Applications of Linear Programming

lecturer: András London

University of Szeged
Institute of Informatics
Department of Computational Optimization

Lecture 6
based on Juraj Stacho's lecture notes ad the Columbia university

Graphs

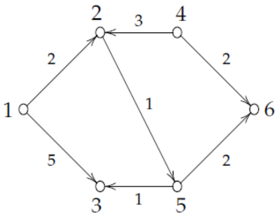
G = **graph** or **network** consists of

- a set V of **vertices** (nodes, points) and
- a set E of **edges** (arcs, lines) which are connections between vertices.

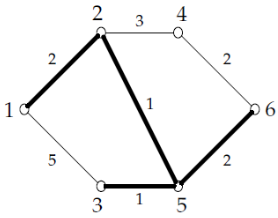
write $G = (V, E)$; write $V(G)$ for vertices of G, and $E(G)$ for edges of G.

(vertices are usually denoted u or v with subscripts; edges we usually denote e)

edges may have **direction**: an edge e between u and v may go from u to v, we write $e = (u, v)$,



vertices $V = \{1, 2, 3, 4, 5, 6\}$
edges $E = \Big\{(1,2), (1,3), (2,5), (4,2), (4,6), (5,3), (5,6)\Big\}$
weights $c(1,2) = 2$ $c(1,3) = 5$
$c(2,5) = 1$ $c(4,2) = 3$ $c(4,6) = 2$
$c(5,3) = 1$ $c(5,6) = 2$



Graphs

if all edges do not have a direction (are undirected), we say that the network is **undirected**

edges may have **weight**: a weight of edge $e = (u, v)$ is a real number denoted $c(e)$ or $c(u, v)$, c_e, c_{uv}

a sequence of nodes and edges $v_1, e_1, v_2, e_2, \ldots v_{k-1}, e_k, v_k$ is

- a **path** (directed path) if each e_i goes from v_i to v_{i+1}
- a **chain** (undirected path) if each e_i connects v_i and v_{i+1} (in some direction)

(often we write: $e_1, e_2, \ldots, e_k$ is a path (we omit vertices) or write: $v_1, v_2, \ldots, v_k$ is a path (we omit edges))

a network is **connected** if for every two nodes there is a path connecting them; otherwise it is **disconnected**

a **cycle** (loop, circuit) is a path starting and ending in the same node, never repeating any node or edge

a **forest** (acyclic graph) is an undirected graph that contains no cycles

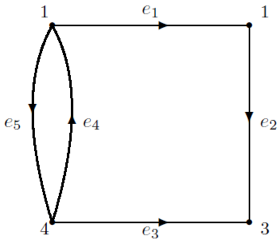
a **tree** is a connected forest

Claim: A tree with n nodes contains exactly $n - 1$ edges. Adding any edge to a tree creates a cycle. Removing any edge from a tree creates a disconnected forest.

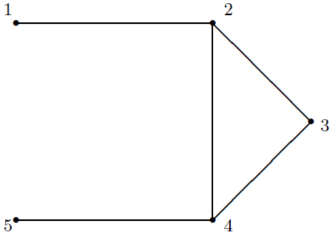
Graph representations

- Graphs can be represented by matrices. For us the most important ones are the
  - incidence matrix
  - adjacency matrix

	e_1	e_2	e_3	e_4	e_5
1	-1	0	0	1	-1
2	1	-1	0	0	0
3	0	1	1	0	0
4	0	0	-1	-1	1



	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	1
5	0	0	0	1	0



A Graph's incidence matrix is TU

Theorem. *The incidence matrix of a directed graph is totally unimodular.*

Proof. Easy to see by induction (according to the size of subdeterminants).

Note: Combining this theorem with that we learnt in Lecture 5 we can see that if an IP is given by an incidence matrix (of a graph) is simplified to an LP problem. This makes easier to solve the problem (can be solved in polynomial time), moreover the strength of duality can be utilized.

Brief history and motivation

The Shortest Path Problem is one of the most important efficient computational tools at this disposal

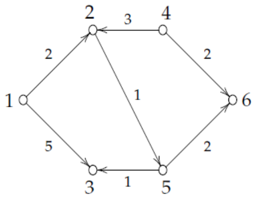
- It is used everywhere, from GPS navigation and network communication to project management, layout design, robotics, computer graphics, and the list goes on
- First algorithms for the Shortest Path problem were designed by Ford (1956), Bellman (1958), and Moore (1959). For non-negative weights, a more efficient algorithm was first suggested by Dijkstra (1959).
- All-pairs Shortest Path problem the first algorithms were founds by Shimbel (1953), and by Roy (1959), Floyd (1962), and Warshall (1962)

Brief history and motivation

The Minimum Spanning Tree problem also has a rich history

- The first known algorithm was developed by Boruvka (1926) for efficient distribution of electricity
- Later independently discovered by many researchers over the years: Jarnik (1930), Kruskal (1956), Prim(1957), and Dijkstra (1959)

Shortest path problem



vertices $V = \{1,2,3,4,5,6\}$
edges $E = \Big\{(1,2),(1,3),(2,5),(4,2),$
$(4,6),(5,3),(5,6)\Big\}$
weights $c(1,2)=2 \quad c(1,3)=5$
$c(2,5)=1 \quad c(4,2)=3 \quad c(4,6)=2$
$c(5,3)=1 \quad c(5,6)=2$

Given a network $G=(V,E)$ with two distinguished vertices $s,t \in V$, find a shortest path from s to t

Example: In Figure 1 (left), a shortest path from $s=1$ to $t=6$ is $1,2,5,6$ of total length 5, while for $t=3$ a shortest path is $1,2,5,3$ of length 4. We say that **distance** from node 1 to node 6 is 5. Note that there is no path from s to $t=4$; we indicate this by defining the distance to 4 as ∞.

LP formulation: decision variables x_{ij} for each $(i,j) \in E$

$$\text{Min} \quad \sum_{(i,j)\in E} w_{ij}x_{ij}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & \text{if } i=s \\ -1 & \text{if } i=t \\ 0 & \text{otherwise} \end{cases} \qquad \text{for each } i \in V$$

$$x_{ij} \in \{0,1\} \qquad \text{for each } (i,j) \in E$$

Dijksra's algorithm

Algorithm finds the length of a shortest path from s to every vertex of G (not only t)

Weights of edges are assumed to be **non-negative**, else the algorithm may output incorrect answer.

variables: d_u for each $u \in V$, an **estimate** on the distance from s to u

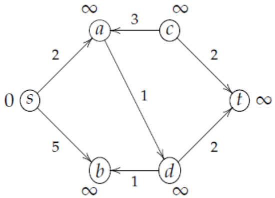
initialize: $d_u = \begin{cases} 0 & \text{if } u = s \\ \infty & \text{otherwise} \end{cases}$ all vertices are initially *unprocessed*

1. Find an *unprocessed* vertex u with smallest d_u
2. For each $(u, v) \in E$, update $d_v = \min\{d_v, d_u + c_{uv}\}$
3. Mark u as processed; repeat until all vertices are processed.
4. Report d_t as distance from s to t

Dijksra's algorithm

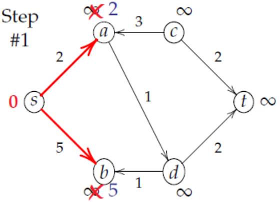
1. Find an *unprocessed* vertex u with smallest d_u
2. For each $(u, v) \in E$, update $d_v = \min\{d_v,\, d_u + c_{uv}\}$
3. Mark u as processed; repeat until all vertices are processed.
4. Report d_t as distance from s to t

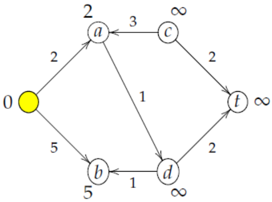
Example:



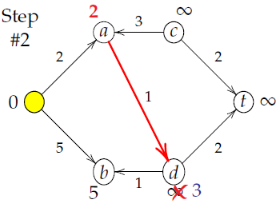
Step#	s	a	b	c	d	t
1.	$\underline{0}$	∞	∞	∞	∞	∞
		2	5			
2.	0^*	$\underline{2}$	5	∞	∞	∞
					3	
3.	0^*	2^*	5	∞	$\underline{3}$	∞
			4			5
4.	0^*	2^*	$\underline{4}$	∞	3^*	5
5.	0^*	2^*	4^*	∞	3^*	$\underline{5}$
6.	0^*	2^*	4^*	$\underline{\infty}$	3^*	5^*
final	0^*	2^*	4^*	∞^*	3^*	5^*

Step #1

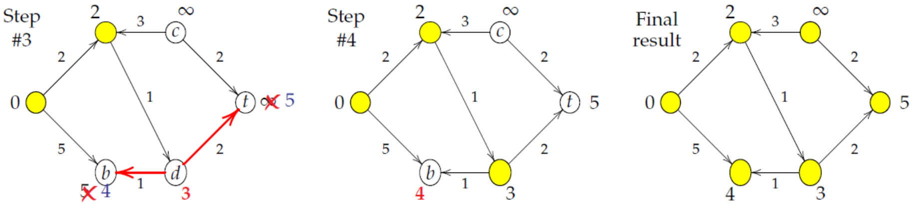




Step #2



Dijksra's algorithm



We can read the shortest path from 1 to 6: that is path 1,2,5,6.

Minimum spanning tree

A power company delivers electricity from its power plant to neighbouring cities. The cities are interconnected by power lines operated by various operators. The power company wants to rent power lines in the grid of least total cost that will allow it to send electricity from its power plant to all cities.

Given an undirected network $G = (V, E)$ find a collection $F \subseteq E$ of minimum weight so that (V, F) is a tree.

(we say that (V, F) is a **spanning** tree because it spans all vertices)

Kruskal's (Prim's) algorithm

initialize: F to be empty; all edges are initially *unprocessed*

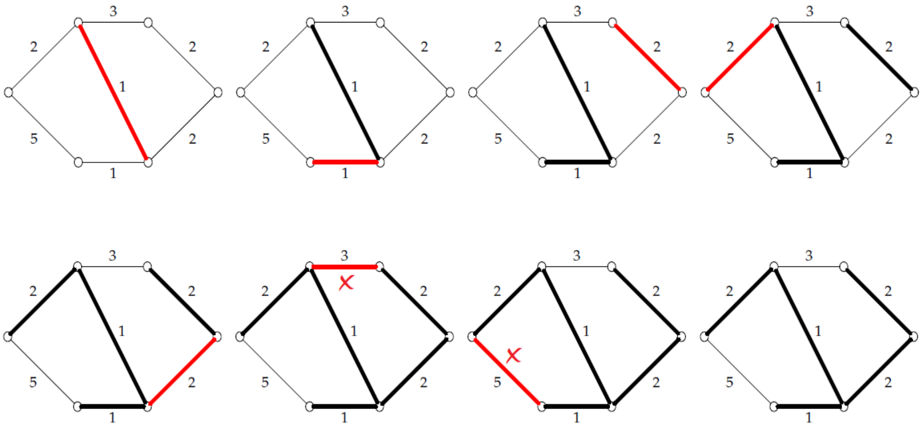
Kruskal's algorithm:

1. Find an unprocessed edge e of smallest weight w_e.
2. If $(V, F \cup \{e\})$ is a forest, then add e to F.
3. Mark e as processed and repeat until all edges have been processed.
4. Report (V, F) as a minimum-weight spanning tree.

Prim's algorithm: replace 1 by 1′

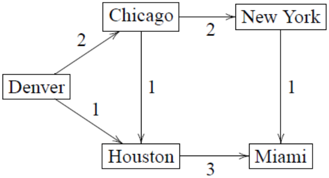
1′ Find an unprocessed edge e of smallest weight that shares an endpoint with some edge in F

Minimum spanning tree - Kruskal's algorithm



Maximum 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). A customer is interested in hiring the company to deliver his products daily from Denver to Miami, and needs to know how much product can be delivered on a daily basis.



maximize z

$$
\underbrace{\begin{array}{rrrrrrrcr}
-x_{DC} & -x_{DH} & & & & & & = & -z \\
x_{DC} & & -x_{CH} & -x_{CN} & & & & = & 0 \\
& x_{DH} & +x_{CH} & & & & -x_{HM} & = & 0 \\
& & & x_{CN} & -x_{NM} & & & = & 0 \\
& & & & x_{NM} & & +x_{HM} & = & z
\end{array}}_{\text{conservation of flow}}
$$

$$
\begin{array}{l}
0 \leq x_{DC} \leq 2 \\
0 \leq x_{DH} \leq 1 \\
0 \leq x_{CH} \leq 1 \\
0 \leq x_{CN} \leq 2 \\
0 \leq x_{NM} \leq 1 \\
0 \leq x_{HM} \leq 3
\end{array}
$$

Maximum flow problem

In general, network $G = (V, E)$:

$s =$ **source** (Denver)

$t =$ **sink** (Miami)

u_{ij} =*capacity* of an edge ij (# trucks dispatched daily between i and j)

x_{ij} =*flow* on an edge ij (# trucks delivering the customer's products)

$$\max \ z$$

$$\underbrace{\sum_{\substack{j\in V\\ ji\in E}} x_{ji}}_{\text{flow into } i} - \underbrace{\sum_{\substack{j\in V\\ ij\in E}} x_{ij}}_{\text{flow out of } i} = \begin{cases} -z & i = s \\ z & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for all } ij \in E$$

Maximum flow problem - The Ford-Fulkerson algorithm

In general, network $G = (V, E)$:

$s =$ **source** (Denver)

$t =$ **sink** (Miami)

u_{ij} =*capacity* of an edge ij (# trucks dispatched daily between i and j)

x_{ij} =*flow* on an edge ij (# trucks delivering the customer's products)

$$\max \ z$$

$$\underbrace{\sum_{\substack{j\in V\\ ji\in E}} x_{ji}}_{\text{flow into } i} - \underbrace{\sum_{\substack{j\in V\\ ij\in E}} x_{ij}}_{\text{flow out of } i} = \begin{cases} -z & i = s \\ z & i = t \\ 0 & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{for all } ij \in E$$

Maximum flow problem - The Ford-Fulkerson algorithm

Initial feasible flow $x_{ij} = 0$ for all $ij \in E$.

A sequence of nodes $v_1, v_2, \ldots, v_n$ is a *chain* if $v_i v_{i+1} \in E$ (*forward* edge) or $v_{i+1} v_i \in E$ (*backward* edge) for all $i = 1, \ldots, n-1$. If $v_1 = s$ and $v_n = t$, then we call it an (s,t)-chain. Consider an (s,t)-chain P.

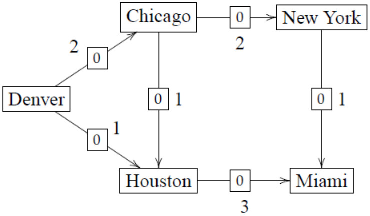
The *residual capacity* of a forward edge ij on P is defined as $u_{ij} - x_{ij}$ (the remaining capacity on the edge ij). The *residual capacity* of a backward edge ij on P is defined as x_{ij} (the used capacity of the edge ij).
The *residual capacity* of P is the **minimum** taken over residual capacities of edges on P.

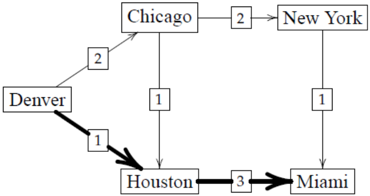
If the *residual capacity* of P is positive $\varepsilon > 0$, then P is an **augmenting chain**. If this happens, we can increase the flow by increasing the flow on all forward edges by ε, and decreasing the flow on all backward edges by ε. This yields a feasible flow of larger value $z + \varepsilon$. (Notice the similarity with the Transportation Problem and the ratio test in the Simplex Method – same thing in disguise.)

Optimality criterion: The flow x_{ij} is optimal if and only if there is no augmenting chain.

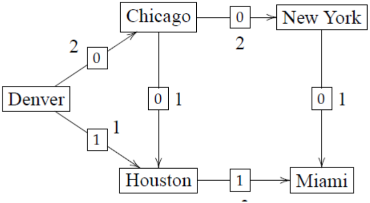
Maximum flow problem - The Ford-Fulkerson algorithm

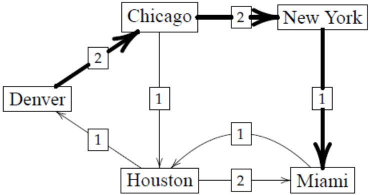
Starting feasible flow $x_{ij} = 0$ (indicated in boxes) $\rightarrow$ residual network (residual capacity shown on edges)





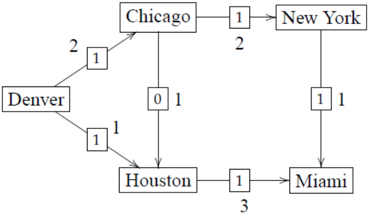
augmenting chain of residual capacity 1 $\rightarrow$ increase flow by 1

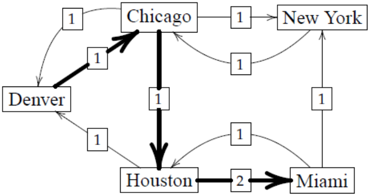




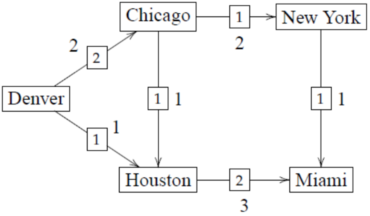
Maximum flow problem - The Ford-Fulkerson algorithm

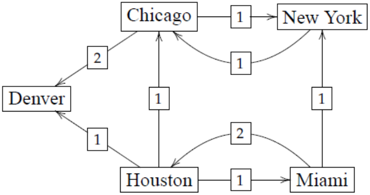
augmenting chain of residual capacity 1 → increase flow by 1





augmenting chain of residual capacity 1 → increase flow by 1





no path from Denver to Miami in the residual network → no augmenting chain → **optimal solution** found

→ maximum flow has value 3

Maximum flow problem - Minimum cut

Minimum Cut

For a subset of vertices $A \subseteq V$, the edges going between the nodes in A and the rest of the graph is called a **cut**. We write $(A, \overline{A})$ to denote this cut. The edges going out of A are called **forward edges**, the edges coming into A are **backward edges**. If A contains s but not t, then it is an (s,t)-cut.

The **capacity** of a cut $(A, \overline{A})$ is the sum of the capacities of its forward edges.

For example, let $A = \{\text{Denver,Chicago}\}$. Then $(A, \overline{A})$ is an (s,t)-cut of capacity 4. Similarly, let $A_* = \{$Denver, Chicago, New York$\}$. Then $(A_*, \overline{A_*})$ is an (s,t)-cut of capacity 3.

Theorem. (Max. flow - Min. cut) *The maximum value of an (s,t)-flow is equal to the minimum capacity of an (s,t)-cut.*

Maximum flow problem - Minimum cut

This is known as the Max-Flow-Min-Cut theorem – a consequence of strong duality of linear programming.

maximize z

$$
\begin{array}{rcl}
-x_{DC}-x_{DH} & = & -z\\
x_{DC}-x_{CH}-x_{CN} & = & 0\\
x_{DH}+x_{CH}-x_{HM} & = & 0\\
x_{CN}-x_{NM} & = & 0\\
x_{NM}+x_{HM} & = & z
\end{array}
\qquad
\begin{array}{l}
0\le x_{DC}\le 2\\
0\le x_{DH}\le 1\\
0\le x_{CH}\le 1\\
0\le x_{CN}\le 2\\
0\le x_{NM}\le 1\\
0\le x_{HM}\le 3
\end{array}
$$

Dual:

minimize $2v_{DC}+v_{DH}+v_{CH}+2v_{CN}+v_{NM}+3v_{HM}$

$$
\begin{array}{rcl}
y_D-y_C & \le & v_{DC}\\
y_D-y_H & \le & v_{DH}\\
y_C-y_H & \le & v_{CH}\\
y_C-y_N & \le & v_{CN}\\
y_N-y_M & \le & v_{NM}\\
y_H-y_M & \le & v_{HM}\\
y_D-y_M & \ge & 1
\end{array}
$$

$v_{DC},v_{DH},v_{CH},v_{CN},v_{NM},v_{HM}\ge 0$

y_D,y_C,y_H,y_N,y_M unrestricted

Optimal solution (of value 3)

$y_D=y_C=y_N=1$ $\quad\rightarrow\quad$ $A=\{D,C,N\}$

$y_H=y_M=0$ $\quad$ min-cut

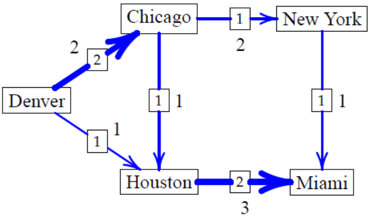
$v_{DH}=v_{CH}=v_{NM}=1$

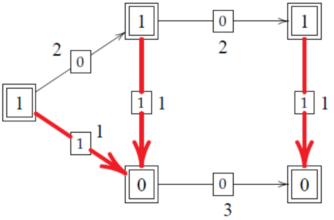
$v_{DC}=v_{CN}=v_{HM}=0$

$\rightarrow$ given an optimal solution, let A be the nodes whose y value is the same as that of source

$\rightarrow (A,\overline{A})$ **minimum cut**

Maximum flow problem - Minimum cut



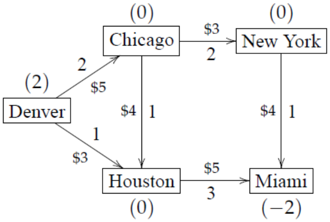


$$
\begin{aligned}
\max\ & z \\
& \sum_{\substack{j \in V \\ ji \in E}} x_{ji} - \sum_{\substack{j \in V \\ ij \in E}} x_{ij} = \begin{cases} -z & i = s \\ z & i = t \\ 0 & \text{otherwise} \end{cases} \\
& 0 \le x_{ij} \le u_{ij} \qquad \text{for all } ij \in E
\end{aligned}
$$

$$
\begin{aligned}
\min \sum_{ij \in E} & u_{ij} v_{ij} \\
y_i - y_j &\le v_{ij} && \text{for all } ij \in E \\
y_s - y_t &\ge 1 \\
v_{ij} &\ge 0 && \text{for all } ij \in E \\
y_i &\text{ unrestricted} && \text{for all } i \in V
\end{aligned}
$$

Minimum 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

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

$$\underbrace{\begin{array}{rcl} x_{DC}+x_{DH} & = & 2 \\ -x_{DC} \qquad +x_{CH}+x_{CN} & = & 0 \\ -x_{DH}-x_{CH} \qquad\qquad +x_{HM} & = & 0 \\ -x_{CN}+x_{NM} & = & 0 \\ -x_{NM}-x_{HM} & = & -2 \end{array}}_{\text{conservation of flow}} \qquad \begin{array}{l} 0 \leq x_{DC} \leq 2 \\ 0 \leq x_{DH} \leq 1 \\ 0 \leq x_{CH} \leq 1 \\ 0 \leq x_{CN} \leq 2 \\ 0 \leq x_{NM} \leq 1 \\ 0 \leq x_{HM} \leq 3 \end{array}$$

Minimum cost flow problem

Network $G = (V, E)$:

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)

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

$$\underbrace{\sum_{\substack{j\in V\\ ij\in E}} x_{ij}}_{\text{flow out of } i} - \underbrace{\sum_{\substack{j\in V\\ ji\in E}} x_{ji}}_{\text{flow into } i} = \underbrace{b_i}_{\text{net supply}}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for all } ij \in E$$

Necessary condition: $\sum_i b_i = 0$.

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

Summary

Network $G = (V, E)$ has **nodes** V and **edges** E.

- Each edge $(i, j) \in E$ has a **capacity** u_{ij} and **cost** c_{ij}.
- Each vertex $i \in V$ provides **net supply** b_i.

For a set $S \subseteq V$, write $\overline{S}$ for $V \setminus S$ and write $E(S, \overline{S})$ for the set of edges $(i, j) \in E$ with $i \in S$ and $j \in \overline{S}$. The pair $(S, \overline{S})$ is called a **cut**. (Where applicable) there are two distinguished nodes: s =**source** and t =**sink**.

Minimum spanning tree

Primal

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

$$\underbrace{\sum_{(i,j)\in E(S,\overline{S})}}_{\text{edges from } S \text{ to } \overline{S}} x_{ij} > 0 \qquad \text{for all } S \subseteq V \text{ where } \emptyset \neq S \neq V$$

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

Obstruction (to feasibility):
set $S \subseteq V$ with $\emptyset \neq S \neq V$ such that $E(S, \overline{S}) = \emptyset$

Summary

Shortest path problem

Primal

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

$$\underbrace{\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij}}_{\text{flow out of } i} - \underbrace{\sum_{\substack{j\in V\\(j,i)\in E}} x_{ji}}_{\text{flow into } i} = \begin{cases} 1 & i=s \\ -1 & i=t \\ 0 & \text{else} \end{cases} \quad \text{for all } i \in V$$

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

Dual

$$\max y_s - y_t$$

$$y_i - y_j \leq c_{ij} \quad \text{for all } (i,j)\in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i \in V$$

Obstruction (to feasibility): set $S \subseteq V$ with $s \in S$ and $t \in \overline{S}$ such that $E(S,\overline{S}) = \emptyset$

Summary

Maximum-flow problem

Primal

$$\max\ z$$

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji} = \begin{cases} z & i=s \\ -z & i=t \\ 0 & \text{else} \end{cases} \quad \text{for all } i\in V$$

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

$$z \text{ unrestricted}$$

Dual

$$\min \sum_{(i,j)\in E} u_{ij}v_{ij}$$

$$y_i - y_j + v_{ij} \ge 0 \quad \text{for all } (i,j)\in E$$

$$y_t - y_s = 1$$

$$v_{ij} \ge 0 \quad \text{for all } (i,j)\in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i\in V$$

Obstruction (to feasibility): set $S\subseteq V$ with $s\in S$ and $t\in\overline{S}$ such that $z > \underbrace{\sum_{(i,j)\in E(S,\overline{S})} u_{ij}}_{\text{capacity of the cut } (S,\overline{S})}$

(no flow bigger than the capacity of a cut)

Summary

Minimum-cost (s,t)-flow problem

Primal

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

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji} = \begin{cases} f & i=s \\ -f & i=t \\ 0 & \text{else} \end{cases} \quad \text{for all } i\in V$$

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

Dual

$$\max f y_s - f y_t - \sum_{(i,j)\in E} u_{ij}v_{ij}$$

$$y_i - y_j - v_{ij} \le c_{ij} \quad \text{for all } (i,j)\in E$$

$$v_{ij} \ge 0 \quad \text{for all } (i,j)\in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i\in V$$

Obstruction (to feasibility): set $S\subseteq V$ with $s\in S$ and $t\in\overline{S}$ such that $f > \sum_{(i,j)\in E(S,\overline{S})} u_{ij}$

Summary

Transshipment problem

Primal

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

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji} = \underbrace{b_i}_{\text{net supply}} \quad \text{for all } i \in V$$

$$x_{ij} \geq 0 \quad \text{for all } (i,j) \in E$$

Dual

$$\max \sum_{i\in V} b_i y_i$$

$$y_i - y_j \leq c_{ij} \quad \text{for all } (i,j) \in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i \in V$$

Obstruction (to feasibility): set $S \subseteq V$ such that $\sum_{i\in S} b_i > 0$ and $E(S,\overline{S}) = \emptyset$

Minimum-cost network flow problem

Primal

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

$$\sum_{\substack{j\in V\\(i,j)\in E}} x_{ij} - \sum_{\substack{j\in V\\(j,i)\in E}} x_{ji} = b_i \quad \text{for all } i \in V$$

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

Dual

$$\max \sum_{i\in V} b_i y_i - \sum_{(i,j)\in E} u_{ij}v_{ij}$$

$$y_i - y_j - v_{ij} \leq c_{ij} \quad \text{for all } (i,j) \in E$$

$$v_{ij} \geq 0 \quad \text{for all } (i,j) \in E$$

$$y_i \text{ unrestricted} \quad \text{for all } i \in V$$

Obstruction (to feasibility): set $S \subseteq V$ such that $\sum_{i\in S} b_i > \sum_{(i,j)\in E(S,\overline{S})} u_{ij}$

Example #1

A new car costs $\$12,000$. Annual maintenance costs are as follows: $m_1 = \$2,000$ first year, $m_2 = \$4,000$ second year, $m_3 = \$5,000$ third year, $m_4 = \$9,000$ fourth year, and $m_5 = \$12,000$ fifth year and on. The car can be sold for $s_1 = \$7,000$ in the first year, for $s_2 = \$6,000$ in the second year, for $s_3 = \$2,000$ in the third year, and for $s_4 = \$1,000$ in the fourth year of ownership.

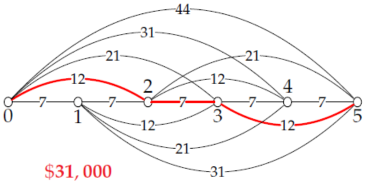
An existing car can be sold at any time and another new car purchased at $\$12,000$. What buying/selling strategy for the next 5 years minimizes the total cost of ownership?

Nodes = $\{0, 1, 2, 3, 4, 5\}$

Edge (i, j) represents the act of buying a car in year i and selling in year j. The weight is the price difference plus the maintanence cost, i.e., the weight is

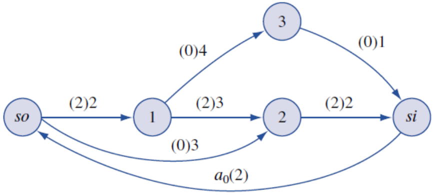
$$c(i,j) = \$12,000 - s_{(i-j)} + m_1 + m_2 + \ldots + m_{(i-j)}$$

Answer: the length of a shortest path from node 0 to node 5.



Example #2

Sunco Oil wants to ship the maximum possible amount of oil (per hour) via pipeline from node so to node si in Figure. On its way from node so to node si, oil must pass through some or all of stations 1, 2, and 3. The various arcs represent pipelines of different diameters. The maximum number of barrels of oil (millions of barrels per hour) that can be pumped through each arc is shown in the Table. Each number is called an arc capacity. Formulate an LP that can be used to determine the maximum number of barrels of oil per hour that can be sent from so to si.



Arc	Capacity
$(so, 1)$	2
$(so, 2)$	3
$(1, 2)$	3
$(1, 3)$	4
$(3, si)$	1
$(2, si)$	2

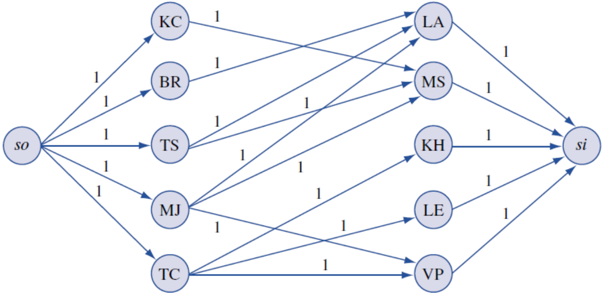
Example #3

Five male and five female entertainers are at a dance. The goal of the matchmaker is to match each woman with a man in a way that maximizes the number of people who are matched with compatible mates. Table describes the compatibility of the entertainers. Draw a network that makes it possible to represent the problem of maximizing the number of compatible pairings as a maximum-flow problem.

	Loni Anderson	Meryl Streep	Katharine Hepburn	Linda Evans	Victoria Principal
Kevin Costner	—	C	—	—	—
Burt Reynolds	C	—	—	—	—
Tom Selleck	C	C	—	—	—
Michael Jackson	C	C	—	—	C
Tom Cruise	—	—	C	C	C

Note: C indicates compatibility.

Example #3



the arc joining each woman to the sink has a capacity of 1, conservation of flow ensures that each woman will be matched with at most one man. Similarly, because each arc from the source to a man has a capacity of 1, each man can be paired with at most one woman. Because arcs do not exist between noncompatible mates, we can be sure that a flow of k units from source to sink represents an assignment of men to women in which k compatible couples are created.