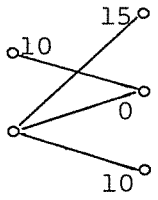
- Selecting x_{22} to enter the basis gives:

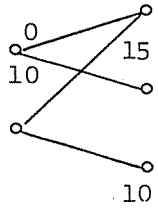
$u_i \backslash v_j$	2	5	4
-3	(2)	10	(2)
0	15	0	10



Optimal

- Or selecting x_{11} (instead of x_{22}) to enter the basis gives:

$u_i \backslash v_j$	2	5	4
-1	0	10	(0)
0	15	(2)	10



Optimal

(Note: We get the same x_{ij}^* but different optimal spanning tree representations for alternative optimal solutions.)

Assignment Problem (AP)

Consider assigning n jobs to n machines such that one job is assigned to one machine and one machine gets only one job. (Total number of possible assignment is $n!$) A cost c_{ij} is associated with assigning job i to machine j, $i = 1, 2, \cdots, n$; $j = 1, 2, \cdots, n$. The least total cost assignment is then a (zero-one) linear program as follows:

$$\begin{cases} \text{Min} \quad x_0 = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\ \text{subject to} \quad \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, 2, \cdots, m) & \text{[one job sets one machines]} \\ \displaystyle\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, 2, \cdots, n) & \text{[one machine gets one job]} \\ x_{ij} = 0, 1 \quad (1 = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n) & \end{cases}$$

REMARK: Both TP and AP have their coefficient matrices as node-edge incidence matrices, which are *totally unimodular*, i.e. the determinants of all their square submatrices are equal to $0, \pm 1$. This implies that the BFS solutions (hence the optimal solutions) are integer-valued even if the integral constraints are discarded, provided all s_i and d_j are integers in the case of TP.

Standard LP methods are applicable for AP (with $x_{ij} = 0, 1$ replaced by $0 \leq x_{ij} \leq 1$); however, there is a much more efficient direct method (generally known as the assignment algorithm). Take as an example a 3×3 cost matrix of AP:

$$C \equiv \begin{array}{|c|c|c|} \hline c_{11} & c_{12} & c_{13} \\ \hline c_{21} & c_{22} & c_{23} \\ \hline c_{31} & c_{32} & c_{33} \\ \hline \end{array}$$

The key observation of the assignment algorithm is that without loss of generality, we may assume $c_{ij} \geq 0 \ \forall i, j$. This is justified by the following claim.

CLAIM. *If a constant is added to or subtracted from any row or column of C, giving C'; the minimization of the modified objective function $x_0' = \sum_{i,j} c_{ij}' x_{ij}$ yields the same solution x_{ij} as the original objective function $x_0 = \sum_{i,j} c_{ij} x_{ij}$.*

PROOF: Suppose $-p_i$ is added to row i and q_j is subtracted from column j. Then

$$\begin{aligned} x_0' &= \sum_{i,j} c_{ij}' x_{ij} = \sum_{i,j} (c_{ij} - p_i - q_j) x_{ij} = \sum_{i,j} c_{ij} x_{ij} - \sum_i p_i \sum_j x_{ij} - \sum_j q_j \sum_i x_{ij} \\ &= \sum_{i,j} c_{ij} x_{ij} - \left(\sum_i p_i + \sum_j q_i \right) = x_0 - \text{constant} \end{aligned}$$

∎

We use this idea to create a new coefficient matrix C' with at least one zero element in each row and in each column, and if using *zero elements only* (or a subset of which) yields a feasible assignment (with total cost = 0, of course), then this assignment is optimal because the total cost of any feasible assignment ≥ 0, since $c_{ij}' \geq 0 \ \forall_{ij}$.

Basically this approach is a dual method because at any time, p_i and q_j together yield a feasible solution to the dual of AP. [Exercise. Construct this dual.] This is because $c_{ij}' \geq 0 \Rightarrow c_{ij} - p_i - q_j \geq 0$ or $p_i + q_j \leq c_{ij}$.

EXAMPLE

5	7	9
14	10	12
15	13	16

$p_1 = 5$

$p_2 = 10$

$p_3 = 13$

$\sum_i p_i = 28$

$\Rightarrow$

0	2	4
4	0	2
2	0	3

$q_1 = 0\ q_2 = 0\ q_3 = 2; \sum_j q_j = 2$

L

L	0*	2	2
$\Rightarrow L$	4	0	0*
	2	0*	1

Now the number of zero cells $(= 4) \geq n(= 3)$ and $\sum_i p_i + \sum_j q = 30$ [which is the cost of any optimal assignment. Why?]

Define a *"cover"* by a *minimum* number of lines to cover all zero elements. Then "cover" $\leq n$. If the "cover" is n then we have an assignment on only the zero elements. The actual assignment (of jobs to machines) is obtained by a *trace-back* as follows:

Let $z_{ij} \equiv$ number of zeros in row i + column j, where the i-j^{th} entry is zero. Make successive assignments in *increasing* z_{ij} order. Delete row i and column j upon assignment i-j is made.

For our example, we have the optimal assignment of $x^*_{11} = x^*_{23} = x^*_{32} = 1$, $x^*_{ij} = 0\ \forall$ other i, j with $x^*_0 = 30$.

Improvement algorithm when "cover" $<$ n

Let $h \equiv \text{Min}_{(i,j)|c_{ij}-(p_i+q_j)>0}[c_{ij} - (p_i + q_j)] > 0$

Set $\begin{cases} p_i \leftarrow p_i - h & \text{if } i \text{ is a covered row} \\ p_i \text{ unchanged} & \text{if not} \end{cases}$

Set $\begin{cases} q_j \leftarrow q_j + h & \text{if } j \text{ is } \textit{not} \text{ a covered column} \\ q_j \text{ unchanged} & \text{if covered} \end{cases}$

Graphically,

	covered	uncovered
covered	$+h$	0
uncovered	0	$-h$

Observe that since all zeros are previously covered, all entries after this change remain ≥ 0. This algorithm creates (at least) one more zero entry, which is previously uncovered and positive; while possibly increasing some previously zero entries that are covered by 2 lines. To apply this algorithm, select the smallest uncovered element, subtract that from every *uncovered* element and add that to every element covered by two lines.

EXAMPLE ON ASSIGNMENT PROBLEM ($n = 4$)

Subtracting constants from rows and columns of the assignment tableau gives:

$$\begin{array}{cccc} 1 & 4 & 6 & 3 \\ 8 & 7 & 10 & 9 \\ 4 & 5 & 11 & 7 \\ 6 & 7 & 8 & 5 \end{array} \quad \begin{array}{c} 1 \\ 7 \\ 4 \\ 5 \\ \hline \sum = 17 \end{array} \quad \begin{array}{cccc} 0 & 3 & 5 & 2 \\ 1 & 0 & 3 & 2 \\ 0 & 1 & 7 & 3 \\ 1 & 2 & 3 & 0 \\ \hline 0 & 0 & 3 & 0 \end{array} \Big| \sum = 3$$

$(3 \text{ lines} < 4 = n)$

$$\begin{array}{cc} & L \\ & \begin{array}{cccc} 0 & 3 & 2 & 2 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 4 & 3 \\ 1 & 2 & 0 & 0 \end{array} \\ L & \\ & \\ L & \end{array} \qquad (\textstyle\sum p_i + \sum q_j = 20)$$

Selecting the smallest uncovered element ($h = 1$ in this case), subtracting that from every *uncovered* element and adding that to every element covered by 2 lines gives:

$(h = 1)$

$$\begin{array}{cc} & L \\ & \begin{array}{cccc} 0 & 3 & 2 & 2 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 4 & 3 \\ 1 & 2 & 0 & 0 \end{array} \\ L & \\ & \\ L & \end{array} \quad \begin{array}{c} \\ -h \\ \\ -h \\ \\ +h \ +h \ +h \end{array} \quad \begin{array}{cccc} \boxed{0} & 2 & 1 & 1 \\ 2 & 0 & \boxed{0} & 2 \\ 0 & \boxed{0} & 3 & 2 \\ 2 & 2 & 0 & \boxed{0} \end{array}$$

Assignment (in increasing order of number of zeros in row and column) gives:

$$x_{11}{}^* = x_{23}{}^* = x_{32}{}^* = x_{44}{}^* = 1 \text{ and all other } x_{ij}{}^* = 0.\ x_0{}^* = 20 + [3(1) - 2(1)] = 21\ .$$

(Alternatively, $x_0{}^* = c_{11} + c_{23} + c_{32} + c_{44} = 1 + 10 + 5 + 5 = 21$.)

Network Flows Problems

Consider a *directed* graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs, i.e. A is a set of *ordered* pairs (i, j) such that $i, j \in N$. A (simple) path in G is a