12 Transportation and Transshipment Problems

12.1 The transportation algorithm

1. Set out the supplies and costs in a table as below

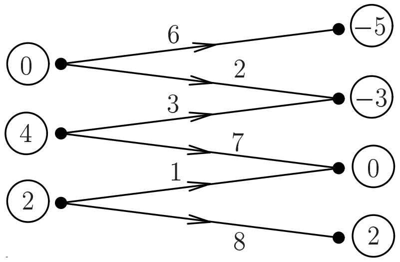
	D_1	D_2	D_3	D_4	s_i
S_1	5	3	4	6	8
S_2	2	7	4	1	10
S_3	5	6	2	4	9
d_j	6	5	8	8	27

2. Allocate an initial feasible flow (by North-West corner rule or any other sensible method). NW corner rule says start at top left corner and dispose of supplies and fulfill demands in order i, j increasing. In our case we get

6	2		
	3	7	
		1	8

In the absence of degeneracy (which we assume) we will obtain $(m + n - 1)$ non-zero entries in a 'stair-case' arrangement.

Remark. In our network picture we have constructed a feasible flow on a **spanning tree** of $m + n - 1$ arcs connecting n sources and m destinations.



A set of arcs is spanning if it connects all nodes. It is a **tree** if it contains no circuits. A spanning tree is the equivalent of a basic solution for this problem.

3. For optimality we require $d_{ij} - \lambda_i + \mu_j = 0$ on any arc with non-zero flow. Set $\lambda_1 = 0$ (arbitrarily) and then compute the remaining λ_i, μ_j by using $d_{ij} - \lambda_i + \mu_j = 0$

on arcs for which $x_{ij} > 0$. On the table we have

$\lambda_i \setminus \mu_j$	-5	-3	0	-2
0	6 $\boxed{5}$	2 $\boxed{3}$	$\boxed{4}$	$\boxed{6}$
4	$\boxed{2}$	3 $\boxed{7}$	7 $\boxed{4}$	$\boxed{1}$
2	$\boxed{5}$	$\boxed{6}$	1 $\boxed{2}$	8 $\boxed{4}$

The node numbers are also shown on the network version above. With non-zero flows forming a spanning tree we will always be able to compute uniquely all node numbers given one of them.

4. We now compute $d_{ij} - \lambda_i + \mu_j$ for all the remaining boxes (arcs). We obtain

0	0	4	4
-7	0	0	-5
-2	1	0	0

which can be written elsewhere in the boxes if desired.

5. If all $d_{ij} - \lambda_i + \mu_j \geq 0$, then the flow is optimal. Stop.

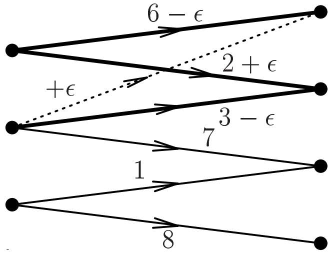
6. If not, (e.g., $i = 2$, $j = 1$ gives -7) we attempt to increase the flow in arc (i, j) for some (i, j) such that $d_{ij} - \lambda_i + \mu_j < 0$. We seek an adjustment of $+\epsilon$ to the flow in arc (i, j) which keeps the solution feasible (and therefore preserves total supplies and demands). In our case we do this by

$6 - \epsilon$	$2 + \epsilon$	0	0
$+\epsilon$	$3 - \epsilon$	7	0
0	0	1	8

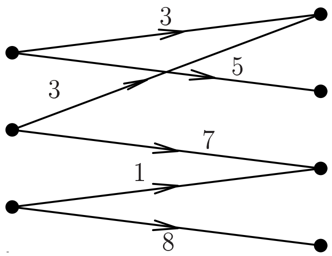
and pick ϵ as large as possible (without any flow going negative) to obtain the new flow (for $\epsilon = 3$) of

3	5	0	0
3	0	7	0
0	0	1	8

There is only one way to do this in a non-degenerate problem. The operation is perhaps clearer in the network picture.



We attempt to increase flow in the dotted arc. Adding an arc to a spanning tree creates a circuit. Increase flow around the circuit until one arc drops out, leaving a new spanning tree. The new solution is



7. Now return to step 3 and recompute node numbers.

In our example we obtain $\lambda_i = 0, -3, -5$ and $\mu_j = -5, -3, -7, -9$ at the next stage. The expression $d_{ij} - \lambda_i + \mu_j < 0$ for $(i,j) = (1,3), (2,4)$ and $(1,4)$. Increase the flow in $(2,4)$ by 7 to obtain the new flow below. This is now optimal. (Check!)

3	5	0	0
3	0	0	7
0	0	8	1

Remark. The route around which you may need to alter flows can be quite complicated though it is always clear how you should do it. For example, had we tried to increase the flow in arc $(3,1)$ instead of $(2,3)$ at step 5 we would have obtained

$6-\epsilon$	$2+\epsilon$	0	0
0	$3-\epsilon$	$7+\epsilon$	0
$+\epsilon$	0	$1-\epsilon$	8

and $\epsilon = 1$. Try this out on a picture of the network and complete the algorithm from this position.

To summarise:

1. Pick initial feasible solution with $m+n-1$ non-zero flows (NW corner rule).
2. Set $\lambda_1 = 0$ and compute λ_i, μ_j using $d_{ij} - \lambda_i + \mu_j = 0$ on arcs with non-zero flows.
3. If $d_{ij} - \lambda_i + \mu_j \geq 0$ for all (i,j) then flow is optimal.
4. If not, pick (i,j) for which $d_{ij} - \lambda_i + \mu_j < 0$.
5. Increase flow in arc (i,j) by as much as possible without making the flow in any other arc negative. Return to 2.

12.2 *Simplex-on-a-graph*

The transportation algorithm can easily be generalised to a problem of minimizing costs in a general network with constraints $0 \leq x_{ij} < \infty$ on each arc and flows b_i into the network at each i (though it is hard to keep track of all the numbers by hand). Here we don't label sources and destinations separately, but do allow $b_i \geq 0$ and $b_i \leq 0$. Clearly, $\sum_i b_i = 0$ for conservation of flow. The **simplex-on-a-graph algorithm** solves this problem in an identical fashion to the transportation algorithm. Once again a basic solution is a spanning tree of non-zero flow arcs.

1. Pick an initial basic feasible solution. Obtain $n-1$ non-zero flow arcs.
2. Set $\lambda_1 = 0$ and compute λ_i on other nodes using $d_{ij} - \lambda_i + \lambda_j = 0$ on arcs of the spanning tree.
3. Compute $d_{ij} - \lambda_i + \lambda_j$ for other arcs. If all these are ≥ 0 then optimal. If not, pick (i,j) such that $d_{ij} - \lambda_i + \lambda_j < 0$.
4. Add arc (i,j) to the tree. This creates a circuit. Increase flow around the circuit (in direction of arc (i,j)) until one non-zero flow drops to zero and a new basic solution is created. Return to 2.

If it is hard to find an initial basic solution then there is a two-phase version of the algorithm (just as for the ordinary simplex algorithm).

Phase I. Add artificial node 0 with arcs to all nodes with $b_i > 0$ and from all nodes with $b_i < 0$. For Phase I objective put costs of 1 on all arcs to and from node 0 and costs 0 elsewhere in the network. The initial basic solution for Phase I is the spanning tree consisting of the node 0 and all arcs joining it to the original nodes.

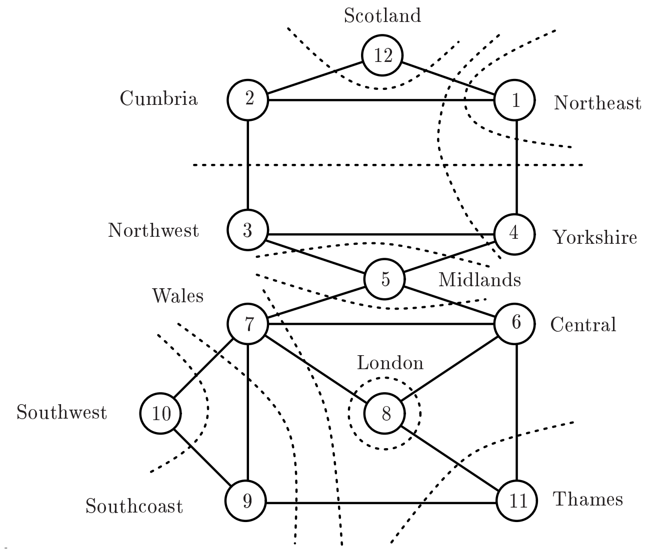
At the end of Phase I (if the original problem was feasible) you will have reduced the Phase I cost to 0 (no flow in arcs to or from node 0), so have a basic solution (Spanning tree solution) for the original problem.

Phase II. Solve the original problem using the initial solution found by Phase I.

There are several applications of the network theory to problems of graph theory and operations research on the further examples sheet.

12.3 Example: optimal power generation and distribution

The following real-life problem can be solved by the simplex-on-a-graph algorithm.



The demand for electricity at node i is d_i. Node i has k_i generators, that can generate electricity at costs of $a_{i1}, \dots, a_{ik_i}$, up to amounts $b_{i1}, \dots, b_{ik_i}$. There are $n = 12$ nodes and 351 generators in all. The capacity for transmission from node i to j is c_{ij} $(= c_{ji})$.

Let x_{ij} = amount of electricity carried $i \to j$ and let y_{ij} = amount of electricity generated by generator j at node i. The LP is

$$\text{minimize } \sum_{ij} a_{ij} y_{ij}$$

$$\text{subject to } \sum_j y_{ij} - \sum_j x_{ij} + \sum_j x_{ji} = d_i, \quad i = 1, \dots, 12,$$

$$0 \le x_{ij} \le c_{ij}, \; 0 \le y_{ij} \le b_{ij}.$$

In addition, there are constraints on the maximum amount of power that may be shipped across the cuts shown by the dotted lines in the diagram.