (i.e., $i \not\prec j$ iff j can be processed by the same machine after it processes job i). Verify that this satisfies all the conditions for being a partial order. It is required to find the minimum number of machines needed to meet the given job schedule. Formulate this as the problem of finding a minimal chain decomposition of a poset. Solve the numerical problem with the following data: $n = 7, r_{ij} = 4$ for all $i \neq j$,

i	1	2	3	4	5	6	7
a_i	0	2	19	12	11	29	37
b_i	9	8	23	25	22	33	47

(Ford and Fulkerson [1962 of Chapter 1]).

3.30 Consider n men and n women such that each man-woman pair is either 'compatible' or 'incompatible.' If there is no way to match the men and women into n compatible couples, then prove that for some $p > 0$, there is a subset of p men who together are compatible with only r women where $r \leqq p - 1$.

3.31 Consider the transportation problem with an additional constraint on the left-hand side, where n, M and (c_{ij}) are data. Prove that this problem is equivalent to that of minimizing z subject to the system of constraints on the right-hand side.

$$\text{minimize } z = \sum_{i=1}^{n} \sum_{1=j}^{n} c_{ij} x_{ij}$$

$$\text{subject to } \sum_{j=1}^{n} x_{ij} \leqq 1, i = 1 \text{ to } n$$

$$\sum_{i=1}^{n} x_{ij} \leqq 1, j = 1 \text{ to } n$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \leqq n - M$$

$$x_{ij} \geqq 0, \text{ for all } i, j$$

$$\sum_{j=1}^{n+1} x_{ij} = 1, i = 1 \text{ to } n$$

$$\sum_{i=1}^{n+1} x_{ij} = 1, j = 1 \text{ to } n$$

$$\sum_{j=1}^{n+1} x_{n+1,j} = M + n$$

$$\sum_{i=1}^{n+1} x_{i,n+1} = M + n$$

$$x_{ij} \geqq 0$$

$$0 \leqq x_{n+1,n+1} \leqq n.$$

(Glover, Klingman, and Phillips [1984])

3.32 Let c be the cost matrix for an assignment problem of order n, for which $\bar{x}$ is an optimum assignment, and $(\bar{u}, \bar{v})$ an optimum dual solution. Let c' be a matrix obtained by changing the values in a single row, or a single column of c. Beginning with $\bar{x}, (\bar{u}, \bar{v})$, show that an optimum assignment with c' as the cost matrix can be obtained with a computational effort of at most $0(n^{\mathbf{2}})$. (Weintraub [1973])

3.33 Let a, b be two given vectors in $\mathbb{R}^n$. It is required to find a permutation of the vector b which brings it as close to a as possible. This is equivalent to finding a permutation P of order n which minimizes $\| a - Pb \|$. For any $p-$ norm $(1 \leqq p \leqq \infty)$ $\| \cdot \|$, show that this problem can be transformed into an assignment problem.

3.34 Consider the cost minimizing assignment problem with the cost matrix (c_{ij}) where $c_{ij} = u_i v_j$, with $u_1 \geqq u_2 \geqq \ldots \geqq u_n > 0$ and $0 < v_1 \leqq v_2 \leqq \ldots \leqq v_n$. Prove that the unit matrix is an optimum assignment for this problem.

3.35 Consider an $m \times n$ transportation problem. Let a *block* of cells in this problem refer to a subset of cells in the array of the following forms: a subset of cells within a single row or a single column of the array, or the set of all cells in a subset of rows or a subset of columns of the array. Suppose there are additional constraints in the problem, where each constraint is either a lower bound or an upper bound on the sum of flows in all the cells in some block. Show that the overall problem can still be posed as a minimum cost network flow problem.

3.36 Consider the following school timetable problem. There are n classes, m teachers, and p time periods in which lectures could be scheduled every week. Each period is one hour long. Following data is available. Discuss a method for constructing a timetable for class-teacher meetings over the available periods each week, subject to the constraints given, so that as many of the meetings as possible are scheduled. (DeWerra [1971])

α_i = number of periods that class i should meet per week,

$i = 1$ to n

β_j = number of periods that teacher j should teach per week, $j = 1$ to n

$$e_{it} = \begin{cases} 1, & \text{if class } i \text{ is unavailable for lecture during period } t \\ & \text{every week (they may have other non-lecture} \\ & \text{activities scheduled for that period), } i = 1 \text{ to,} n \\ 0, & \text{otherwise} \end{cases}$$

$$d_{jt} = \begin{cases} 1, & \text{if teacher } j \text{ is unavailable to lecture in period } t \\ & \text{every week, } j = 1 \text{ to } m,\ t = 1 \text{ to } p \\ 0, & \text{otherwise.} \end{cases}$$

3.37 Consider the assignment problem (3.1), (3.2). There may be many assignments which are optimal to this problem. Define a **second best valued assignment** in this problem to be an assignment whose objective value is strictly greater than that of an optimum assignment but has minimum cost among all such assignments. Develop a suitable modification of the partitioning routine using an optimum assignment to the problem, discussed in Section 3.6, to find a second best valued assignment with a computational effort of at most $O(n^3)$. (Matsui, Tamura, and Ikebe [1991]).

3.38 Assignment Using Choice Lists Giving numerical measures for preferences is hard, it is more natural for preferences to be expressed by choice lists without actual numerical measures. Consider a situation involving people $P_1, \ldots, P_n$ and items $x_1, \ldots, x_n$. Each person gives his/her *choice list* which is the list of the items in decreasing order of preference.

A choice list is a *linear ordering* if each item in the list is strictly preferred over those appearing later (example: x_3, x_2, x_1, x_4; here x_3 is strictly preferred over x_2, etc.) It is a *weak preference ordering* if some set of consecutive items in the list are considered equal in the individuals choice. We will enclose these within brackets (example: $x_4, (x_2, x_3), x_1$; here x_4 is strictly preferred over x_2 or x_3, x_2 and x_3 are equally preferred and either of these is strictly preferred over x_1).

(i) If each person gives his/her choice list, but items have no choice lists for people, develop an algorithm for assigning each person

a different item, so that each gets his/her highest favored item as far as possible. Apply this algorithm to the problem in which $n = 4$ and the choice lists are

$$\begin{array}{ll} P_1: & x_4, (x_2, x_3), x_1 \\ P_2: & (x_1, x_3), (x_2, x_4) \\ P_3: & x_3, x_2, x_1, x_4 \\ P_4: & (x_1, x_3, x_4), x_2. \end{array}$$

(ii) Suppose each person may not list all items in his/her choice list, or the number of people and items may not be equal. Modify the algorithm developed above to assign at most one item per person, so that as many people as possible are assigned their highest favored items as far as possible.

(iii) Consider a case with n people, n items again. Each person gives his/her choice list for items. Also, each item gives its choice list of persons. With respect to these choice lists, an assignment a of items to people is said to be a *stable assignment* if there exists no pair (P_i, x_j) without an allocation in a such that both P_i and x_j prefer each other to their partners in a. Develop an algorithm for finding a stable assignment of items to people.

(Wilson [1977], Gale and Shapley [1962])

3.39 Stable Assignment Problem A group consists of boys, $b_1, \ldots, b_n$; and girls, $g_1 \ldots, g_n$. Each person lists the persons of the other sex in the order in which he/she prefers them, this is called that person's choice list. An assignment of boys to girls, a, is said to be a stable assignment with respect to these choice lists, if there is no pair (b_i, g_j) without an allocation in a, such that both b_i and g_j prefer each other to their partners in a. Gale and Shapley [1962] proposed the following algorithm for finding a stable assignment.

"To start, let each girl propose to her favorite boy. Each boy who receives more than one proposal rejects all but his favorite from among those proposed to him. However, he does not accept her yet, but keeps

her on a string to allow for the possibility that someone better may come along later.

We are now ready for the second stage. Those girls who were rejected now propose to their second choices. Each boy receiving proposals chooses his favorite from the group consisting of the new proposers and the girl on his string, it any. He rejects all the rest and again keeps the favorite in suspense.

We proceed in the same manner. Those who are rejected at the second stage propose to their next choices, and the boys again reject all but the best proposals they have had so far. As soon as the last boy gets his proposal the "courtship" is declared over, and each boy is now required to accept the girl on his string." Remember that this algorithm terminates as soon as every boy receives at least one proposal.

(i) Prove that this algorithm terminates with a stable assignment.

(ii) In the assignment obtained under this method, prove that at most one girl ends up with her last choice as a partner.

(iii) Prove that this algorithm terminates after at most $n^{\mathbf{2}} - 2n + 2$ stages.

(iv) Apply this algorithm when $n = 5$, and the choice lists are:

$b_1 : g_4, g_3, g_2, g_1, g_5,$	$g_1 : b_1, b_2, b_3, b_4, b_5$
$b_2 : g_3, g_2, g_1, g_5, g_4,$	$g_2 : b_4, b_1, b_2, b_3, b_5$
$b_3 : g_2, g_1, g_5, g_4, g_3,$	$g_3 : b_3, b_4, b_1, b_2, b_5$
$b_4 : g_1, g_5, g_4, g_3, g_2,$	$g_4 : b_2, b_3, b_4, b_1, b_5$
b_5 : arbitrary	$g_5 : b_1, b_2, b_3, b_4, b_5$

(v) Verify that the algorithm described above goes through exactly $n^{\mathbf{2}} - 2n + 2$ stages before termination when the choice lists are

as specified below.

$b_1 : g_{n-1}, g_{n-2}, \ldots, g_1, g_n,$	$g_1 : b_1, b_2, \ldots, b_{n-1}; b_n$
$b_2 : g_{n-2}, g_{n-3}, \ldots, g_1, g_n, g_{n-1}$	$g_2 : b_{n-1}, b_1, b_2, \ldots, b_{n-2}, b_n$
$\vdots$	$g_3 : b_{n-2}, b_{n-1}, b_1, \ldots, b_{n-3}; b_n$
$b_{n-1} : g_1, g_n, g_{n-1}, \ldots, g_2$	$\vdots$
$b_n :$ arbitrary	$g_{n-1} : b_2, b_3, \ldots, b_{n-1}, b_1; b_n$
	$g_n : b_1, b_2, \ldots, b_{n-1}, b_n$

(vi) If the method described above goes through the upper bound of $n^{\mathbf{2}}-2n+2$ stages before termination, prove that one girl must get her last choice and the other girls must get their second to last choices in the assignment obtained. Also, in this case prove that the boy who received the last proposal must be the last choice of all the girls. Also in this case prove that each boy, except possibly the last one to receive a proposal, must get his first choice.

(vii) Show that the upper bound on the number of proposals made in the above algorithm before termination is $n+(n-1)^{\mathbf{2}} = n^{\mathbf{2}}-n+1$, and that this is attained in the problem with the following data, even though the method does not go through the upper bound on the number of stages in this problem.

$n = 4$ and the choice lists are given below

$b_1 : g_3, g_2, g_1, g_4$	$g_1 : b_1, b_3, b_2, b_4$
$b_2 : g_1, g_4, g_2, g_3$	$g_2 : b_2, b_1, b_3, b_4$
$b_3 : g_2, g_1, g_3, g_4,$	$g_3 : b_2, b_3, b_1, b_4$
$b_4 :$ arbitrary	$g_4 : b_1, b_3, b_2, b_4$

(viii) In the same manner consider the following choice lists.

$b_1 : g_3, g_4, g_5, \ldots, g_1, g_2,$	$g_1 : b_{n-1}, b_1, b_2, b_3, \ldots, b_{n-2}, b_n$
$b_2 : g_4, g_5, g_6, \ldots, g_2, g_3,$	$g_2 : b_1, b_2, b_3, b_4, \ldots, b_{n-1}, b_n$
$b_3 : g_5, g_6, g_7, \ldots, g_3, g_4,$	$g_3 : b_2, b_3, b_4, b_5 \ldots, b_1, b_n$
$\vdots$	$\vdots$
$b_{n-1} : g_2, g_3, g_4, \ldots, g_n, g_1,$	$g_{n-1} : b_{n-2}, b_{n-1}, b_1, b_2, \ldots, b_{n-3}, b_n$
$b_n : g_1, g_2, g_3, \ldots, g_{n-1}, g_n,$	$g_n : b_{n-1}, b_1, b_2, b_3, \ldots, b_{n-2}, b_n$

For these lists prove the following: (1) in stage 1, all girls propose and the only girl whose proposal is rejected is g_1. (2) At every stage at most one proposal is made. (3) At stage $1 < i < n^2 - 2n + 2$, girl g_r will make her tth proposal to the boy b_s where r, s, t can be expressed in terms of i and n as

$$r = \begin{cases} (i-1) \bmod n & \text{if } (i-1) \bmod n \neq 0 \\ n & \text{otherwise} \end{cases}$$
$$t = 1 + \lceil (i-1)/n \rceil, \quad \text{and}$$

$$s = \begin{cases} t + r - 2 & \text{for } t + r - 2 < n \\ t + r - 1 - n & \text{otherwise.} \end{cases}$$

This proposal is accepted and a girl $g_{r'}$ is jilted by boy b_s where $r' = r + 1$, if $r \neq n$; 1 otherwise. (4) At stage $n^2 - 2n + 2$, g_1 makes her nth proposal to b_n and that proposal is accepted. (5) This takes $n^2 - 2n + 2$ stages.

(Itoga [1978], Kapur and Krishnamoorty [1985])

3.40 Consider the problem discussed in Exercise 3.39. Each person lists all the persons of the other sex in decreasing order of his/her preference. Show that every stable assignment $x = (x_{ij})$ of boys to girls is an extreme point of the set of feasible solutions of the following system and vice versa.

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= 1, i = 1 \text{ to } n \\ \sum_{i=1}^{n} x_{ij} &= 1, j = 1 \text{ to } n \\ x_{pq} + \sum (x_{pj} &: \text{ over girls } j \text{ preferred over girl } q \text{ by boy } p) + \\ \sum (x_{iq} &: \text{ over boys } i \text{ preferred over boy } p \text{ by girl } q) \geqq 1, p, q = 1 \text{ to } n \\ x_{ij} &\geqq 0, \text{ for all } i, j. \end{aligned}$$

(Vande Vate [1989])

3.41 The following (in Figure 3.10) is a minimum cost network flow model for a 3-period production planning problem with inventory and backorder bounds. k_1, k_2, k_3 are production capacities; and d_1, d_2, d_3 are the demands in the three periods. s_1, s_2 are respectively the inventory limits from periods 1 to 2, and 2 to 3; b_1, b_2 are the backorder bounds in periods 1 and 2. These data provide the capacities for arcs in the following network, all lower bounds are 0. The cost data is not shown. Transform this problem into an equivalent uncapacitated transportation problem. (Evans [1985]).

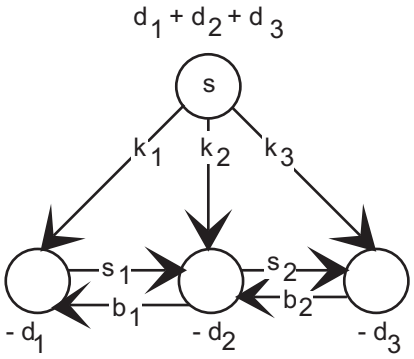

Figure 3.10:

3.42 Allocation of Candidates to Jobs There are m jobs, and n candidates. For each candidate, we are given a nonempty subset of jobs (called *candidate's job-set*) to which that candidate could be allocated. Each candidate must be allocated to exactly one job in his/her job-set, but each job can be allotted any number of candidates. Let $r_1 \ldots, r_m$ be the number of candidates allocated to various jobs in an allocation. For the ith job, we are given a monotonic strictly decreasing cost function $f_i(r_i), i = 1$ to m. Arrange the costs $f_1(r_1), \ldots, f_m(r_m)$ in non-increasing order. The resulting vector is called the *ranked cost vector* associated with the allocation. It is required to find an allocation which corresponds to a lexico minimum of the ranked cost vector subject to the conditions discussed above.

As a numerical example consider $m = 3, n = 5$, and the job-sets of candidates 1 to 5 to be $\{1\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{3\}$ respectively. Let

the cost functions $f_i(r_i)$ for jobs $i = 1$ to 3 be $\frac{1}{r_1}, \frac{2}{r_2}, \frac{5}{r_3}$ respectively. Consider the two allocations listed below

Alloc.	Job allocated to cand. j $j=1$	2	3	4	5	r_1	r_2	r_3	$f_1(r_1)$	$f_2(r_2)$	$f_3(r_3)$
a_1	1	2	3	3	3	1	1	3	1	2	$\frac{5}{3}$
a_2	1	1	1	2	3	3	1	1	$\frac{1}{3}$	2	5

The ranked cost vectors associated with the allocations a_1, a_2 are $(2, \frac{5}{3}, 1), (5, 2, \frac{1}{3})$ respectively, and hence by the lexico minimum criterion a_1 is better than a_2. In fact in this numerical example it can be shown that a_1 is an optimum allocation for the problem.

(i) Model this problem using a bipartite network. Show that each allocation of candidates to jobs corresponds to a subnetwork of this bipartite network.

(ii) Given an allocation a, define an alternating path wrt it, to be a path beginning and terminating with job nodes, with successive edges having a common node and alternately belonging/not belonging to a.

Given a feasible allocation a and an alternating path $\mathcal{P}$ wrt it, define the operation of rematching a using $\mathcal{P}$ to be that of obtaining an allocation a^1 by (A) including in a allocations corresponding to edges in $\mathcal{P}$ which are not already in a, and (B) deleting all allocations common to a and $\mathcal{P}$. Show that the resulting allocation a^1 will be feasible.

(iii) Define an alternating path $\mathcal{P}$ wrt a feasible allocation a, to be an improving alternating path wrt a, if rematching a using $\mathcal{P}$ leads to an allocation a^1 whose ranked cost vector is lexico smaller than that of a.

Prove that a feasible allocation corresponds to a lexico minimum ranked cost vector iff there exists no improving alternating path wrt it.

(iv) Develop an algorithm which finds an optimum allocation by stepwise improvement of a feasible allocation.

(v) Consider the following instance that arises in the army. $m = 2$, the jobs are driving and cooking. $n = 3600$ candidates who are reservists. Of the 3600; 900 can only drive, 2100 can only cook, and the remaining 600 can do both.

The army needs 1200 man months of driving time and 1800 man months of cooking time annually. Let r_1, r_2 be the number of candidates allocated to driving, cooking respectively. Then $f_1(r_1) = 1200/r_1$, this is the months of army service per annum for drivers. Likewise $f_2(r_2) = 1800/r_2$ is the months of army service for cooks. Beginning with the feasible allocation with $r_1 = 900, r_2 = 2700$, obtain an optimum allocation.

(Cramer and Pollatschek [1979])

3.43 A problem of interest in core management of pressurized water nuclear reactors is to find an optimal allocation of the given fuel assemblies which may be differing on their burn-up states, to particular locations in the reactor, such that each full assembly is assigned to a location and vice versa, under a constraint on the power distribution form factor. This gives rise to an assignment problem subject to one additional constraint, of the following form

$$\begin{aligned}
\text{minimize } z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \quad & x_{ij} \\
\text{subject to} \sum_{j=1}^{n} \quad & x_{ij} \quad = 1, i = 1 \text{ to } n \\
\sum_{i=1}^{n} \quad & x_{ij} \quad = 1, j = 1 \text{ to } n \\
& x_{ij} \quad \geqq 0, \text{ for all } \ i, j \qquad (3.23) \\
\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \quad & x_{ij} \quad \leqq b \\
\text{and} \quad & x_{ij} \quad \text{integer for all } i, j
\end{aligned}$$

where, without any loss of generality, we can assume that $c_{ij}, d_{ij} \geqq 0$ for all i, j and $b > 0$. (3.23) is an integer program. Show that it is an NP-hard problem. Develop a practically efficient method for solving it, based on Lagrangian relaxation and assignment ranking. Apply this method to solve the numerical problem with the following data: $n = 4, b = 26$.

$$(c_{ij}) = \begin{pmatrix} 1 & 5 & 7 & 4 \\ 9 & 7 & 9 & 9 \\ 5 & 5 & 11 & 5 \\ 8 & 7 & 8 & 5 \end{pmatrix}, (d_{ij}) = \begin{pmatrix} 9 & 6 & 4 & 8 \\ 7 & 5 & 9 & 6 \\ 5 & 8 & 7 & 11 \\ 6 & 3 & 2 & 10 \end{pmatrix}$$

(Aggarwal [1985], Gupta and Sharma [1981])

3.44 In some transportation models, the costs of different shipments are borne by different individuals, and the sum of all the costs is not a good objective to minimize. In these models, a better objective is to minimize the maximum cost incurred by any single individual, called the bottleneck objective function. This leads to:

$$\begin{aligned} \text{minimize } (\max \{c_{ij} \quad &: \quad (i,j) \text{ such that } x_{ij} > 0\}) \\ \text{subject to} \sum_{j=1}^{n} x_{ij} \quad &\leqq \quad a_i, i = 1 \text{ to } m \\ \sum_{i=1}^{n} x_{ij} \quad &\geqq \quad b_j, j = 1 \text{ to } n \\ x_{ij} \quad &\geqq \quad 0, \text{ for all } \ i, j \end{aligned} \tag{3.24}$$

(i) Develop an efficient algorithm for this problem. Apply this algorithm on the numerical problem with the following data.

	c_{ij}							
$j =$	1	2	3	4	5	6	7	a_i
$i = 1$	10	6	4	8	8	10	0	27
2	4	2	2	2	8	1	0	26
3	14	12	9	4	9	3	0	26
4	4	7	6	9	1	4	0	27
b_j	19	17	17	15	10	8	20	

(ii) A commonly encountered constraint in distribution problems is the requirement that the entire demand of each customer must be supplied from a single source or supplier. Consider the problem (3.24) with such an additional constraint. Develop a practical approach for solving this combined problem. Apply this approach on the numerical problem with the data given above.

(Nagalhout and Thompson [1984])

3.45 Classroom Allocation to Courses A big university has a total of L classrooms of varying capacities (the capacity of a room is the number of students it can accommodate) spread over various buildings. In a term the university is planning to offer a total of N courses. Each course meets for a total of 2, 3, or 4 hours per week (this is known as the *number of credit hours for the course*) and this may all be in a single session on one day of the week, or split into several sessions (each of the sessions are either one, or one and a half, or three hours in length) over several days of the week. For example a 3 credit hour course may meet in one session of three hours say from 7-10 PM on one day; or in three hourly sessions say from 2 to 3 PM on three different days of the week; etc. For each course the number of credit hours, and the sessions in which they will meet (i.e. on which days the course meets and the beginning and ending time of the session on each day) has already been determined and all this information is given. The expected enrollment in each course is available, and using this and other information, the university has compiled a subset $\mathbf{X}_i$ of classrooms in which they would prefer to hold course i, $i = 1$ to N.

The constraints in allotting classrooms to courses are the following: At any point of time there can be only one course allotted to a classroom; also, if a course consists of several sessions during a week, all the sessions must meet in the same classroom. It is required to allot classrooms to courses from their preferred subset, in such a way that the number of courses for which allocations are made is maximized (i.e., the number of courses for which you are unable to allot a classroom from its preferred subset, should be as low as possible). Model this problem.

3.46 Bottleneck Assignment Problem with Node Weights Let $a_1, \ldots, a_n$; $b_1, \ldots, b_n$ be weights associated with the rows; and columns of the n x n assignment array corresponding to a bipartite network G. For each $i = 1$ to n, let $\mathbf{S}_i = \{j : (i; j)$ is an edge in G$\}$, and correspondingly for each $j = 1$ to n let $\mathbf{P}_j = \{i\colon\ j \in \mathbf{S}_i\}$. Consider the following bottleneck assignment problem: find $x = (x_{ij})$ to

$$\begin{aligned} \text{minimize (maximum } \{(a_i + b_j) \quad & x_{ij} \quad : i, j = 1 \text{ to } n\}) \\ \text{subject to} \sum_{j \in \mathbf{S}_i} \quad & x_{ij} \quad = 1, i = 1 \text{ to } m \\ \sum_{i \in \mathbf{P}_j} \quad & x_{ij} \quad = 1, j = 1 \text{ to } n \qquad (3.25) \\ & x_{ij} \quad = 0 \text{ or } 1, \text{ for all } \ i, j \\ & x_{ij} \quad = 0, \text{ for } \ j \notin \mathbf{S}_i \end{aligned}$$

(i) Show that this is a special case of the bottleneck assignment problem discussed in Section 3.1.5 in which $c_{ij} = a_i + b_j$ for $j \in \mathbf{S}_i, \infty$ for $j \notin \mathbf{S}_i$.

(ii) Consider the special case of (3.25) in which all $\mathbf{S}_i$ and $\mathbf{P}_j$ are $\{1, \ldots, n\}$. In this case order the $a_i s$ in nonincreasing order, and $b_j s$ in nondecreasing order. Suppose these orders are $a_{i_1} \geqq a_{i_2} \geqq \ldots \geqq a_{i_n}$ and $b_{j_1} \leqq b_{j_2} \leqq \ldots \leqq b_{j_n}$. In this case, show that the assignment $\{(i_1, j_1), (i_2, j_2), \ldots, (i_n, j_n)\}$ is an optimum solution of (3.25).

(iii) Consider cell (i, j) of the $n \times n$ assignment array *admissible* if $j \in \mathbf{S}_i$, *inadmissible* otherwise. Develop a dual algorithm for (3.25) which starts with an infeasible assignment (i.e., one containing some allocations in inadmissible cells), while maintaining optimality conditions, and tries to reduce the infeasibility in each iteration.

(iv) Let $\bar{x} = (\bar{x}_{ij})$ be a feasible assignment to (3.25), and $\mathbf{M}= \{(i; j); \bar{x}_{ij} = 1\}$ be the corresponding matching in G. The cell (r, s) is said to be a bottleneck cell (corresponding to a bottleneck edge in G) with respect to $\bar{x}$ if $\bar{x}_{rs} = 1$ and $a_r + b_s = \max\{a_i + b_j; (i, j)$ such that $\bar{x}_{ij} = 1\}$.

A decreasing alternating path from r to s is G with respect to $\mathbf{M}$ or $\bar{x}$ is a path from r to s satisfying the following properties: (1) it does not contain $(r;s)$, (2) edges in it are alternately in $\mathbf{M}$ and not in $\mathbf{M}$, (3) for every edge $(i;j)$ on it which is not from $\mathbf{M}$, we have $a_i + b_j < a_r + b_s$. When $(r;s)$ is a bottleneck cell with respect to $\bar{x}$ and there is no decreasing alternating path from r to s, prove that $\bar{x}$ is an optimum solution for (3.25)

If a decreasing alternating path exists from r to s, define $\hat{x} = (\hat{x}_{ij})$ by

$$\hat{x}_{ij} = \begin{cases} 1 - \bar{x}_{ij} & \text{for all } (i,j) \text{ on the path.} \\ 0, & \text{for } (i,j) = (r,s). \\ \bar{x}_{ij} & \text{for all } (i,j) \neq (r,s) \text{ not on the path.} \end{cases}$$

Then show that $\hat{x}$ is a better assignment for (3.25) than $\bar{x}$. Using these results develop a primal algorithm for (3.25) that starts with and maintains feasible assignments and strictly decreases the objective value in each iteration by finding a decreasing alternating path and using it as above. Discuss an initialization phase for this algorithm if an initial feasible assignment is not available.

(Lawler [1976 of Chapter 1], Carraresi and Gallo [1984])

3.47 The Bus Driver Rostering Problem Consider an m day time horizon, with n shifts in each day. Each shift on each day is to be manned by a single bus driver. w_{pj} denotes the weight of the jth shift on day $p, j = 1$ to $n, p = 1$ to m, this may be the time duration or some other measure of the workload for that shift. There are n drivers. This problem is concerned with the assignment of drivers to shifts over the days of the horizon so that each driver receives an even balance of each type of shift. It can be formulated as a bottleneck problem, to minimize the maximum total weight of the shifts assigned to a driver.

Represent the jth shift of pth day by a node numbered (p, j). This node has weight w_{pj}. Let $\mathbf{S}_p(j)$ be the set of shifts in day $p+1$ that can be assigned to a driver who has been assigned shift j in day p by union rules or other work constraints. Draw an arc from node (p, j) to

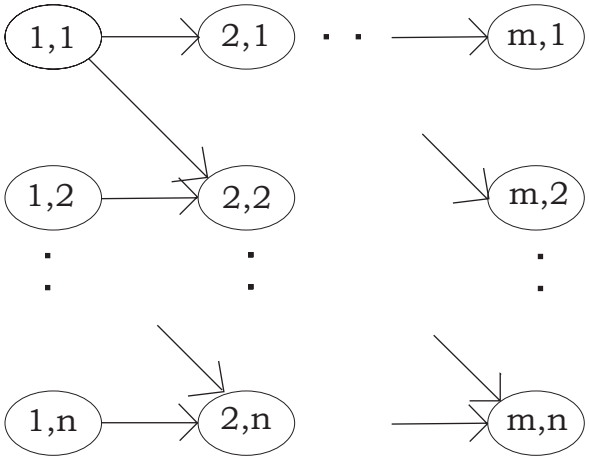

Figure 3.11:

each node $(p+1, i)$ for $i \in \mathbf{S}_p(j)$. Let $\mathbf{P}_p(i)$ be the set of shifts in day p which could have been assigned in day p to a driver to whom shift i has been assigned in day $p+1$. The results is a layered acyclic network of the form shown in Figure 3.11.

A feasible work assignment to a single worker corresponds to a chain from a node in the first layer to a node in the mth layer in this network. Its total workload being given by the sum of the weights of the nodes on the chain. The problem of finding work assignments to all the n workers, such that the maximum workload is minimized, can be formulated as the problem of finding, in the network in Figure 3.11, n node disjoint chains from layer 1 to layer m so as to minimize the longest among these chains. Give a mathematical formulation of this problem using 0-1 variables.

Prove that a feasible solution to this problem exists iff for each $p = 1$ to $m-1$, the constraints

$$\begin{array}{l} \sum_{j \in \mathbf{S}_p(i)} x_{ij}^p = 1, i = 1 \text{ to } n \\ \sum_{i \in \mathbf{P}_p(j)} x_{ij}^p = 1, j = 1 \text{ to } n \\ x_{ij}^p = 0 \text{ or } 1, \text{ for all } \ i, j \end{array}$$

are feasible. If $m = 2$, show that this problem reduces to a bottleneck assignment problem of the form discussed in Exercise 3.46.

When $m > 2$, develop a heuristic algorithm for this problem based on the bottleneck assignment problem discussed in Exercise 3.46. (Carraresi and Gallo [1984])

3.48 A Vehicle Scheduling Application This application is concerned with the optimal assignment of vehicles to time-tabled trips so that each trip is carried out by one vehicle subject to some constraints. A trip is defined by a quadruple (τ_i, l_i, o_i, d_i) where

τ_i = scheduled start time of the ith trip.

l_i = duration or length of the ith trip.

o_i = the origin or the start terminal for the ith trip.

d_i = the destination terminal for the ith trip.

Suppose the time-table consists of n trips, denoted by $\zeta_1, \ldots, \zeta_n$. In addition to these regular trips, deadheading trips are allowed between terminals. δ_{ij} denotes the duration of a deadheading trip from d_i to $o_j, i, j = 1$ to n, $i \neq j$. The ordered pair of trips (ζ_i, ζ_j) is said to be a *compatible pair* if $\tau_i + l_i + \delta_{ij} + \varepsilon \overset{\leq}{=} \tau_j$, where $\varepsilon \overset{\geq}{=} 0$ is a tolerance parameter to absorb possible delays. If (ζ_i, ζ_j) is compatible, it is clearly feasible to have trips ζ_i, ζ_j operated in sequence by the same vehicle. A *vehicle duty* is a sequence $\vartheta = (\zeta_{i_1}, \zeta_{i_2}, \ldots, \zeta_{i_r})$ of trips satisfying the property that every consecutive pair of trips in this sequence is compatible, all these trips can be operated by the same vehicle. A feasible vehicle schedule is a family $\{\vartheta_1, \ldots, \vartheta_g\}$ of vehicle duties such that each trip $\zeta_1, \ldots, \zeta_n$ belongs to exactly one $\vartheta_h, h = 1$ to g.

Construct a bipartite network G=$(\mathbf{S}, \mathbf{T}, \mathcal{A})$ where $\mathbf{S}=\{s_1, \ldots, s_n\}$, $\mathbf{T}=\{t_1, \ldots, t_n\}$ and $\mathcal{A} = \{(s_i, t_j)$; for all i, j such that (ζ_i, ζ_j) is a compatible pair$\}$.

(i) A vehicle duty $\vartheta = (\zeta_{i_1}, \zeta_{i_2}, \ldots, \zeta_{i_r})$ can be represented in G by the set of arcs $\{(s_{i_1}, t_{i_2}), (s_{i_2}, t_{i_3}), \ldots, (s_{i_{r-1}}, t_{i_r})\}$ which can be verified to be a matching in G. A vehicle duty containing only one trip $\{\zeta_j\}$ can be represented by leaving both nodes s_j, t_j as exposed nodes in G. Using this, show that there is a one-to-one correspondence between vehicle schedules and matchings in G.

Consider the example in which there are 4 terminals a,b,c,d, and 5 trips with the following data: $\varepsilon = 0$

Trip	τ_i	l_i (minutes)	o_i	d_i
ζ_1	7:10 a.m.	20	a	b
ζ_2	7:20 a.m.	20	c	d
ζ_3	7:40 a.m.	25	b	a
ζ_4	8:00 a.m.	30	d	c
ζ_5	8:35 a.m.	30	c	d

$(\delta_{ij}) =$

		to a	b	c	d
$from$	a	0	15	20	20
	b	15	0	25	10
	c	20	25	0	15
	d	20	10	15	0

.

Construct the network G for this example and obtain the vehicle schedule corresponding to the matching $\{(s_1, t_3), (s_2, t_4), (s_4, t_5)\}$ in it.

(ii) Define the set of arcs $\mathcal{A}^* = \{(s_i, t_j); (s_i, t_j) \notin \mathcal{A}$ defined above, and either $i = j$ or $\tau_i \stackrel{\geq}{=} \tau_j + l_j + \delta_{ji}\}$. Let G′ denote the network obtained by augmenting G with the additional arcs $\mathcal{A}^*$, and introducing a unit exogenous supply at each $s_i, i = 1$ to n (these are now source nodes), and a unit demand at each $t_j, j = 1$ to n, and lower bound of 0 and capacity of 1 on each arc in $\mathcal{A} \cup \mathcal{A}^*$.

If $f = (f_{s_i,t_j})$ is an integer feasible flow vector in G′, it defines an assignment or perfect matching in G′. If $f_{s_i,t_j} = 1$ then the arc (s_i, t_j) is in the perfect matching; and in addition if $(s_i, t_j) \in \mathcal{A}$ make the trips ζ_i and ζ_j belong in that order to the same vehicle duty, and if $(s_i, t_j) \in \mathcal{A}^*$ then make ζ_i and ζ_j to be the last and first trips respectively of a vehicle duty (in this case they may or may not belong to the same vehicle duty). Also, if $f_{s_i,t_i} = 1$ then make ζ_i as the only trip of a vehicle duty. Under this convention, show that every feasible integer flow vector in G′ (or a perfect matching in G′) corresponds to a feasible vehicle schedule where

the number of vehicles used is equal to the number of units of flow (or arcs in the perfect matching) on arcs from the set $\mathcal{A}^*$ and vice versa. Draw the network G′ for the example given in (i) and illustrate this point.

(iii) Now consider the case in which all the vehicles are housed in one depot (common in small size transit companies, or those in which the service area is partitioned into zones with each zone assigned to one single depot). Assume that there are no constraints other than compatibility and that all vehicles are of the same type.

Formulate the problem of finding a vehicle schedule to minimize the fleet size as an assignment problem.

If it is required to find a vehicle schedule that minimizes the operational costs (these include deadheading travel costs, and cost of any idle time between the end of a trip and the starting of the next), formulate the problem of finding it as an assignment problem.

Also discuss how one can find a vehicle schedule that minimizes a combination of the above two costs, or one that minimizes the operational cost subject to the constraint that the fleet size is the minimum.

(iv) Now consider the case in which there are multiple depots, each with a given capacity for the number of vehicles it can house. Formulate this multiple depot problem as a 0-1 integer program and discuss heuristic approaches to obtain reasonable solutions for it.

(Carraresi and Gallo [1984])

3.49 Let G=(**S**,**T**; $\mathcal{A}$) be a bipartite network with $|\mathbf{S}| = |\mathbf{T}|$. $\mathbf{E} \subset \mathcal{A}$ is a specified subset of edges in G. It is required to determine whether there exists a perfect matching in G containing at most r edges from **E**. Formulate this as a minimum cost network flow problem.

3.50 Let G=($\mathcal{N}, \mathcal{A}$) be a directed acyclic network. Define a *chain cover* for G to be a node disjoint union of chains in G (degenerate

chains consisting of a single node by itself being admissible) which contains all the nodes in $\mathcal{N}$. The size of a chain cover for G is defined to be the number of chains in it. The *chain covering number* for G is the size of a minimum size chain cover.

(i) Show that a chain cover for G is of minimum size iff it contains the maximum number of arcs among all chain covers.

(ii) Based on the result in (i), a greedy approach for finding a minimum size chain cover in G, consists of successively determining longest chains. Show that this greedy approach fails to give a minimum size chain cover in the following network in Figure 3.12 (minimum size is 4, greedy approach gives a cover of size 5).

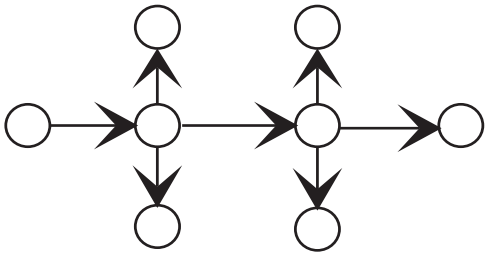

Figure 3.12:

(iii) Obtain a new network $G' = (\mathcal{N}', \mathcal{A}')$ from G by the following procedure. Replace each node i in $\mathcal{N}$ by a pair of nodes i' and i''. Each are incident into i in G becomes an arc incident into i', and each arc incident out of i becomes an arc incident out of i'', in G′. So, every node in G′ has either zero in-degree or zero out-degree. Show that G′ is bipartite.

(iv) Show that the transformation in (iii) converts the arcs in any collection of node-disjoint chains in G into a matching in G′. Using the fact that G is acyclic, show that every matchings in G′ corresponds to a unique set of arcs of some node-disjoint collection of chains in G.

From these results show that a minimum size chain cover in G can be obtained by finding a maximum cardinality matching in G′.

(Boesch and Gimpel [1977])

3.51 Let G=$(\mathcal{N}, \mathcal{A}, \breve{s}, \breve{t})$ be an acyclic network with $\breve{s}$ as the source node with in-degree zero, and $\breve{t}$ as the sink node with out-degree zero. An $\breve{s} - \breve{t}$ chain cover for the nodes of G is a collection of chains from $\breve{s}$ to $\breve{t}$ in G such that each node $i \in \mathcal{N}$ is contained on at least one chain in the collection, its size is defined to be the number of chains in it. A minimum $\breve{s} - \breve{t}$ chain cover for nodes in G is one of the smallest possible size, let α denote its size.

A pair of distinct nodes $i, j \in \mathcal{N}$ are said to be incomparable in G, if there exists no chain from i to j or from j to i in G. A subset $\mathbf{X} \subset \mathbf{N}$ is an incomparable node set in G if every pair of distinct nodes in $\mathbf{X}$ is incomparable. Let β be the cardinality of a maximum cardinality incomparable node set in G.

Transform G into G′ by splitting each node $i \in \mathcal{N} \backslash \{\breve{s}, \breve{t}\}$ into two nodes i', i'' and adding the arc (i', i''), all arcs incident into (out of) i in $\mathcal{A}$ will be incident into i' (out of i'') in G′. Define lower bounds for flows on arcs in G′ to be 1 for all arcs of the form (i', i'') and 0 for all other arcs, and capacities to be $+\infty$ on all the arcs. Find a minimum value feasible flow vector $\bar{f}$ in G′, let its value be $\bar{v}$. Then prove that $\bar{v} = \alpha = \beta$. Use this to develop an algorithm for finding a minimum $\breve{s} - \breve{t}$ chain cover for nodes in G.

This provides a method based on the minimum value flow problem for finding α and β. An alternate method for the same based on maximum cardinality matching is discussed in Section 3.1.4. (Ntafos and Hakimi [1979])

3.52 Let G =$(\mathcal{N}, \mathcal{A}, \breve{s}, \breve{t})$ be an acyclic network with $\breve{s}$ = the source node with in-degree zero, and $\breve{t}$ = the sink node with out-degree zero. An $\breve{s} - \breve{t}$ chain cover for arcs in G is a collection of chains from $\breve{s}$ to $\breve{t}$ in G that contains all the arcs in $\mathcal{A}$, its size is the number of chains in it. Let ϑ be the size of a minimum size $\breve{s} - \breve{t}$ chain cover for arcs in G.

Two arcs e_1, e_2 in G are said to be incomparable if there is no chain from $\breve{s}$ to $\breve{t}$ containing both of them. An incomparable arc set in G

is a set of arcs every pair of which are incomparable. Let δ be the cardinality of a maximum cardinality incomparable arc set in G.

Prove that $\vartheta = \delta$. As in Exercise 3.51, show that a minimum size $\breve{s} - \breve{t}$ chain cover for arcs in G can be found by the minimum value flow method by applying it directly on G by placing a lower bound of one unit on each arc in G.

Let $\{e_1, \ldots, e_{|\mathcal{A}|}\}$ be the arcs in $\mathcal{A}$, and $\mathcal{N}'' = \{1, \ldots, |\mathcal{A}|\}$ with $j \in \mathcal{N}''$ corresponding to the arc e_j in $\mathcal{A}$. Define $\mathcal{A}'' = \{(i, j) : i, j \in \mathcal{N}''$, and there is a chain in G from the head node of e_i to the tail node of $e_j\}$. Let $G'' = (\mathcal{N}'', \mathcal{A}'')$.

Show that G'' is also acyclic. Show that a minimum size $\breve{s} - \breve{t}$ chain cover for arcs in G, and a maximum cardinality incomparable arc set in G, can both be found by applying the maximum matching method discussed in Section 3.1.1 to the network G''. (Ntafos and Hakimi [1979])

3.53 Let $G = (\mathcal{N}, \mathcal{A})$ be a directed connected network. Define a chain cover for the nodes in G to be a set of chains (not necessarily simple or even elementary) such that each node in $\mathcal{N}$ is contained on at least one chain in the set. Let $P_N(G)$ denote the cardinality of a minimum cardinality chain cover for nodes in G.

A node $i \in \mathcal{N}$ reaches (is reached from) a node j if there is a chain from i to j (from j to i) in G. A set $\mathbf{X} \subset \mathcal{N}$ of nodes in an incomparable node set if there is no chain in G between any pair of distinct nodes in $\mathbf{X}$. Let $I_N(G)$ denote the cardinality of a maximum cardinality incomparable node set in G.

Since any two vertices in a strongly connected component are mutually reachable, it follows that there always exists a chain containing all the nodes in a strongly connected component. Hence, for the purpose of finding a minimum cardinality chain cover for nodes, each strongly connected component in G can be replaced by a single new vertex.

Find all the strongly connected components (SCCs) in G; suppose there are r of them. Construct a new network G' with node set $\{1, \ldots, r\}$, in which node i corresponds to the ith SCC in G. Introduce an arc (i, j) in G' if there exists a chain from some node in the ith SCC in G, to some node in the jth SCC. Show that G' is acyclic. Prove that $P_N(G) = P_N(G')$ and that $I_N(G) = I_N(G')$. Using this prove that

$P_N(\mathrm{G}) = I_N(\mathrm{G})$. Discuss an efficient algorithm for finding a minimum cardinality chain cover for nodes, and a maximum cardinality incomparable node set in G, using the results in Exercise 3.52. (Ntafos and Hakimi [1979])

3.54 Let $\mathrm{G} = (\mathcal{N}, \mathcal{A})$ be a connected directed network. Define chain covers for arcs in G, reachability among arcs, incomparable arc sets, the same way it was done for nodes in Exercise 3.53. Let $P_A(\mathrm{G})$ be the cardinality of a minimum cardinality chain cover for arcs, and let $I_A(\mathrm{G})$ be the cardinality of a maximum cardinality incomparable arc set in G. Prove that $P_A(\mathrm{G}) = I_A(\mathrm{G})$, and discuss an efficient method for finding a minimum cardinality chain cover for arcs, and a maximum cardinality incomparable arc set in G, using the results in Exercises 3.52, 3.53. (Ntafos and Hakimi [1979])

Comment 3.4 The Hungarian method for the assignment problem, a combinatorial procedure, was developed by H. Kuhn [1955]. Since the main ideas underlying the method came from Egerváry's proof of the König-Egerváry theorem concerning bipartite graphs, he called it the "Hungarian method." The primal-dual setting has made it possible to solve this problem by combinatorial methods through a sequence of maximum value flow problems. The method generalizes directly to the transportation problem, and to minimum cost flow problems in capacitated networks which are not necessarily bipartite (see Chapter 5). The extension of the primal-dual approach to solve minimum cost matching problems in nonbipartite networks required a new technique, namely the characterization of the convex hull of matching incidence vectors by a system of linear inequalities. This was done by J. Edmonds [1965 of Chapter 10], the resulting combinatorial algorithms for matching problems are discussed in Chapter 10.

The $\mathrm{O}(n^3)$ implementation of the Hungarian method is due to Lawler [1976 of Chapter 1].

When the method of Section 5.8.1 for minimum cost flows is specialized to the assignment problem, it leads to a shortest augmenting path method for it. This has been discussed by Derigs and Metz [1986], Jonker and Volgenant [1987], and Tomizawa [1972]. Related methods based on successive shortest chains using a relaxation approach have

been discussed by Dinic and Kronrod [1969], Engquist [1982], and Hung and Rom [1980]. The primal algorithm of Balinski and Gomory [1964] maintains a feasible assignment and reaches an optimum assignment by augmenting flows along negative cost cycles, this method is again based on shortest chain computations.

Several variants of the primal simplex method for solving the assignment problem have been discussed in the literature. Barr, Glover, and Klingman [1977] discuss the finite version based on using strongly feasible partitions (see Chapter 5), and report good computational performance of it. Akgul [1987], Hung [1983], and Roohy-Laleh [1981] discuss polynomial time variants of the primal simplex algorithm for the assignment problem.

Signature methods for the assignment problem have been proposed by Balinski[1985, 1986]. Goldfarb [1985] developed the inductive signature method. Recently signature methods have been generalized to solve transportation problems by Paparrizos [1990].

Computational studies on these different algorithms for the assignment problem by different groups at different times have rated the Hungarian method, relaxation methods, shortest augmenting path methods, as having produced excellent performance. Carpento, Martello, and Toth [1988], and Burkard and Derigs [1980 of Chapter 1] present FORTRAN implementations of assignment algorithms for dense and sparse cases.

Metric and Maybee [1973] present a FORTRAN implementation of the assignment ranking algorithm.

The scaling technique for modifying the primal-dual method into a polynomially bounded algorithm, by applying it on a sequence of better and better approximations of the original problem, was introduced by Edmonds and Karp [1972 of Chapter 2]. However, this technique is mainly of theoretical interest, as the performance of the resulting algorithm in computational tests does not compare favorably with that of other minimum cost flow algorithms discussed in Chapter 5.

Algorithms for stable assignments are discussed in Gale and Shapley [1962], Irving, Leather, and Gusfield [1987], Kapur and Krishnamoorty [1981], and Wilson [1977]. These algorithms are not discussed in the text, but are presented among various exercises given above.

3.8 References

V. AGGARWAL, 1985, "A Lagrangian Relaxation Method for the Constrained Assignment Problem," *COR*, 12, no. 1(97-106).

M. AKGUL, 1987, "A Genuinely Polynomial Primal Simplex Algorithm for the Assignment Problem," Working paper IEOR 87-07, Bilikent University, Ankara, Turkey.

M. AKGUL, 1988, "A Sequential Dual Simplex Algorithm for the Linear Assignment Problem," *OR Letters*, 7(155-158).

M. AKGUL and O. EKIN, June 1990, "A Dual Feasible Forest Algorithm for the Linear Assignment Problem," Research Report IEOR-9011, Bilkent University, Ankara, Turkey.

M. L. BALINSKI, 1985, "Signature Methods for the Assignment Problem," *OR*, 33(527-537).

M. L. BALINSKI, March 1986, "A Competitive (Dual)Simplex Method for the Assignment Problem," *MP*, 34, no. 2(125-141).

M. L. BALINSKI and R. E. GOMORY, 1964, "A Primal Method for the Assignment and Transportation Problems," *MS*, 10(578-593).

R. S. BARR, F. GLOVER and D. KLINGMAN, 1977, "The Alternating Path Basis Algorithm for Assignment Problems" *MP*, 13(1-13).

D. BERTSEKAS, 1981, "A New Algorithm for the Assignment Problem," *MP*, 21(152-171).

F. T. BOESCH and J. F. GIMPEL, April 1977, "Covering the Points of a Digraph With Point-Disjoint Paths and its Application to Code Optimization," *JACM*, 24, no. 2(192-198).

G. CARPANETO, S. MARTELLO and P. TOTH, 1988, "Algorithms and Codes for the Assignment Problem," (193-224) in B. Simeone, et. al. (Eds.), *FORTRAN Codes for Network Optimization, AOR, 13.*

G. CARPANETO and P. TOTH, 1987, "Primal-Dual Algorithms for the Assignment Problem," *DAM*, 18(137-153).

P. CARRARESI and G. GALLO, 1984, "Network Models for Vehicle and Crew Scheduling," *EJOR*, 16(139-151).

P. CARRARESI and G. GALLO, 1984, "A Multi-Level Bottleneck Assignment Approach to the Bus Drivers' Rostering Problem," *EJOR*, 16(163-173).

J. CRAMER and M. POLLATSCHEK, May 1979, "Candidate Job Allocation Problem With a Lexicographic Objective," *MS*, 25, no. 5(466-473).

U. DERIGS and A. METZ, 1986, "An In-Core/Out-of-Core Method for Solving Large Scale
Assignment Problems," *Zeitschrift für Operations Research*, 30, no. 5(A181-A195).

U. DERIGS and U. ZIMMERMANN, 1978, "An Augmenting Path Method for Solving Linear Bottleneck Assignment Problems," *Computing*, 19(285-295).

D. DEWARRA, March 1971, "Construction of School Timetables by Flow Methods," *INFOR*, 9, no. 1(12-22).

E. A. DINIC and M. A. KRONROD, 1969, "An Algorithm for Solution of the Assignment Problem," *Soviet Math. Doklady*, 10(1324-1326).

J. EGERVÁRY, 1931, "Matrixok Kombinatorikus Tulajdonságairól," *Mat. és Fiz. Lapok* , 38(16-28).

M. ENGQUIST, Nov. 1982, "A Successive Shortest Path Algorithm for the Assignment Problem," *INFOR*, 20, no, 4(370-384).

J. R. EVANS, March 1985, "On Equivalent Transportation Models for Production Planning Problems,"*IIE Transactions*, 17, no, 1(102-104).

L. R. FORD and D. R. FULKERSON, 1957, "A Primal-Dual Algorithm for the Capacitated Hitchcock Problem," *NRLQ*, 4(47-54).

D. R. FULKERSON, 1956, "Note On Dilworth's Decomposition Theorem for Partially Ordered Sets," *Proc. Amer. Math. Soc.*, 7(701-702).

D. GALE and L. S. SHAPLEY, 1962, "College Admissions and the Stability of Marriage," *American Math. Monthly*, 69(9-15).

I. GERTSBAKH and H. I. STERN, Jan. - Feb. 1978, "Minimal Resources for Fixed and Variable Job Schedules," *OR*, 26, no. 1(68-85).

F. GLOVER, D. KLINGMAN and N. PHILLIPS, 1984, "An Equivalent Subproblem Relaxation for Improving the Solution of a Class of Transportation Scheduling Problems," *EJOR*, 17(123-124).

D. GOLDFARB, 1985, "Efficient Dual Simplex Methods for the Assignment Problem," *MP*, 33(187-203).

O. GROSS, March 1959, "The Bottleneck Assignment Problem," P-1630, The RAND Corp.

A. GUPTA and J. SHARMA, 1981, "Tree Search Method for Optimal Core Management of Pressurized Water Reactors," *COR*, 8(263-266).

H. HAMACHER, Nov. 1982, "An O($kn^{\mathbf{4}}$) Algorithm for finding the k Best Cuts in a Network," *OR Letters*, 1, no. 5(186-189).

H. HAMACHER, J. C. PICARD and M. QUEYRANNE, March 1984, "On Finding the k Best Cuts in a Network," *OR Letters*, 2, no. 6(303-305).

H. HAMACHER, J. C. PICARD and M. QUEYRANNE, 1984, "Ranking the Cuts and Cut-Sets of a Network," *Annals of Discrete Applied Mathematics*, 19(183-200).
M. S. HUNG, 1983, "A Polynomial Simplex Method for the Assignment Problem," *OR*, 31(595-600).
M. S. HUNG and W. D. ROM, 1980, "Solving the Assignment Problem by Relaxation," *OR*, 28(969-982).
R. W. IRVING, P. LEATHER and D. GUSFIELD, July 1987, "An Efficient Algorithm for the 'Optimal' Stable Marriage," *JACM*, 34, n0. 3(532-543).
S. Y. ITOGA, Aug. 1978, "The Upper bound for the Stable Marriage Problem," *JORS*, 29, no. 8(811-814).
R. JONKER and T. VOLGENANT, Oct. 1986, "Improving the Hungarian Assignment Algorithm," *OR Letters*, 5, no. 4(171-175).
R. JONKER and A. VOLGENANT, 1987, "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems," *Computing*, 38(325-340).
D. KAPUR and M. S. KRISHNAMOORTHY, July 1981, "Worst Case Choice for the Stable Marriage Problem," *IPL*, 21, no. 1(27-30).
P. KLEINSCHMIDT, C. W. LEE, and H. SCHANNATH, Mar. 1987, "Transportation Problems Which Can be Solved by the Use of Hirsch Paths for the Dual Problems," *MP*, 37, no. 2(153-168).
D. KÖNIG, 1950, "*Theorie der Endlichen und Unendlichen Graphen*," Chelsea Publishing Co. NY.
H. W. KUHN, 1955, "The Hungarian Method for the Assignment Problem," *NRLQ*, 2(83-97).
E. L. LAWLER, Mar. 1972, "A Procedure for Computing the k Best Solutions to Discrete Optimization Problems and its Application to the Shortest Path Problem," *MS*, 18(401-405).
R. E. MACHOL and M. WIEN, April 1977, "Errata," *OR*, 25, no. 2(364).
T. MATSUI, A. TAMURA, and Y. IKEBE, March 1991, "Algorithms for finding a kth best valued assignment," Tech. report, Dept. of Industrial Administration, Science University of Tokyo, Tokyo, Japan.
L. B. METRICK and J. D. MAYBEE, 1973, "Assignment Ranking," Tech. Report 73-7, Dept. IOE, University of Michigan, Ann Arbor, MI.
K. G. MURTY, May-June 1968, "An Algorithm for Ranking All the Assignments in Order of Increasing Cost," *OR*, 16, no. 3(682-687).
K. G. MURTY and C. WITZGALL, 1977, "Dual Simplex Method for the Uncapacitated Transportation Problem Using Tree Labelings" Tech. Report 77-9, Dept.

IOE, University of Michigan, Ann Arbor, MI.

R. V. NAGELHOUT and G. L. THOMPSON, 1984, "A Study of the Bottleneck Single Source Transportation Problem," *COR*, 11, no. 1(25-36).

S. C. NTAFOS and S. L. HAKIMI, Sept. 1979, "On Path Cover Problems in Digraphs and Applications to Program Testing," *IEEE Transactions on Software Engineering*, SE-5, no. 5(520-529).

K. PAPARRIZOS, 1990, "Generalization of a Signature Method to Transportation Problems," Math. Democritus University of Thrace, Xanthi, Greece.

E. ROOHY-LALEH, 1981, "Improvements to the Theoretical Efficiency of the Network Simplex Method," Ph. D thesis, Carleton University.

R. SILVER, 1960, "An Algorithm for the Assignment Problem," *CACM*, 3(603-606).

N. TOMIZAWA, 1972, "On Some Techniques Useful for Solution of Transportation Network Problems," *Networks*, 1(179-194).

J. H. VANDE VATE, 1989, "Linear Programming Brings Marital Bliss," *OR Letters*, 8(147-153).

A. WEINTRAUB, 1973, "The Shortest and the k-Shortest Routes as Assignment Problems," *Networks*, 3(61-73).

L. B. WILSON, 1977, "Assignment Using Choice Lists," *ORQ*, 28, no. 3(569-578).

Index

For each index entry we provide the page number where it is defined or discussed first.