

MVE165/MMG631

Linear and integer optimization with applications

Lecture 11

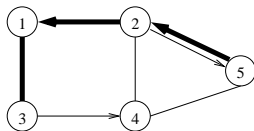
Shortest paths; dynamic programming;
linear programming formulations of network flows

Ann-Brith Strömberg

2019-05-07

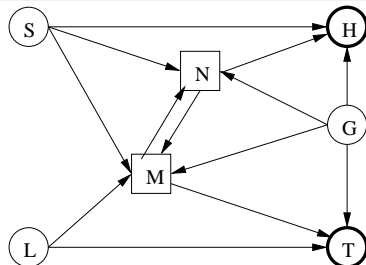
Flows in networks, in particular shortest paths

A path from node 5 to node 3



A flow network

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

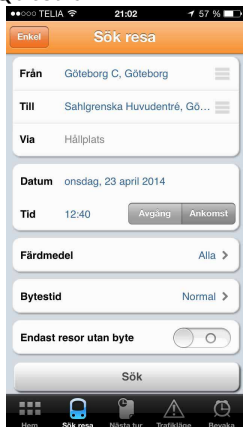


Many different problems can be formulated as graph or network flow models

- 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?
- Find the optimal set of maintenance occasions over a time period for *one* component

A useful application

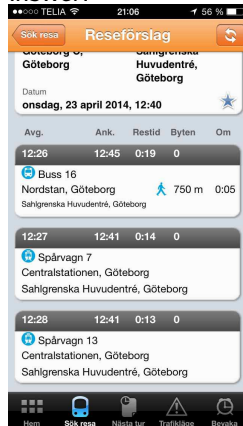
Question:



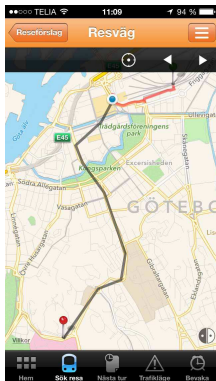
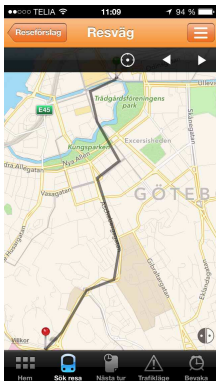
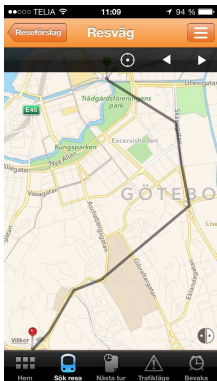
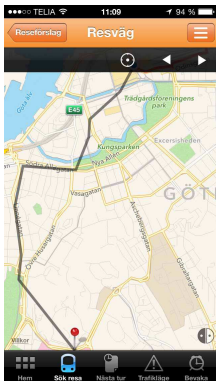
In terms of networks

- What question do we ask?
- Discuss with your neighbour!
- Suggestions?

Answer:



The shortest path problem: a useful application

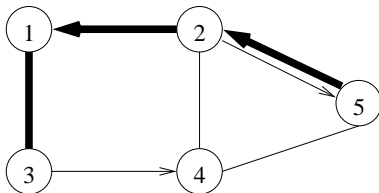


A number of “short” (or fast) paths that

- depart at the earliest “now”, and
- arrive at the latest “around 12:40”

Shortest path problem—properties & solution

- What properties of the problem can be utilized to construct an efficient solution method for the shortest path problem?



- Discuss
- ... draw on the board ...

Solving the problem ...

- How long is the shortest path from 1 to 6? Why?
- Discuss
- How can we find this path, using the “spatial” properties of the network?
- Discuss
- ... *adjust spatially the illustration on the board ...*

A mathematical model

Let y_i = length of the shortest path from node 1 to node i

- “Stretch the arcs” between the nodes 1 and 6 \Leftrightarrow maximize the difference of the “potentials” y_6 and y_1 :

$$(y_6 - y_1) \longrightarrow \max$$

- The arcs are not elastic:

$$\begin{array}{lll} y_2 - y_1 \leq 4 & y_4 - y_2 \leq 2 & y_5 - y_4 \leq 5 \\ y_3 - y_1 \leq 2 & y_4 - y_3 \leq 3 & y_6 - y_4 \leq 3 \\ y_2 - y_3 \leq 3 & y_5 - y_2 \leq 5 & y_6 - y_5 \leq 1 \end{array}$$

- A system of nine inequalities (*not* equations) and six unknowns,
- An objective function to be maximized
- All involved functions are differences between potentials

Another mathematical model—based on flows

Send one unit of flow along the shortest path from node 1 to node 6

- Let $x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is in the shortest path from 1 to 6} \\ 0 & \text{otherwise} \end{cases}$
- Objective:
 $(4x_{12} + 2x_{13} + 3x_{32} + 2x_{24} + 3x_{34} + 5x_{25} + 5x_{45} + 3x_{46} + 1x_{56}) \rightarrow \min$
- Node balance (any flow that enters a node must also leave it)

$$\begin{array}{rcccccccc} -x_{12} & -x_{13} & & & & & & & = & -1 \\ +x_{12} & & +x_{32} & -x_{24} & & -x_{25} & & & = & 0 \\ & +x_{13} & -x_{32} & & -x_{34} & & & & = & 0 \\ & & & +x_{24} & +x_{34} & & -x_{45} & -x_{46} & = & 0 \\ & & & & & +x_{25} & +x_{45} & & -x_{56} & = & 0 \\ & & & & & & & +x_{46} & +x_{56} & = & 1 \\ x_{12} & , & x_{13} & , & x_{32} & , & x_{24} & , & x_{34} & , & x_{25} & , & x_{45} & , & x_{46} & , & x_{56} & \geq & 0 \end{array}$$

The mathematical models are each other's LP duals

The optimal solutions to the two models

- $y_1^* = 0, y_2^* = 4, y_3^* = 2, y_4^* = 5, y_5^* = 9, y_6^* = 8$
- \Leftrightarrow maximal difference of the potentials:

$$(y_6^* - y_1^*) = 8$$

- Fulfilment of the constraints:

$$\begin{array}{lll} y_2^* - y_1^* = 4 = 4 & y_4^* - y_2^* = 1 < 2 & y_5^* - y_4^* = 4 < 5 \\ y_3^* - y_1^* = 2 = 2 & y_4^* - y_3^* = 3 = 3 & y_6^* - y_4^* = 3 = 3 \\ y_2^* - y_3^* = 2 < 3 & y_5^* - y_2^* = 5 = 5 & y_6^* - y_5^* = -1 < 1 \end{array}$$

- The optimal solution to the flow model:

$$x_{13}^* = x_{34}^* = x_{46}^* = 1$$

$$x_{12}^* = x_{32}^* = x_{24}^* = x_{25}^* = x_{45}^* = x_{56}^* = 0$$

[Illustrate the complementarity]

A general LP formulation: shortest path from node $s \in N$ to node $t \in N$ in a directed graph $G = (N, A, \mathbf{d})$

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

Linear programming formulation:

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

Linear programming dual:

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

- Given: a network/graph of nodes N , (directed) arcs A , and arc lengths d_{ij} , $(i, j) \in A$
- Denoted $G = (N, A, \mathbf{d})$
- Find the shortest path from a source node ($s \in N$) to a destination node ($t \in N$)
- Important concept: For $i_\ell \in N$, $\ell \in \{1, \dots, k\}$, a *negative cycle* $\{i_1, i_2, \dots, i_k\}$ is such that

$$\sum_{\ell=1}^{k-1} d_{i_\ell i_{\ell+1}} + d_{i_k i_1} < 0$$

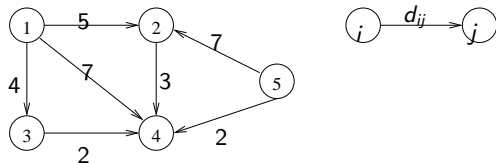
What impact does this have on a shortest path?

Principle of optimality formulated by Bellman's equations (Ch. 8.4.1)

- 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 to node k
- Let y_j denote the length of the shortest path from node s to j

Bellman's equations:

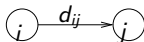
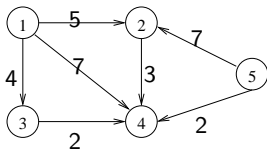
- $y_s = 0$
- $y_j = \min_i \{y_i + d_{ij} : (i,j) \in A\}$ for all $j \neq s$



Solution method I: Bellman's equations

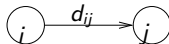
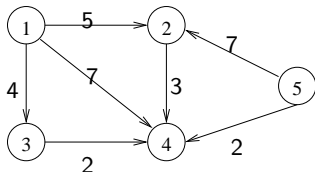
A special case of dynamic programming

- If the graph is directed with no cycles: solve Bellman's equations in topological order
- Shortest path from node 1 to each of the other nodes (1,5,2,3,4):
 - $y_1 := 0$
 - $y_5 := \min\{\infty\} = \infty$
 - $y_2 := \min\{\infty; y_1 + d_{12}; y_5 + d_{52}\} = \min\{\infty; 0 + 5; \infty\} = 5$
 - $y_3 := \min\{\infty; y_1 + d_{13}\} = \min\{\infty; 0 + 4\} = 4$
 - $y_4 := \min\{\infty; y_1 + d_{14}; y_2 + d_{24}; y_3 + d_{34}; y_5 + d_{54}\} = \min\{\infty; 0 + 7; 5 + 3; 4 + 2; \infty + 2\} = 6$



Solution method II: Dijkstra's algorithm

- The graph may contain cycles but all arc costs must be non-negative (i.e., $d_{ij} \geq 0$)



- 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 arc lengths are non-negative (cycles may exist)
- N = set of all nodes; source node $s \in N$
- d_{ij} = length of arc from i to j for all $i, j \in N$
- $d_{ij} := \infty$ if no direct arc from i to j

Dijkstra's shortest path algorithm

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

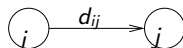
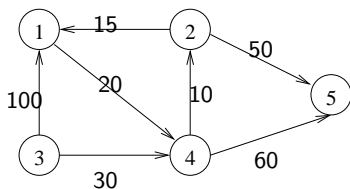
Step 1:

- (a) If $\bar{S} = \emptyset$, stop. Otherwise, find node $j \in \bar{S}$ such that $y_j = \min_{i \in \bar{S}} \{y_i\}$. Set $S := S \cup \{j\}$ 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 $pred(k) := i$. Repeat from (a).

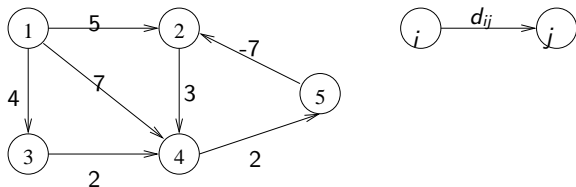
- The vector $pred$ keeps track of the predecessors
- Dijkstra's algorithm actually finds shortest paths from the source to *all* others nodes (this is not formulated in the LP)

Example: Dijkstra's algorithm

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



Negative arc lengths and negative cycles



- Negative length of arcs: extend Dijkstra's algorithm according to "move nodes back from S to \bar{S} " (known as Ford's algorithm)
 - There may be a cycle of *negative* total length
- ⇒ "Length" of the shortest path $\rightarrow -\infty$
- ⇒ 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 lengths are allowed; negative cycles will be detected
- Idea: Three nodes i, k, j and lengths d_{ik} , d_{kj} , and d_{ij}
- $i \rightarrow k \rightarrow j$ is a short-cut if $d_{ik} + d_{kj} < d_{ij}$
- In each iteration $1 \dots k$, check whether d_{ij} can be improved by using the short-cut via k
- Administration of the algorithm: Maintain two matrices per iteration: $D[k]$ for the lengths and $pred[k]$ to keep track of the predecessor of each node

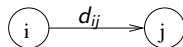
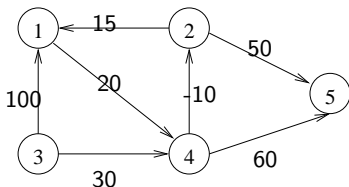
Floyd–Warshall's algorithm

Floyd–Warshall's algorithm

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

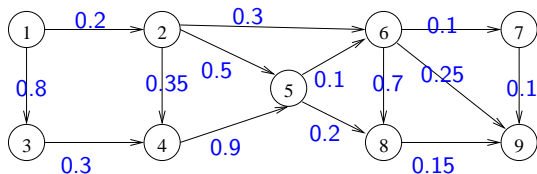
- Step k :**
- $D[k] := D[k - 1]$, $pred[k] := pred[k - 1]$
 - For each element d_{ij} in $D[k]$:
If $d_{ik} + d_{kj} < d_{ij}$, set $d_{ij} := d_{ik} + d_{kj}$ and $pred_{ij}[k] := k$
 - Set $k := k + 1$
 - If $k > n$ stop, else repeat Step k

Find the shortest path from node 3 to all other nodes



Example: Most reliable route

- Mr Q drives to work daily
- Every link in the road network is patrolled by the police
- A probability $p_{ij} \in [0, 1]$ of *not* being stopped by the police is assigned to 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 $Prob(\text{not being stopped})$



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

Alternative objectives \Rightarrow Variants of Bellman's equations (Ch. 8.4.4)

Most reliable path (failure probability $p_{ij} \in [0, 1]$ for arc (i, j)):

- $y_s = 1$
- For all $j \in N \setminus \{s\}$:
 - $y_j = \max_{i \in N} \{y_i \cdot p_{ij} : (i, j) \in A\}$

Cannot be formulated as a *linear* optimization problem

Highest capacity path (capacity $K_{ij} \geq 0$ on arc (i, j)):

- $y_s = \infty$
- For all $j \in N \setminus \{s\}$:
 - $y_j = \max_{i \in N} \{ \min\{y_i, K_{ij}\} : (i, j) \in A \}$

Can be formulated as a *linear* optimization problem. How?