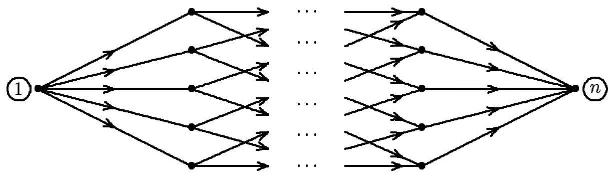
5. NETWORK FLOWS

5.1 Maximal flow in a network

We consider a network with nodes $N = \{1, 2, \ldots, n\}$ and directed arcs between some, or all, of them. We will study the problem of flow in the network between two designated nodes: 1, called the **source**, and n, known as the **sink** of the network. For each arc (i, j) from node i to node j there is an associated capacity $c_{ij} \geqslant 0$ and if x_{ij} is the flow in that arc then we require that x_{ij} satisfy $0 \leqslant x_{ij} \leqslant c_{ij}$.



Think of a flow of value v entering the network through the source 1 and exiting through the node n; at all other nodes the net flow is zero so that flow is conserved except at the source and sink; the object is to find the largest flow value v subject to the capacity constraints on the flows in the arcs. Mathematically, the problem is to

$$\text{maximize } v \quad \text{subject to} \qquad 0 \leqslant x_{ij} \leqslant c_{ij}, \text{ for all } i, j \quad \text{and}$$

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} v & \text{if } i = 1, \\ 0 & \text{if } i \neq 1, n \\ -v & \text{if } i = n. \end{cases}$$

This is a linear programming problem but the special structure enables us to analyze it in a different way. For a subset $S \subseteq N$ of nodes denote its complement in N as $\overline{S} = N \backslash S$. For subsets S and T of nodes in the network denote by $(S, T) = \{(i, j) : \ i \in S, \ j \in T\}$ the set of arcs in the network from nodes i in S to nodes j in T.

Definition A **cut** separating the source 1 and the sink n is a set of arcs of the form $(S, \overline{S})$ with $1 \in S$ and $n \in \overline{S}$.

A cut has the property that if all the arcs in the cut were removed from the network it would separate 1 and n, so there could be no flow from the source to the sink.

Definition The **capacity** of a cut $(S, \overline{S})$ is $c(S, \overline{S}) = \sum_{i \in S, j \in \overline{S}} c_{ij}$; that is, the sum of the capacities over all arcs in the cut.

It is intuitively clear that the capacity of any cut should be an upper bound for the value of any flow since any flow must flow through the cut. We can prove something stronger which is the central characterization of a maximal flow.

Theorem 5.1 (Max-Flow Min-Cut Theorem) *For any network, the maximal value of a flow from 1 to n equals the minimum cut capacity over all cuts separating* 1 *and* n*.*

Proof. For any flow (x_{ij}) and subsets of nodes S, T, denote by $f(S, T) = \sum_{i \in S, j \in T} x_{ij}$ the flow in arcs from S to T. If $(S, \overline{S})$ is a cut and (x_{ij}) is a feasible flow then summing the constraints

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} v & \text{if } i = 1, \\ 0 & \text{if } i \neq 1, n \\ -v & \text{if } i = n, \end{cases}$$

over nodes $i \in S$ we get

$$\begin{aligned} v &= \sum_{i \in S} \sum_{j \in N} (x_{ij} - x_{ji}) = \sum_{i \in S} \sum_{j \in N} x_{ij} - \sum_{i \in S} \sum_{j \in N} x_{ji} \\ &= f(S, N) - f(N, S) = f(S, S) + f(S, \overline{S}) - f(S, S) - f(\overline{S}, S) \\ &= f(S, \overline{S}) - f(\overline{S}, S) \leqslant f(S, \overline{S}) \leqslant c(S, \overline{S}), \end{aligned} \tag{1}$$

the first inequality being because $x_{ij} \geqslant 0$ and the second since $x_{ij} \leqslant c_{ij}$. This shows that the value of any flow is less than the capacity of any cut, so that Max Flow $\leqslant$ Min Cut. We show now that there is equality. Let (x_{ij}) be a maximal flow (it is clear that a maximal flow exists, why?). Define a set of nodes S in the following recursive way:

(i) $1 \in S$;

(ii) if $i \in S$ and $x_{ij} < c_{ij}$ then $j \in S$;

(iii) if $i \in S$ and $x_{ji} > 0$ then $j \in S$.

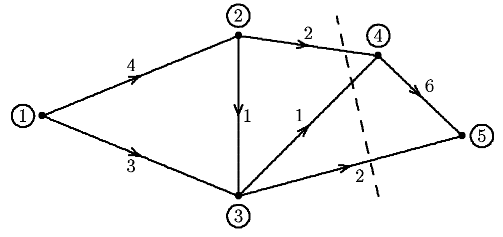
It is clear that the set of nodes S is constructed so that it is the set of nodes to which more flow can be pumped from the source 1, consistent with the constraints. Then we must have $n \in \overline{S}$, for otherwise there would be a path from 1 to n along which more flow could be pumped which would contradict the fact that the flow is maximal. Hence $(S, \overline{S})$ is a cut separating 1 and n with the properties that

(a) if $i \in S$ and $j \in \overline{S}$ then $x_{ij} = c_{ij}$; and

(b) if $i \in \overline{S}$ and $j \in S$ then $x_{ij} = 0$.

It follows that for the maximal flow and this cut $(S, \overline{S})$ we have equality in the two inequalities in (1), completing the proof. □

Example 5.2 Consider the network with the capacities shown



A maximal flow with value 5 is $x_{12} = 2$, $x_{13} = 3$, $x_{34} = 1$, $x_{35} = 2$, $x_{23} = 0$, $x_{24} = 2$ and $x_{45} = 3$. The corresponding minimal cut is indicated by the dashed line. □

Note From the proof of the theorem, we see that to prove that a particular flow is optimal, it is sufficient to find a cut that has capacity equal to the value of the flow.

5.2 The Ford–Fulkerson Algorithm

The proof of the max-flow min-cut theorem suggests an algorithm for finding a maximal flow; this algorithm is known as the **Ford–Fulkerson** algorithm (the theorem is sometimes known as the Ford–Fulkerson Theorem). The algorithm starts with some feasible flow (x_{ij}), say $x_{ij} \equiv 0$. Then construct the set of nodes S iteratively as in the theorem:

(i) $1 \in S$;

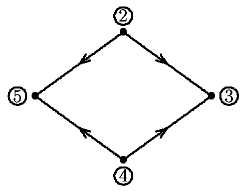
(ii) if $i \in S$ and $x_{ij} < c_{ij}$ then $j \in S$;

(iii) if $i \in S$ and $x_{ji} > 0$ then $j \in S$.

This process continues until either (i) $n \in S$, in which case the current flow may be increased, or (ii) no more nodes may be included in S and $n \notin S$, in which case the current flow is optimal. The construction of S ensures that S consists of those nodes j to which more flow may be 'pumped' from 1 consistent with the capacity constraints; if the node $i \in S$ (so more flow can reach i) and the flow in the arc (i, j) is below capacity, $x_{ij} < c_{ij}$ then more flow can reach j, while if the arc (j, i) is such that $x_{ji} > 0$ then more flow can reach j by reducing the flow in the backward arc (j, i). If $n \in S$ then by the construction of S, there is a path from 1 to n along which the flow can be increased (by the amount $\delta = \min\left[\min_{(i,j)}(c_{ij} - x_{ij}), \min_{(j,i)} x_{ji}\right]$, where the minimum over (i, j) is over forward arcs of the path and the minimum over (j, i) is over backward arcs (j, i) of the path). Now the flow can be incremented by δ along the path to give a new flow and the algorithm can be iterated again.

Note If the arc capacities and the initial flow are rational then this process terminates in a finite number of steps at the optimal flow. To see this, scale up the capacities and the initial flow so that they are all integers, then by the construction above $\delta \geqslant 1$ so that the flow always increases by at least 1 on each iteration.

Example 5.3 *Irrational flows* If the initial flow is irrational then (even if the capacities are rational) the algorithm may not terminate in a finite number of steps and may not even converge to the optimal solution. This example has six nodes $\{1, 2, 3, 4, 5, 6\}$ and (special) arcs, each with capacity 1, as follows



In addition, there are arcs $(1, i)$ and $(i, 6)$, $i = 2, 3, 4, 5$ (not shown), each with capacity 10. The node 1 is the source and 6 is the sink. Start with an initial flow which sends flow 1 along the route $1 \to 2 \to 3 \to 6$ and flow w along the route $1 \to 4 \to 5 \to 6$, where

$w = \left(\sqrt{5}-1\right)/2$, and w satisfies $1-w=w^2$; also $w^{n-1}-w^n=w^{n+1}$, for all n. Now suppose that after the $(2n-1)$th iteration of the algorithm, the flows in the special arcs are shown in Figure 1.

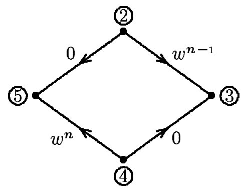


Figure 1

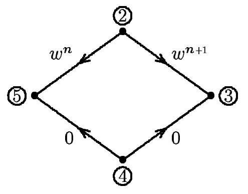


Figure 2

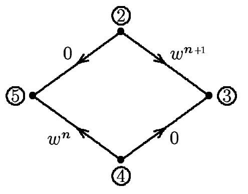


Figure 3

For the $(2n)$th step, route flow of w^n along the path $1\to 3\to 2\to 5\to 4\to 6$ to get the flows in the special arcs shown in Figure 2. For the $(2n+1)$th step send flow of w^n along the route $1\to 4\to 5\to 2\to 6$ to get the flows in Figure 3. Notice that the form of Figure 3 is the same as Figure 1 (with 2 and 4 interchanged and 3 and 5 interchanged). Between the two steps the flow is increased by $2w^n$; these steps may be repeated in the obvious way and the total flow then converges to

$$1+w+2\sum_{n=1}^{\infty} w^n = 1+w+2w/(1-w),$$

which is less than 5 and considerably less than the optimal flow, which is clearly 40. □

5.3 Minimal-cost circulation

Many optimization problems can be formulated in the form of minimizing the cost of flows which circulate in a closed network; that is, unlike the situation with the maximal-flow problem, where flow enters the network at the source and exits at the sink, in a closed network the flow is conserved at each node, so that

$$\sum_j (x_{ij}-x_{ji}) = 0, \quad \text{for each node } i. \qquad (1)$$

It is usually assumed that the flow in arc (i,j) is restricted so that there are upper and lower bounds on the flow,

$$c_{ij}^- \leqslant x_{ij} \leqslant c_{ij}^+, \quad \text{for each arc } (i,j) \qquad (2)$$

where c_{ij}^- and c_{ij}^+ are finite for each (i,j) and there is a cost d_{ij} per unit flow in that arc. So the problem may be formulated as:

$$\text{minimize } \sum_{i,j} d_{ij}x_{ij} \quad \text{subject to } \sum_j (x_{ij}-x_{ji}) = 0, \quad \text{for each } i, \text{ and } c_{ij}^- \leqslant x_{ij} \leqslant c_{ij}^+.$$

A flow is feasible if it satisfies (1), for each node, and the capacity constraints in (2), for each arc. Important conditions for establishing the optimality of a flow are provided by using the Lagrangian Sufficiency Theorem.

Theorem 5.4 *A feasible flow (x_{ij}) is optimal for the minimal-cost circulation problem if there exist numbers (λ_i) such that*

$$x_{ij} = \begin{cases} c_{ij}^+ & \textit{if} \quad d_{ij}-\lambda_i+\lambda_j < 0, \\ c_{ij}^- & \textit{if} \quad d_{ij}-\lambda_i+\lambda_j > 0 \end{cases}$$

and $d_{ij}-\lambda_i+\lambda_j = 0$ whenever $c_{ij}^- < x_{ij} < c_{ij}^+$.

Proof. We wish to minimize the Lagrangian

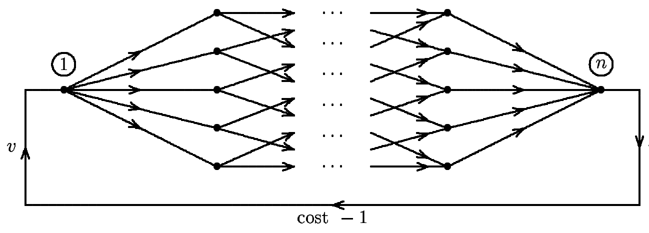
$$L(\boldsymbol{x},\boldsymbol{\lambda}) = \sum_{i,j} d_{ij}x_{ij} - \sum_i \lambda_i \sum_j (x_{ij}-x_{ji}) = \sum_{i,j} (d_{ij}-\lambda_i+\lambda_j)\, x_{ij},$$

over the constraint region $X = \left\{(x_{ij}) : c_{ij}^- \leqslant x_{ij} \leqslant c_{ij}^+\right\}$. The result follows immediately from the Lagrangian Sufficiency Theorem. □

Remarks

1. The dual variables (λ_i) are sometimes known as **potentials** or **node numbers** for the problem.
2. Notice that in the conditions of the Theorem only the differences between the (λ_i) appear; consequently, without loss of generality one λ_i may be chosen arbitrarily, usually it is set $=0$.
3. In the case when one or other of c_{ij}^- or c_{ij}^+ is infinite, e.g., when $c_{ij}^+ = \infty$, then we can see that we obtain a dual-feasibility condition for a finite minimum for the problem, in this case it would be $d_{ij}-\lambda_i+\lambda_j \geqslant 0$.

Example 5.5 *Maximal flow as a minimal-cost circulation* It is interesting to note that the maximal-flow problem may be formulated as a minimal-cost circulation. Add another arc from n to 1 with cost $d_{n1} = -1$, and for the other regular arcs (i,j), take $c_{ij}^- = 0$ and $c_{ij}^+ = c_{ij}$ with the cost $d_{ij} = 0$.



Maximizing the flow, v, in the arc $(n,1)$ is the same as minimizing the total cost in the enlarged network which is $-v$. For the arc $(n,1)$, we will have no capacity constraints, that is $c_{n1}^- = -\infty$ and $c_{n1}^+ = +\infty$, so necessarily we must have $d_{n1} - \lambda_n + \lambda_1 = 0$; one of the node numbers may be chosen arbitrarily, so take $\lambda_n = 0$ which gives $\lambda_1 = 1$. Take a minimal cut $(S, \overline{S})$ and set $\lambda_i = 1$ for $i \in S$ and $\lambda_j = 0$ for $j \in \overline{S}$, then check that

(i) when both $i,j \in S$ or both $i,j \in \overline{S}$, $d_{ij} - \lambda_i + \lambda_j = 0$, so that x_{ij} can take any feasible value;

(ii) when $i \in S$ and $j \in \overline{S}$, $d_{ij} - \lambda_i + \lambda_j = -1 < 0$, so that $x_{ij} = c_{ij}^+ = c_{ij}$; and

(iii) when $i \in \overline{S}$ and $j \in S$, $d_{ij} - \lambda_i + \lambda_j = \ \ 1 > 0$, so that $x_{ij} = c_{ij}^- = 0$.

These are just the conditions for optimality for a maximal flow.

5.4 The transportation problem and transportation algorithm

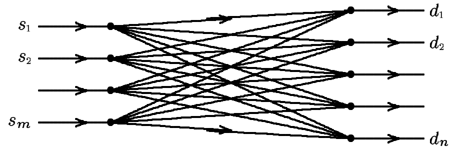
A classical optimization problem, which is a special case of minimal-cost circulation, is the **transportation problem** in which there are m sources of supply of a particular good, $\{S_1, \dots, S_m\}$, with amounts $\{s_1, \dots, s_m\}$ available, and n destinations, $\{D_1, \dots, D_n\}$ at which there are demands $\{d_1, \dots, d_n\}$, respectively, for the good. For each pair (S_i, D_j) there is a cost d_{ij} per unit for shipping from S_i to D_j.

Assumption: $\sum_{i=1}^m s_i = \sum_{j=1}^n d_j$; that is, total supply equals total demand.

The objective is to satisfy the demand from the supplies with the minimal transportation cost. Denote by x_{ij} the flow from S_i to D_j, then the problem is

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^m \sum_{j=1}^n d_{ij} x_{ij} \\ &\text{subject to} && \sum_{j=1}^n x_{ij} = s_i, \quad 1 \leqslant i \leqslant m, \\ &&& \sum_{i=1}^m x_{ij} = d_j, \quad 1 \leqslant j \leqslant n, \quad x_{ij} \geqslant 0, \quad \text{for all } i,j. \end{aligned} \tag{1}$$

The problem may be formulated as a network as shown:



The Lagrangian for the problem is

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) &= \sum_{i=1}^m \sum_{j=1}^n d_{ij} x_{ij} + \sum_{i=1}^m \lambda_i \left(s_i - \sum_{j=1}^n x_{ij} \right) + \sum_{j=1}^n \nu_j \left(d_j - \sum_{i=1}^m x_{ij} \right) \\ &= \sum_{i=1}^m \sum_{j=1}^n (d_{ij} - \lambda_i - \nu_j) x_{ij} + \sum_{i=1}^m \lambda_i s_i + \sum_{j=1}^n \nu_j d_j. \end{aligned}$$

The minimum of the Lagrangian over $x_{ij} \geqslant 0$ will be finite provided:

$$d_{ij} - \lambda_i - \nu_j \geqslant 0, \qquad \text{for each } i,j, \quad \text{(dual feasibility)}$$

and at the optimum

$$(d_{ij} - \lambda_i - \nu_j) x_{ij} = 0, \quad \text{for each } i,j, \quad \text{(complementary slackness).}$$

The existence of (λ_i) and (ν_j) satisfying these two conditions together with of course the primal feasibility conditions, that the (x_{ij}) satisfy the constraints in the problem (1), are necessary and sufficient conditions for optimality.

There is a version of the simplex algorithm, specially tailored for this problem, known as the **transportation algorithm**; it may be interpreted nicely in terms of the network describing the problem. It examines a sequence of basic feasible solutions to the problem and chooses (λ_i) and (ν_j) to satisfy complementary slackness at each stage and works towards their also satisfying dual feasibility (at which point the three necessary and sufficient conditions for optimality are satisfied and an optimal solution is obtained).

First notice that while there are $m + n$ linear constraints in the problem, one of the constraints will always be linearly dependent on the others so that there will be $m + n - 1$ basic variables in a basic feasible solution. A second observation is that a basic feasible solution corresponds to flow in the network just on a spanning tree in the network; a subnetwork **spans** the network if it touches all the nodes in the network and it is a **tree** if it contains no circuits.

We will introduce the algorithm by solving a particular problem with 3 sources of supply with supplies $(12, 8, 11)$ respectively, and 4 destinations with demands $(7, 6, 10, 8)$; the matrix of unit costs for shipping is

$$\begin{pmatrix} 8 & 6 & 7 & 5 \\ 4 & 3 & 5 & 4 \\ 9 & 8 & 6 & 7 \end{pmatrix},$$

where the (i, j) element is the cost of shipping 1 unit from S_i to D_j.

Step 1. Initial assignment We start the algorithm by choosing an initial basic feasible solution; that is one in which there is positive flow (if the problem is non-degenerate) in exactly $m+n-1$ arcs of the network, or $m+n-1$ cells if the problem is set out in tabular form, as here.

					Supplies
	$\boxed{7}$ 8	$\boxed{5}$ 6	7	5	12
	4	$\boxed{1}$ 3	$\boxed{7}$ 5	4	8
	9	8	$\boxed{3}$ 6	$\boxed{8}$ 7	11
Demands	7	6	10	8	

This assignment is chosen by what is known as the North-West (NW) method and the flows in the 'basic' cells are indicated in the upper left-hand corner of the cells with the costs (d_{ij}) in the lower-right hand corners; the NW method puts as much flow as possible

in the cell $(1, 1)$ then either moves over to the cell $(1, 2)$ or down to the cell $(2, 1)$ and assigns as much as possible again in the obvious way, and continues until all the demands are satisfied and all supplies used up. In this case the total cost of this flow is $7 \times 8 + 5 \times 6 + 1 \times 3 + 7 \times 5 + 3 \times 6 + 8 \times 7 = 198$.

Step 2. Assign the Lagrange multipliers Next, choose values for the Lagrange multipliers (λ_i), (ν_j), so that $d_{ij} - \lambda_i - \nu_j = 0$ for the basic cells; this ensures that complementary slackness holds. As only the sum of $\lambda_i + \nu_j$ enter into all the calculations one of these multipliers may be chosen arbitrarily; set $\lambda_1 = 0$, say.

λ_i \ ν_j	8	6	8	9
0	$\boxed{7}$ 8	$\boxed{5}$ 6	7	5
-3	4	$\boxed{1}$ 3	$\boxed{7}$ 5	4
-2	9	8	$\boxed{3}$ 6	$\boxed{8}$ 7

We can then assign, in order, ν_1, ν_2, λ_2, ν_3, λ_3 and ν_4.

Step 3. Check for optimality Identify those (non-basic) cells for which $d_{ij} - \lambda_i - \nu_j < 0$; if all cells have $d_{ij} - \lambda_i - \nu_j \geqslant 0$ then the current solution is optimal. A ✓ has been placed in a non-basic cell which satisfies dual feasibility, $d_{ij} - \lambda_i - \nu_j \geqslant 0$, and a $\times$ in those non-basic cells for which $d_{ij} - \lambda_i - \nu_j < 0$, along with the numerical difference $d_{ij} - \lambda_i - \nu_j$.

λ_i \ ν_j	8	6	8	9
0	$\boxed{7}$ 8	$\boxed{5}$ 6	$\times\ -1$ 7	$\times\ -4$ 5
-3	$\times\ -1$ 4	$\boxed{1}$ 3	$\boxed{7}$ 5	$\times\ -2$ 4
-2	✓ 9	✓ 8	$\boxed{3}$ 6	$\boxed{8}$ 7

In this case the flow is not optimal so the pivot operation must occur.

Step 4. Pivoting Choose one of the non-basic cells for which $d_{ij} - \lambda_i - \nu_j < 0$; the algorithm will work if any such cell is chosen, but in line with our rule-of-thumb in the simplex algorithm we will choose the one with the most negative value of $d_{ij} - \lambda_i - \nu_j$, which is shaded below. Increase the flow in this cell by ϵ and form a loop as shown, increasing or decreasing the flow by ϵ in basic cells to ensure that the row and column totals for the flow are unchanged (and equal the supplies and demands, that is, preserving primal feasibility).

$\boxed{7}$ 8	$\boxed{5-\epsilon}$ 6	7	▨ ϵ 5
4	$\boxed{1+\epsilon}$ 3	$\boxed{7-\epsilon}$ 5	4
9	8	$\boxed{3+\epsilon}$ 6	$\boxed{8-\epsilon}$ 7

We can increase ϵ until the flow in one of the basic cells becomes zero, in this case when $\epsilon = 5$, and this gives a new basic feasible flow :

$\boxed{7}$ 8	6	7	$\boxed{5}$ 5
4	$\boxed{6}$ 3	$\boxed{2}$ 5	4
9	8	$\boxed{8}$ 6	$\boxed{3}$ 7

The algorithm now returns to Step 2 with this flow as the basic feasible flow. Notice that the new total cost is 178; the reduction in the value of the flow is $\epsilon \times (d_{ij} - \lambda_i - \nu_j)$ for the cell which enters the basis. We set out the two further iterations of the algorithm for this problem that are required to obtain the optimal solution.

$\lambda_i \backslash \nu_j$	8	2	4	5
0	$\boxed{7-\epsilon}$ 8	✓ 6	✓ 7	$\boxed{5+\epsilon}$ 5
1	▨ ϵ $\times\ -5$ 4	$\boxed{6}$ 3	$\boxed{2-\epsilon}$ 5	$\times\ -2$ 4
2	$\times\ -1$ 9	✓ 8	$\boxed{8+\epsilon}$ 6	$\boxed{3-\epsilon}$ 7

$\lambda_i \backslash \nu_j$	8	7	4	5
0	$\boxed{5-\epsilon}$ 8	▨ ϵ $\times\ -1$ 6	✓ 7	$\boxed{7}$ 5
-4	$\boxed{2+\epsilon}$ 4	$\boxed{6-\epsilon}$ 3	✓ 5	✓ 4
2	$\times\ -1$ 9	$\times\ -1$ 8	$\boxed{10}$ 6	$\boxed{1}$ 7

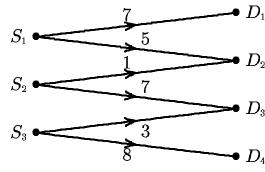
Total cost = 168

$\lambda_i \backslash \nu_j$	7	6	4	5
0	✓ 8	$\boxed{5}$ 6	✓ 7	$\boxed{7}$ 5
-3	$\boxed{7}$ 4	$\boxed{1}$ 3	✓ 5	✓ 4
2	✓ 9	✓ 8	$\boxed{10}$ 6	$\boxed{1}$ 7

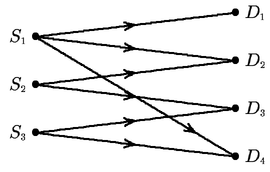
Total cost = 163, optimal

Remarks

1. It is easy to see that the pivot operation (increasing the flow in a non-basic cell by ϵ) will always lead to a circuit, or closed loop, by considering the spanning tree in the network representing the basis. For example, the initial basis in the problem above, with its associated flows, is:



and when the arc (S_1, D_4) enters the basis



leading to the circuit (with orientations on the arcs indicated):

$$S_1 \to D_4 \leftarrow S_3 \to D_3 \leftarrow S_2 \to D_2 \leftarrow S_1.$$

2. When a degenerate basic solution is encountered in the transportation algorithm (the demands may be met from the supplies with fewer than $m + n - 1$ basic cells

with positive flows), then to assign the Lagrange multipliers it will be necessary to add 'basic' cells with zero flow and proceed as before. Degeneracy may occur whenever a subset of demands may be satisfied exactly by a subset of supplies.

3. The NW algorithm described above is only one way to initiate the transportation algorithm – any initial basic feasible flow may be used to start it off. The **greedy algorithm** can be a useful starting point since it may reduce the number of subsequent iterations of the algorithm (see the Example Sheet); what it does is, first, to put as much flow as possible on the smallest cost then put as much flow as possible on the next smallest cost and so on. More formally, choose (i', j') so that $d_{i'j'} = \min_{i,j} d_{ij}$ and put flow equal to $\min(s_{i'}, d_{j'})$ into cell (i', j'); then eliminate row i' if $s_{i'} < d_{j'}$ or eliminate column j' if $s_{i'} > d_{j'}$ (if they are equal then eliminate both, the assignment will be degenerate) and repeat the process on the reduced array; when two or more costs tie to be the smallest at any stage, any one of the corresponding cells may be chosen. For example, for our problem above, an initial assignment (with total cost 167) given by the greedy algorithm is:

					Supplies
	$\boxed{4}$ 8	6	7	$\boxed{8}$ 5	12
	$\boxed{2}$ 4	$\boxed{6}$ 3	5	4	8
	$\boxed{1}$ 9	8	$\boxed{10}$ 6	7	11
Demands	7	6	10	8	

You may check that the algorithm will take you to an optimal solution in one iteration starting from this initial basic feasible solution. The greedy algorithm will not produce a unique assignment if there is a tie for the minimal cost remaining at any stage, e.g., for our problem it may produce the assignment below (which has total cost 173) from which you may verify that it requires two iterations to reach an optimal solution.

					Supplies
	$\boxed{6}$ 8	6	7	$\boxed{6}$ 5	12
	4	$\boxed{6}$ 3	5	$\boxed{2}$ 4	8
	$\boxed{1}$ 9	8	$\boxed{10}$ 6	7	11
Demands	7	6	10	8	

4. The network for the transportation problem can be transformed so that the transportation problem may be regarded as a minimal-cost circulation: add an extra node I, say, with arcs (I, S_i) with costs $= 0$ and upper and lower capacity constraints $= s_i$, for each i, and arcs (D_j, I) with costs $= 0$ and upper and lower capacity constraints $= d_j$, for each j. These have the effect of forcing flow s_i into node S_i and forcing flow d_j out of node D_j. The node numbers for the problem will be the (λ_i) and $(-\nu_j)$ we have obtained above.

5. The transportation problem is also a special case of the **distribution** problem for a general network where associated with each arc (i, j) are costs d_{ij} and upper and lower capacity constraints (c_{ij}^-, c_{ij}^+), and at each node i the net flow is constrained to be b_i where the (b_i) are given numbers with the constraint $\sum_i b_i = 0$ (which expresses the constraint 'what goes in must come out'). So the problem is

$$\text{minimize} \quad \sum_{i,j} d_{ij} x_{ij}$$

$$\text{subject to} \sum_j (x_{ij} - x_{ji}) = b_i, \quad \text{for each } i, \quad c_{ij}^- \leqslant x_{ij} \leqslant c_{ij}^+, \quad \text{for all } i, j.$$

In the transportation problem $c_{ij}^- = 0$ and $c_{ij}^+ = \infty$ for each arc; when i is the node S_i, $b_i = s_i$ and when i is the node D_j, $b_i = -d_j$, so that demands are negative supplies. The transportation algorithm is a special case of a version of the simplex algorithm developed for the distribution problem, known as the **simplex-on-a-graph** algorithm.

6 April 2005