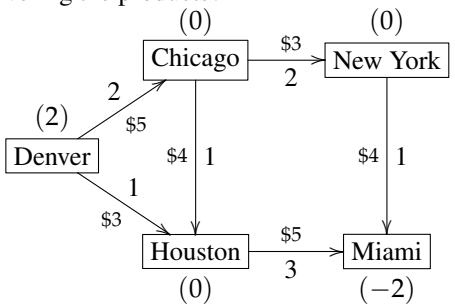
IEOR 4004
Min Cost FLow Problem

A delivery company runs a delivery network between major US cities. Selected cities are connected by routes as shown below. On each route a number of delivery trucks is dispatched daily (indicated by labels on the corresponding edges). Delivering along each route incurs a certain cost (indicated by the $ figure (in thousands) on each edge). A customer hired the company to deliver two trucks worth of products from Denver to Miami. What is the least cost of delivering the products?



minimize

$$5x_{DC} + 3x_{DH} + 4x_{CH} + 3x_{CN} + 4x_{NM} + 5x_{HM}$$

$$\begin{aligned}
x_{DC} + x_{DH} &= 2 \\
-x_{DC} + x_{CH} + x_{CN} &= 0 \\
-x_{DH} - x_{CH} + x_{HM} &= 0 \\
-x_{CN} + x_{NM} &= 0 \\
-x_{NM} - x_{HM} &= -2
\end{aligned}$$

conservation of flow

$$\begin{aligned}
0 \le x_{DC} &\le 2 \\
0 \le x_{DH} &\le 1 \\
0 \le x_{CH} &\le 1 \\
0 \le x_{CN} &\le 2 \\
0 \le x_{NM} &\le 1 \\
0 \le x_{HM} &\le 3
\end{aligned}$$

Formulation

We are given a network $G = (V, E)$. The generic formulation for a minimum-cost flow problem uses the following data:

u_{ij} =*capacity* of an edge $(i, j) \in E$ (# trucks dispatched daily between i and j)
x_{ij} =*flow* on an edge $(i, j) \in E$ (# trucks delivering the customer's products)
c_{ij} =*cost* on an edge $(i, j) \in E$ (cost of transportation per each truck)
b_i =*net supply* of a vertex $i \in V$ (amount of products produced/consumed at node i)

The formulation is given next:

$$\min \sum_{(i,j)\in E} c_{ij} x_{ij}$$

$$\underbrace{\sum_{\substack{j\in V \\ (i,j)\in E}} x_{ij}}_{\text{flow out of } i} - \underbrace{\sum_{\substack{j\in V \\ (j,i)\in E}} x_{ji}}_{\text{flow into } i} = \underbrace{b_i}_{\text{net supply}} \qquad \text{for each node } i$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for all } (i, j) \in E$$

Necessary condition: $\sum_i b_i = 0$.

If there are no capacity constraints, the problem is called the **Transshipment problem**.

Application: Inventory Routing Problems, II

Consider a two-level supply chain. Here there is demand for a product over time. The demand must be met at each time period. Units of the product require two-level processing, with different costs capacity amounts over time. In the following (simple) example we have four time periods and per-unit costs and capacities are given as per the table below

	Level 1		Level 2	
	Prod. Cost	**Prod. Capacity**	**Prod. Cost**	**Prod. Capacity**
1	50	100	200	1000
2	4000	50	82	70
3	30	54	27	100
4	22	90	44	200

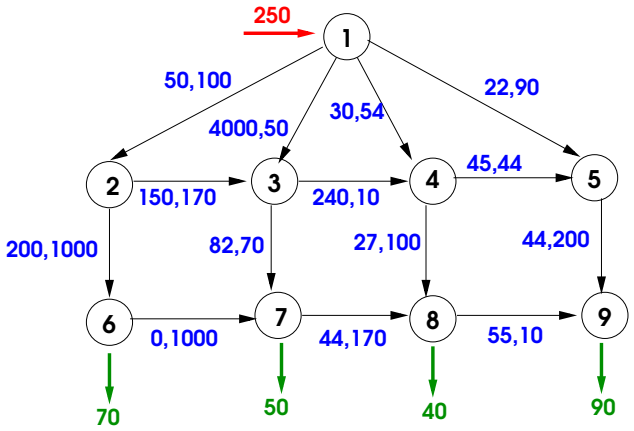
Additionally, units product of the product at each level of completion can be stored as inventory, at a per-unit cost and also subject to capacities.

	Level 1		Level 2	
	Inv. Cost	**Inv. Capacity**	**Inv. Cost**	**Inv. Capacity**
1	150	170	0	1000
2	240	10	44	170
3	45	44	55	10

Finally, demands are as follows.

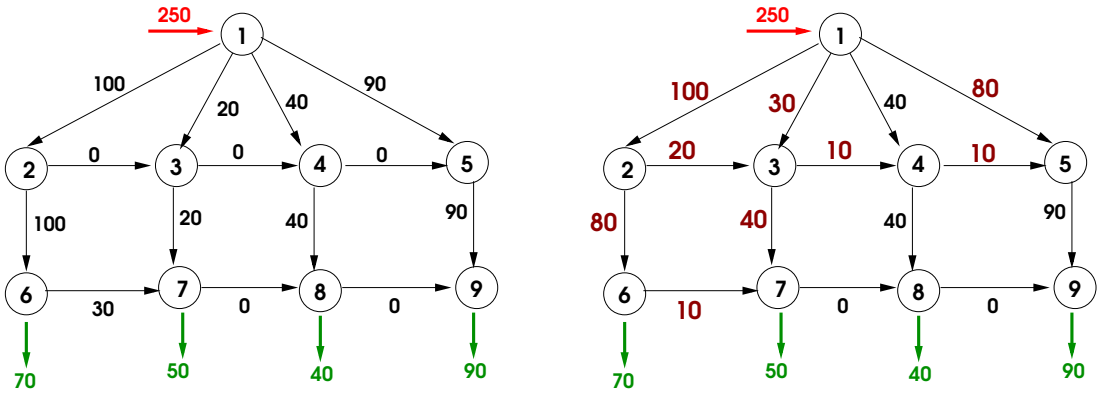
period	1	2	3	4
demand	70	50	40	90

The problem is to establish a production and inventory plan that delivers all demands at minimum cost. It can be represented as a min-cost flow problem using the following network.



In this figure, arcs $(2,6),(3,7),(4,8)$ and $(5,9)$ represent product assembly at level 2 and delivery at time periods 1 through 4; arcs $(1,2),(1,3),(1,4)$ and $(1,5)$ represent product assembly at level 1; finally node 1 is an auxiliary node. The horizontal arcs represent inventory (bottom arcs are inventory at level 2). Next to each arc we show its cost and capacity. Demands are shown in green and supplies (only one) in red.

Now suppose we solve this min-cost flow problem. The solution is indicated on the left, and has cost 114860:



The solution on the right is **suboptimal** (its cost is 158130). Can you see how to improve on it?

Negative cost augmenting cycle algorithm

A methodology for systematically improving on feasible solutions is described next. This method relies on the concept of **negative cost augmenting cycles**. Suppose we have a min-cost flow problem written in the generic form given above. Let x be a given feasible solution and suppose C is a cycle in the network. This cycle will be raversed in a given direction. The cycle is called **negative cost augmenting with respect to x** if it satisfies the following conditions:

(i) All arcs (i,j) that point forward satisfy $x_{ij} < u_{ij}$.

(ii) All arcs (i,j) that point backward satisfy $x_{ij} > 0$.

(iii) The cycle satisfies

$$\sum_{(i,j)\,\text{forward}} c_{ij} \; - \sum_{(i,j)\,\text{backward}} c_{ij} \; < \; 0.$$

For example, in the picture on the right, the cycle $2-6-7-3-2$, traversed in that order, is augmenting (for condition (iii) you get $200-232<0$.

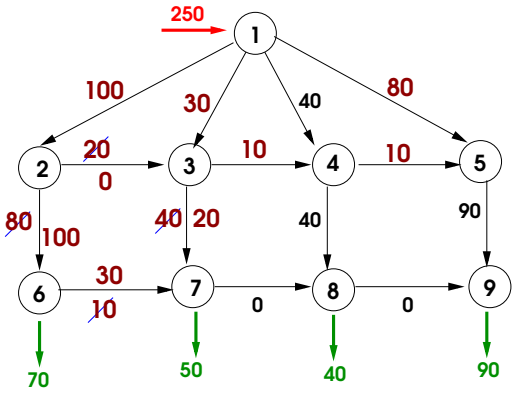
The interpretation of such a cycle C is as follows:

a. Suppose we take any number $\theta > 0$ such that $x_{ij}+\theta \le u_{ij}$ for any foward arc (i,j) and $x_{ij} \ge \theta$ for any backward arc (i,j). Then increasing the flow on any forward arc by θ, and decreasing the flow on any backward arc by θ, and leaving the arcs not in θ unchanged, yields a new feasible solution. (Can you see why?) Let us call the new solution y.

b. The cost of solution y equals the cost of x, plus the value

$$\theta \left(\sum_{(i,j)\,\text{forward}} c_{ij} \; - \sum_{(i,j)\,\text{backward}} c_{ij} \right) \; < \; 0.$$

In other words, y is a better solution than x! Clearly, we should choose θ as large as possible while satisfying condition **a**. In the example above, using the cycle $2-6-7-3-2$. As a result $\theta = 20$, and we get the (cheaper) solution:



Here we have crossed out the old flows. The cost of this solution is 157490 ($= 158130 + 20 \times (200-232)$) , so shis solution is still not optimal – in fact we can find negative cost augmenting cycles in it, as well. By repeating this process we obtain an algorithm that will have finite termination.

Flow decomposition

The figure above shows the solution to the two-level inventory problem – however we would rather visualize the solution in terms of *routes* indicating flow of material from the source (node 1) to the destinations (node 6 through 9). This is accomplished through the concept of *flow decomposition*.

For example, the above solution can be decomposed into the following routes:

1. Route $1-2-6$, carrying 70 units of flow.
2. Route $1-2-6-7$, carrying 30 units of flow.
3. Route $1-3-7$, carrying 20 units of flow.
4. Route $1-4-8$, carrying 40 units of flow.
5. Route $1-5-9$, carrying 90 units of flow.

You can verify that this routing of product meets all demands, and that the total flow on every arc is exactly as shown on the figure above. For example, arc $(2,6)$, which appears in the first two routes, carries a total of $70+40=100$ units of flow.

To develop the general principle, consider a generic min-cost flow problem,

where as before we assume that $\sum_i b_i = 0$. Suppose somebody has given as a *feasible solution* $\hat{x}$ for this problem. A *flow decomposition* of $\hat{x}$ consists of a family of directed paths (the routes) $P_1, P_2, \ldots, P_K$ and for each route P_k a value $v_k > 0$, such that:

(a) Each route starts at a source and ends at a destination.

(b) For any source node i the sum of all the values v_k over the routes P_k that *start at* i equals precisely b_i.

(c) For any destination node i the sum of all the values v_k over the routes P_k that *end at* i equals precisely $-b_i$.

(d) For any arc (i,j), the sum of the values v_k over the routes P_k that use (i,j) equals precisely $hatx_{ij}$.

We have seen how (d) applies in the example above. Continuing with the same example, note that the second and third routes end at node 7, and that altogether they carry $50 = -b_7$ units of flow. It is easy to verify that (a)-(d) apply to the five-route family we constructed above.

Here is a simple algorithm that produces a flow decomposition. At the start of the algorithm we initialize data as follows:

- $\bar{b}_i = b_i$ for every node i,
- $\bar{x}_{ij} = \hat{x}_{ij}$ for ever arc (i,j).

Then we repeat the following step:

(a) Find a directed path P from any node s with $\hat{b}_s > 0$ to any node t with $\hat{b}_t < 0$ and $\bar{x}_{ij} > 0$ for every arc $(i,j) \in P$.

(b) Let $\Delta = \min\{b_s, -b_t, \min_{(i,j)\in P} \bar{x}_{ij}\}$. This is a positive amount by construction. The route P with the value Δ will be one of those used in the flow decomposition.

(c) We reset $\bar{b}_s \leftarrow \bar{b}_s - \Delta$, $\bar{b}_t \leftarrow \bar{b}_t + \Delta$, and $\bar{x}_{ij} \leftarrow \bar{x}_{ij} - \Delta$ for every arc $(i,j) \in P$.

This step is repeated until $\bar{b}_i = 0$ for all nodes i. This point will be reached in at most $n+m$ iterations of steps (a)-(c) above, where n is the number of nodes and m is the number of arcs. Let's see this in action.

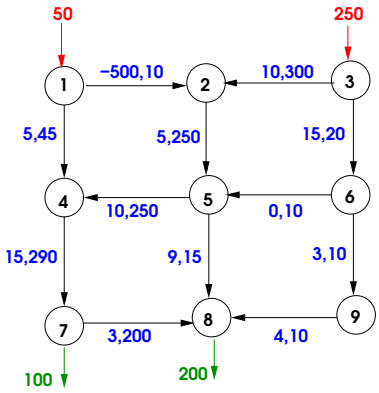
Consider the following min-cost flow problem:

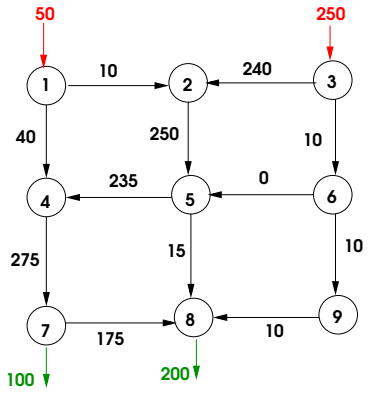
An optimal solution is indicated here:

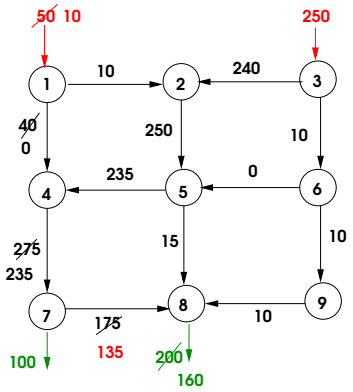
Now we will run the flow decomposition algorithm on this flow. The fact that the solution is *optimal* is actually irrelevant to the procedure, by the way – it applies to *any* feasible solution.

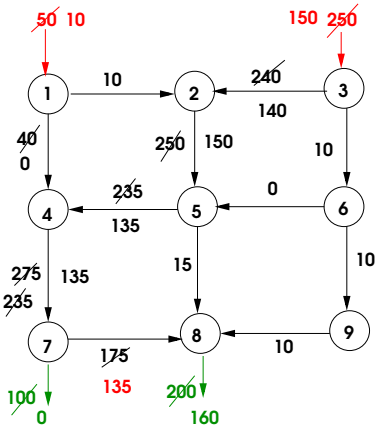
First iteration. We discover the route 1-4-7-8. Here $\Delta = \min\{50, 200, 40\} = 40$ (why?). After updating the solution we have:

Second iteration. We discover the route 3-2-5-4-7. So $\Delta = \min\{250, 100, 235\} = 100$ (why?). After updating the solution we have:

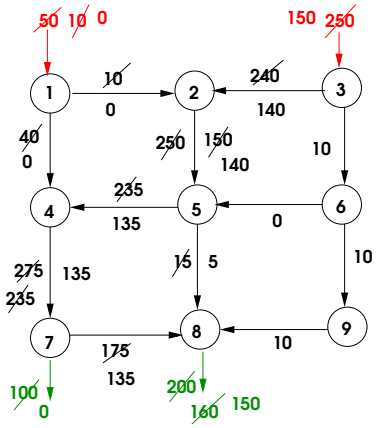

50
250
1
2
3
−500,10
10,300
5,45
5,250
15,20
4
5
6
10,250
0,10
15,290
9,15
3,10
7
8
9
3,200
4,10
100
200


50
250
1
2
3
10
240
40
250
10
4
5
6
235
0
275
15
10
7
8
9
175
10
100
200

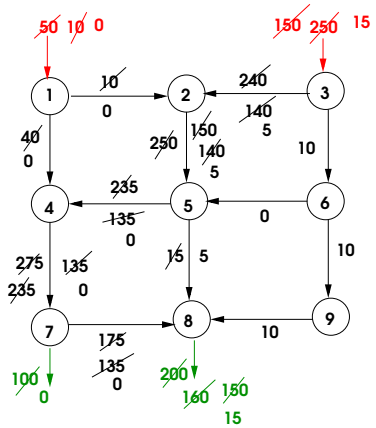

50 10
250
1
2
3
10
240
40
0
250
10
4
5
6
235
0
275
235
15
10
7
8
9
175
135
10
100
200
160


50 10
150 250
1
2
3
10
240
140
40
0
250
150
10
4
5
6
235
135
0
275
235
135
15
10
7
8
9
175
135
10
100
0
200
160

Third iteration. We discover the route 1-2-5-8. So $\Delta = \min\{10, 160, 10\} = 10$ (why?). After updating the solution we have:



Fourth iteration. We discover the route 3-2-5-4-7-8, with $\Delta = \min\{150, 150, 135\} = 135$ (why?). After updating the solution we have:



Fifth iteration. We discover the route 3-2-5-8, with $\Delta = 5$.

Sixth iteration. We discover the route 3-6-9-8, with $\Delta = 10$.