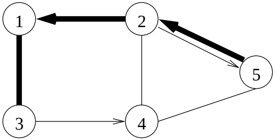
MVE165/MMG631
Linear and integer optimization with applications
Lecture 11
Shortest paths and network flows; linear programming formulations of flows in networks

Ann-Brith Strömberg

2014–05–06

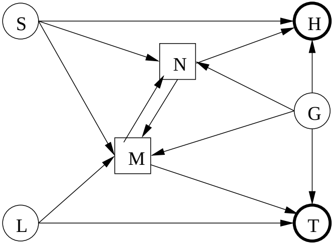
Flows in networks, in particular shortest paths

A path from node 5 to node 3



A flow network

- Supply nodes: S, G, L
- Demand nodes: H, T
- Storage: M, N
- Limited capacities on links
- Minimize costs for transport and storage



Many different problems can be formulated as graph or network flow models:

- Find the total capacity of a given water pipeline network
- Find a time schedule (starting and completion times) for the activities in a project
- How much goods should be transported from each supplier to each point of demand in a transportation system, and which links should be used to what extent

A linear programming formulation: shortest path from node $s \in N$ to node $t \in N$

- For each arc $(i,j) \in A$, let x_{ij} be the flow on the arc
- *Flow balance in each node $k \in N$*
- $x_{ij} = 1$ if arc (i,j) is in the shortest path and $x_{ij} = 0$ otherwise
- Linear programming formulation (assume $d_{ij} \geq 0$):

$$\begin{aligned}
\min \quad & \sum_{(i,j) \in A} d_{ij} x_{ij}, \\
\text{s.t.} \quad & \sum_{i:(i,k) \in A} x_{ik} - \sum_{j:(k,j) \in A} x_{kj} = \begin{cases} -1, & k = s, \\ 1, & k = t, \\ 0, & k \in N \setminus \{s,t\}, \end{cases} \\
& x_{ij} \geq 0, \quad (i,j) \in A.
\end{aligned}$$

- Linear programming dual:

$$\begin{aligned}
\max \quad & y_t - y_s, \\
\text{s.t.} \quad & y_j - y_i \leq d_{ij}, \quad (i,j) \in A \\
& y_k \quad \text{free}, \quad k \in N
\end{aligned}$$

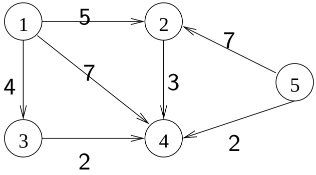
The shortest path problem (Ch. 8.4)

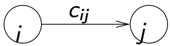
- Given: a network of nodes N, (directed) arcs A, and arc distances d_{ij}, $(i, j) \in A$

- Find the shortest path from a source node ($s \in N$) to a destination node ($t \in N$)

Principle of optimality formulated by Bellman's equations

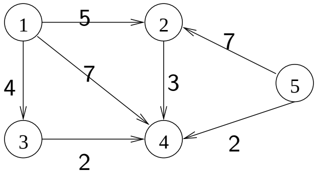
- In a graph with *no negative cycles*, optimal paths have optimal subpaths
- A shortest path from node s node to t that passes through node k contains a shortest path from node s node to k
- Let y_j denote the length of the shortest path from node s to node j
- Bellmans equations:
  - $y_s = 0$
  - $y_j = \min_i \left\{ y_i + c_{ij} : \text{ arc/edge } (i, j) \text{ exists } \right\}$ for all $j \neq s$

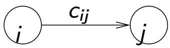




Solution method I: Bellman's equations

- If the graph is directed without cycles: solve Bellman's equations in topological order
- Shortest path from node 1 to each of the other nodes (1,5,2,3,4):
  - $y_1 = 0$
  - $y_5 = \min\{\infty\} = \min\{\infty\} = \infty$
  - $y_2 = \min\{\infty; y_1 + c_{12}; y_5 + c_{52}\} = \min\{\infty; 0+5; \infty\} = 5$
  - $y_3 = \min\{\infty; y_1 + c_{13}\} = \min\{0+4\} = 4$
  - $y_4 = \min\{\infty; y_1 + c_{14}; y_2 + c_{24}; y_3 + c_{34}; y_5 + c_{54}\} = \min\{\infty; 0+7; 5+3; 4+2; \infty + 2\} = 6$

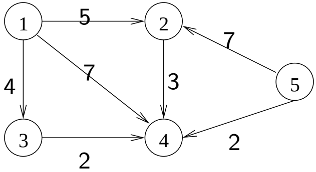


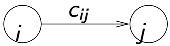


- $y_1 = 0,\ y_2 = 5,\ y_3 = 4,\ y_4 = 6,\ y_5 = \infty$

Solution method II: Dijkstra's algorithm

- The graph may contain cycles but all edge costs must be nonnegative (i.e., $c_{ij} \geq 0$)





- *Solve the example on the board*

Algorithms for the shortest path problem: Dijkstra (Ch.8.4.2)

- Find the shortest path between node s and node i when all arcs distances are non-negative
- $N =$ set of all nodes; source node $s \in N$
- $d_{ij} =$ distance on link from i to j for all $i, j \in N$
- $d_{ij} = \infty$ if no direct link from i to j

Step 0: $S := \{s\}$, $\bar{S} := N \setminus \{s\}$, and $y_i := d_{si}$, $i \in N$

Step 1:

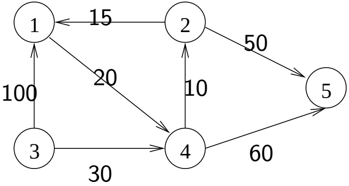
(a) If $\bar{S} = \emptyset$, stop. Else find node j such that $y_j = \min_{i \in \bar{S}} y_i$
$S := S \cup \{j\}$ and $\bar{S} := \bar{S} \setminus \{j\}$

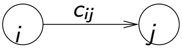
(b) For all $k \in \bar{S}$ and $i \in S$:
If $y_k > y_i + d_{ik}$ set $y_k := y_i + d_{ik}$ and $pred(k) := i$

- The vector *pred* keeps track of the predecessors
- Dijkstra's algorithm actually finds shortest paths from the source to *all* others nodes (this is not formulated in the LP)

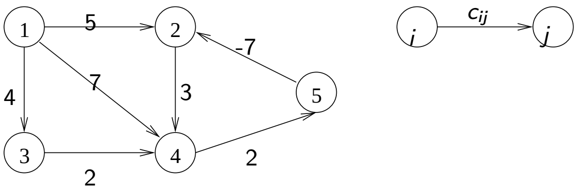
Example: Dijkstra's algorithm

Find the shortest path from node 1 to all other nodes (Homework)





Negative lengths of edges and negative cycles



- Negative length of edges: extend Dijkstra's algorithm according to "move nodes back from S to $\bar{S}$" (Ford's algorithm)
- There may be a cycle of *negative* total length

$\Rightarrow$ "Length" of the shortest path $\rightarrow -\infty$

$\Rightarrow$ Ford's algorithm *either* finds a shortest path *or* detects a cycle with a negative total length

Algorithms for the shortest path problem: Floyd–Warshall (Ch. 8.4.2)

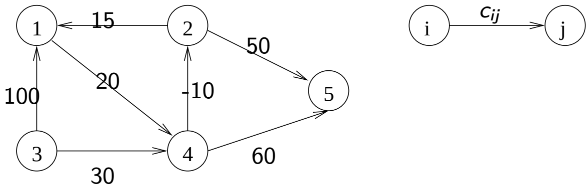
- Computes shortest paths between each pair of nodes
- Negative distances are allowed; negative cycles are detected
- Idea: Three nodes i, k, j and distances c_{ik}, c_{kj}, and c_{ij}
- $i \to k \to j$ is a short-cut if $c_{ik} + c_{kj} < c_{ij}$
- In each iteration $1 \dots k$, check whether c_{ij} can be improved by using the short-cut via k
- Administration of the algorithm: Maintain two matrices per iteration: $D[k]$ for the distances and $pred[k]$ to keep track of the predecessor of each node

Floyd–Warshall's algorithm

Step 0: Initialize $D[0]$ and $pred[0]$

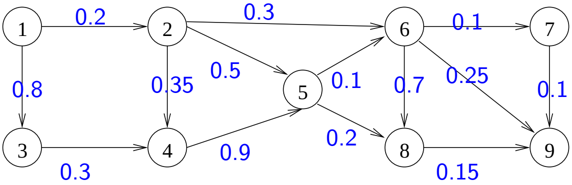
Step k:
- $D[k] := D[k-1]$, $pred[k] := pred[k-1]$
  For each element d_{ij} in $D[k]$:
  If $d_{ik} + d_{kj} < d_{ij}$, set $d_{ij} := d_{ik} + d_{kj}$ and $pred_{ij}[k] := k$
  Set $k := k + 1$
  If $k > n$ stop, else repeat Step k

Find the shortest path from node 3 to all other nodes



Example: Most reliable route

- Mr Q drives to work daily
- All road links he can choose for a path to work are patrolled by the police
- It is possible to assign a probability $p_{ij} \in [0, 1]$ of *not* being stopped by the police on link (i, j)
- Mr Q wants to find the "shortest" (safest?) path in the sense that the probability of being stopped is as low as possible
- maximize $Prob$(not being stopped)



- Ex. $1 \to 4$: $\max\{p_{12}p_{24}; p_{13}p_{34}\} = \max\{0.2 \cdot 0.35; 0.8 \cdot 0.3\}$
- Note: This version *cannot* be formulated as a linear program

Alternative objectives $\Rightarrow$ Variants of Bellman's equations

- Most reliable path (failure probability $p_{ij} \in [0, 1]$ for arc (i, j)):
  - $y_s = 1$
  - $y_j = \max_i \{ y_i \cdot p_{ij} : \text{ arc/edge } (i, j) \text{ exists } \}$ for all $j \neq s$

- Highest capacity path (capacity $K_{ij} \geq 0$ on arc (i, j)):
  - $y_s = \infty$
  - $y_j = \max_i \{ \min\{y_i; K_{ij}\} : \text{ arc/edge } (i, j) \text{ exists } \}, \; j \neq s$