Foundations of Operations Research

Master of Science in Computer Engineering

Roberto Cordone
roberto.cordone@unimi.it

Tuesday 13.15 - 15.15
Thursday 10.15 - 13.15

http://homes.di.unimi.it/~cordone/courses/2014-for/2014-for.html

A routing model

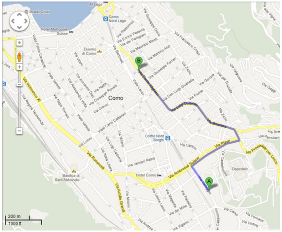
An ambulance must go from the site of an accident to the nearest hospital with an available emergency service. The street network and an estimated travel time on each street are known.

Determine the path from the accident site to the chosen hospital which requires the minimum total time.

The natural combinatorial model is an arc-weighted directed graph (N, A, c)

- N includes the crossings and hospitals, plus
  - the accident site s
  - a fictitious terminal node t
- A includes the lanes:
  - one arc for one-way streets
  - two opposite arcs for two-way streets

  plus one arc from each hospital node to t
- $c : A \to \mathbb{R}^+$ provides the travel time on a lane ($c = 0$ for the arcs entering t)

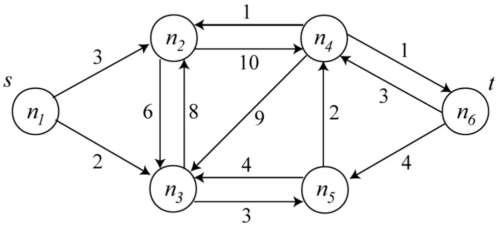


What kind of subgraph $P = (U, X)$ are we looking for?

A routing model

Let us denote by $P = (U, X)$ any feasible subgraph:

- it must be a directed path $X = \{(i_0, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k)\}$
- it must start in s and end in t: $i_0 = s$ and $i_k = t$
- its total cost must be minimum: $X = \arg\min \sum_{(i,j) \in X} c_{ij}$



Minimum path problem

Given

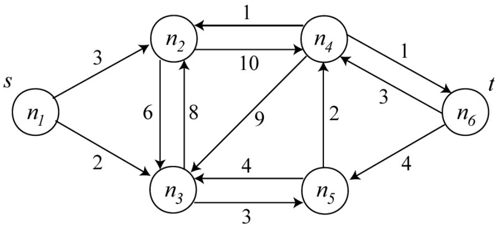
- an directed strongly connected graph $G = (N, A)$ with $n = |N|$ nodes and $m = |A|$ arcs
- a cost function $c : A \to \mathbb{R}$

find a subgraph $P^* = (U^*, X^*)$ such that

1. X^* is a directed path
2. X^* goes from s to t
3. the total cost of X^* is minimum:

$$c_{X^*} \leq c_X \text{ for all } P = (U, X) \text{ enjoying properties 1 and 2}$$

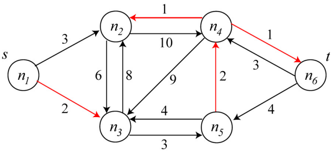
where $c_X = \sum_{(i,j) \in X} c_{ij}$



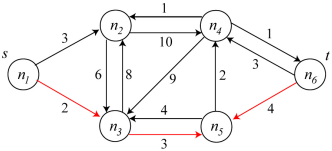
Examples

P is not an optimal solution because it is...

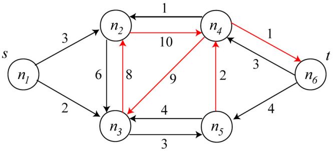
...not a path



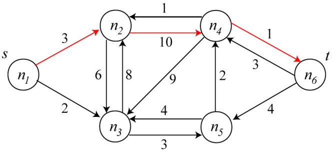
...not a directed path



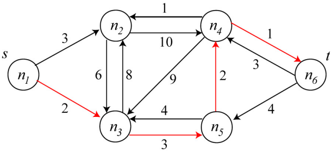
...not going from s to t



...nonminimal ($c_X = 14$)



P^* is optimal ($c_{X^*} = 8$)



A second model: secure message transmission

Transmit a secret message to an agent through the connections provided by a network of agents. The message can be intercepted at any connection. Minimize the probability of interception.

We model the network as a weighted directed graph $G = (N, A)$:

- a node for each agent: s is the transmitter, t the receiver
- two opposite arcs for each available connection
- a probability of interception $p_{ij} : A \to [0; 1)$

The probability of interception using a subset X of the arcs is

$$f(X) = 1 - \prod_{(i,j) \in X} (1 - p_{ij})$$

i. e. the complement of the probability not to be intercepted at any link

$$\min_X f(X) \Leftrightarrow \max_X \log \prod_{(i,j) \in X} (1 - p_{ij}) \Leftrightarrow \min_X \sum_{(i,j) \in X} \log \frac{1}{(1 - p_{ij})}$$

We are looking for a minimum cost path $P = (U, X)$ from s to t

A third model: optimal machine replacement plan

A factory spends 12 000 *euros to buy a machine. The following table reports its estimated yearly maintenance cost, which increases with age, and its market value, which decreases with age.*

machine age (years)	maintenance cost (Keuros/year)	residual value (Keuros)
0	2	12
1	4	7
2	5	6
3	9	2
4	12	1

The machine can be sold and replaced by a new more efficient one at the beginning of each year. The production must never stop.

Minimize the total cost along a 5*-year term.*

Example: replace the machine at the beginning of the 3rd year
$c = 12 + 2 + 4 - 6 + 12 + 2 + 4 + 5 - 2 = 33$

Example: replace the machine at the beginning of the 2nd and 5th year

$c = 12 + 2 - 7 + 12 + 2 + 4 + 5 - 2 + 12 + 2 - 7 = 35$

A third model: optimal machine replacement plan

The model is a directed weighted graph (N, A, c):

- N models the beginning of each year and the end of the term
- A models the potential lifespans of a machine
- c models the net cost of a machine during its lifespan (purchase and maintenance cost minus selling price)

The cost of an arc only depends on the length of the associated lifespan

- one year: $c := 12 + 2 - 7 = 7$
- two years: $c := 12 + (2 + 4) - 6 = 12$
- three years: $c := 12 + (2 + 4 + 5) - 2 = 21$
- four years: $c := 12 + (2 + 4 + 5 + 9) - 1 = 31$
- five years: $c := 12 + (2 + 4 + 5 + 9 + 12) = 44$

A replacement plan during a given time interval is an uninterrupted sequence of machine lifespans: it corresponds to a path from the beginning to the end of the interval

The cost of the replacement plan is equal to the cost of the path

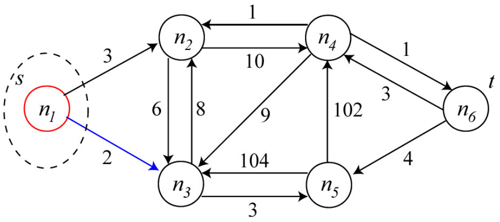
We are looking for a minimum cost path from n_1 to n_6

The greedy algorithm does not work (1)

Start from s; add the minimum cost arc going out of the last added node

It is an adaptation of Prim's algorithm for optimal spanning trees:

- initialize U with the origin s (not chosen *ad libitum*)
- consider only outgoing arcs (the graph is directed)
- consider only arcs going out of the last added node u_{last} (the solution is a path, not a tree)
- stop when the destination t is reached (the solution is not spanning)



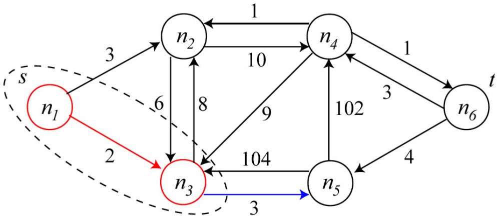
$U = \{n_1\}$, $X = \emptyset$

$u_{\text{last}} = n_1$, $\Delta^+_{u_{\text{last}}} = \{(n_1, n_2), (n_1, n_3)\}$

$a^* = \arg \min_{a \in \Delta^+_{u_{\text{last}}}} c_a = (n_1, n_3)$ $\Rightarrow X \cup \{a^*\} = \{(n_1, n_3)\}$

The greedy algorithm does not work (2)

Start from s; add the minimum cost arc going out of the last added node



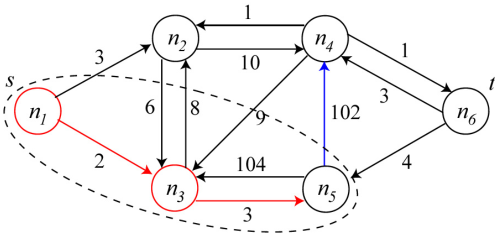
$U = \{n_1, n_3\}$, $X = \{(n_1, n_3)\}$

$u_{\text{last}} = n_3$, $\Delta^+_{u_{\text{last}}} = \{(n_3, n_2), (n_3, n_5)\}$

$a^* = \arg \min_{a \in \Delta^+_{u_{\text{last}}}} c_a = (n_3, n_5)$ $\Rightarrow X \cup \{a^*\} = \{(n_1, n_3), (n_3, n_5)\}$

The greedy algorithm does not work (3)

Start from s; add the minimum cost arc going out of the last added node



$U = \{n_1, n_3, n_5\}$, $X = \{(n_1, n_3), (n_3, n_5)\}$

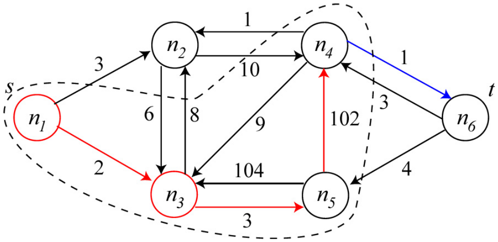
$u_{\text{last}} = n_5$, $\Delta^+_{u_{\text{last}}} = \{(n_5, n_3), (n_5, n_4)\}$

$a^* = \arg \min_{a \in \Delta^+_{u_{\text{last}}}} c_a = (n_5, n_4)$

$$\Rightarrow X \cup \{a^*\} = \{(n_1, n_3), (n_3, n_5), (n_5, n_4)\}$$

The greedy algorithm does not work (4a)

Start from s; add the minimum cost arc going out of the last added node



$U = \{n_1, n_3, n_4, n_5\}$, $X = \{(n_1, n_3), (n_3, n_5), (n_5, n_4)\}$

$u_{\text{last}} = n_4$, $\Delta^+_{u_{\text{last}}} = \{(n_4, n_2), (n_4, n_3), (n_4, n_6)\}$

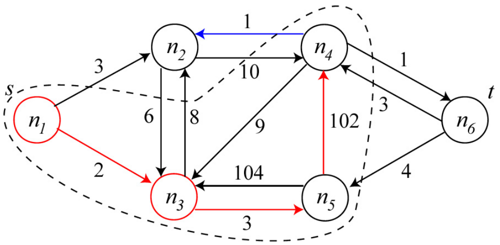
$a^* = \arg \min_{a \in \Delta^+_{u_{\text{last}}}} c_a = (n_4, n_6)$

$\Rightarrow X \cup \{a^*\} = \{(n_1, n_3), (n_3, n_5), (n_5, n_4), (n_4, n_6)\}$, not optimal!

This algorithm can provide solutions of any cost ($c_X = 108 \gg c_{X^*} = 14$)

The greedy algorithm does not work (4b)

Start from s; add the minimum cost arc going out of the last added node



$$a^* = \arg \min_{a \in \Delta^+_{u_{\text{last}}}} c_a = (n_4, n_2)$$

This algorithm can even find no solution at all, i. e. never reach t: either it stops in n_2 or it builds an infinite circuit $(n_2, n_3.n_5, n_4)$

The reason is that each choice strictly limits future choices, but it does not take into account the cost of future steps

In order to take into account the future steps, the solution is to maintain several candidate paths and update them when better solutions are found

Dijkstra's algorithm (1959)

Find by mathematical induction the shortest path from s to all nodes:

1. find the shortest path from s to the nearest node and set $k := 1$
2. find the shortest path from s to the $(k+1)$-th nearest node, knowing the shortest paths from s to its k nearest nodes
3. set $k := k + 1$
4. if $k < n$, go back to step 2; otherwise, terminate

Since each step of the algorithm marks the k nearest nodes, it can terminate as soon as t is marked

The algorithm requires a fundamental assumption

$$c_{ij} \geq 0 \text{ for all } (i,j) \in A$$

The base case

The base case is easy: if $c_{ij} \geq 0$ for all $(i, j) \in A$, the shortest path X_1^* from s to the nearest node t^* includes only the shortest arc going out of s

$$X_1^* = \left\{ \arg \min_{(s,j) \in \Delta_s^+} c_{sj} \right\}$$

Proof: *if X_1^* includes more than one arc, t^* is farther from s than the first node along X_1^*, because all arcs have nonnegative costs. Since this is a contradiction, X_1^* consists of a single arc (the shortest one).*

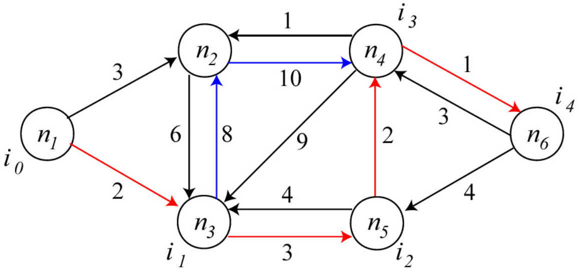
Mathematical induction requires to derive the shortest path to the $k+1$-th nearest node from the shortest paths to the k nearest nodes:

- the path to the second node from the path to the first
- the path to the third node from the paths to the first two
- ...
- the path to the farthest node from the paths to the other ones

Then, it is possible to find the shortest path to all nodes

A basic property

If $P^*_{i_0,i_k} = (i_0, \ldots, i_k)$ is a shortest path from i_0 to i_k, then for all $u, v \in \mathbb{N}$: $0 \leq u < v \leq k$ the subpath $P_{i_u,i_v} = (i_u, \ldots, i_v) \subseteq P^*_{i_0,i_k}$ is a shortest path from i_u to i_v



Proof: *By contradiction, assume that P_{i_u,i_v} is not optimal, and that path P'_{i_0,i_k} is cheaper. Replacing P_{i_u,i_v} with P'_{i_u,i_v} would reduce the total cost of $P^*_{i_0,i_k}$, thus contradicting its optimality.*

The shortest path problem enjoys the optimality principle: an optimal solution can be constructed efficiently combining optimal solutions of subproblems

This does not hold for all problems

A fundamental theorem

Denote as

1. P^*_{si} a shortest path from s to i and ℓ_i its cost
2. S_k the subset of the k nodes with the smallest ℓ_i $(1 \leq k \leq n-1)$

If P^*_{si} and ℓ_i are known for all $i \in S_k$, a shortest path from s to the $(k+1)$-th farthest node is $P^*_{si} \cup (i^*, j^*)$, where

$$(i^*, j^*) = \arg \min_{(i,j) \in \Delta^+_{S_k}} (\ell_i + c_{ij})$$

Proof: $P^*_{si^*} \cup (i^*, j^*)$ *is a path from* s *to* $N \setminus S_k$. *We prove that is it the shortest one. Consider any path* $P_{sz} = (s, \ldots, u, v, \ldots, z)$ *from* s *to a node* $z \in N \setminus S_k$, *and let* $(u, v) \in \Delta^+_{S_k}$ *be the first arc going out of* S_k, *i. e. subpath* $P_{su} = (s, \ldots, u) \subset P_{sz}$ *is fully in* S_k *(no assumption on* P_{vz}*).*

The cost of path P_{sz} *is* $c_{P_{sz}} = c_{P_{su}} + c_{uv} + c_{P_{vz}}$

All arcs have nonnegative cost: $c_{P_{vz}} \geq 0 \Rightarrow c_{P_{sz}} \geq c_{P_{su}} + c_{uv}$

P_{su} *is not necessarily optimal:* $c_{P_{su}} \geq \ell_u \Rightarrow c_{P_{sz}} \geq \ell_u + c_{uv}$

By definition, $\ell_u + c_{uv} \geq c_{P^*_{si^*} \cup (i^*, j^*)} \Rightarrow c_{P_{sz}} \geq c_{P^*_{si^*} \cup (i^*, j^*)}$

Therefore, $P^*_{si} \cup (i, j)$ *is the shortest path from* s *to any node in* $N \setminus S_k$, *i. e. the shortest path to the* $(k+1)$*-th nearest node of* s

Shortest path arborescence

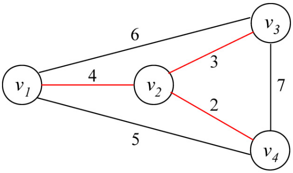
Dijkstra's algorithm provides the set of all shortest paths from s to the other $n-1$ nodes

- s is connected to all other nodes
- each node in $N \setminus \{s\}$ receives one arc
  (if alternative paths exist, the algorithm provides only one)

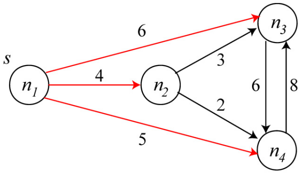
Their union is a spanning arborescence (directed tree) and s is its root

The solution has no relation with the minimum spanning tree

- the graph and the tree are directed, with a clearly defined origin
- the cost is minimum on each path, not overall



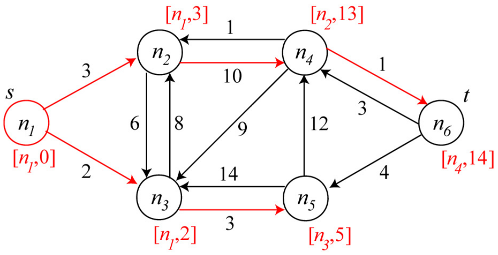
Minimum spanning tree



Shortest paths arborescence

Representing the solution

An s-rooted arborescence is represented compactly by the predecessor vector π, which provides for each node i the previous node along the unique path P_{si}



$$\begin{array}{cccccccc} & & 1 & 2 & 3 & 4 & 5 & 6 & \\ \pi = [& & 1 & 1 & 1 & 2 & 3 & 4 &] \end{array}$$

The paths can be retrieved following the backward chain of predecessors

Example: to retrieve $P_{n_1 n_4}$

- start from the last node n_4
- since $\pi_4 = 2$, go to n_2
- since $\pi_2 = 1$, go to n_1
- since $\pi_1 = 1$, stop

$\rightarrow P_{n_1 n_4} = (n_1, n_2, n_4)$

Dijkstra's algorithm

Dijkstra(N, A, c)

$S := \{s\};$ $\qquad\qquad\qquad\qquad \{\ k := 0\ \}$

$\ell_s := 0;\ \pi_s := s;$

While $S \subset N$

$\quad (i^*, j^*) := \arg \min_{(i,j) \in \Delta_S^+} (\ell_i + c_{ij});$

$\quad \ell_{j^*} := \ell_{i^*} + c_{i^*, j^*};$

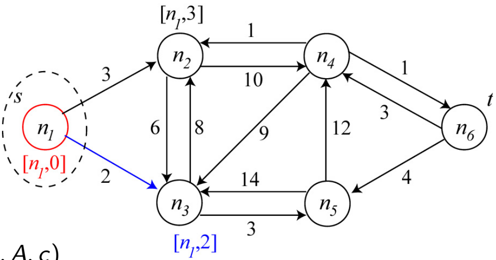
$\quad \pi_{j^*} := i^*;$

$\quad S := S \cup \{j^*\};$ $\qquad\qquad \{\ k := k + 1\ \}$

Return $\pi;$

Notice that

- the base case $(k = 1)$ is solved by the first iteration of the While loop
- the paths P^*_{si} are represented through the predecessor vector π
- if $\Delta_S^+ = \emptyset$ and $S \subset N$, the nodes in $N \setminus S$ cannot be reached from s



Dijkstra(N, A, c)

$S := \{s\}$;

$\ell_s := 0;\ \pi_s := s$;

While $S \subset N$

$\quad (i^*, j^*) := \arg \min_{(i,j) \in \Delta_S^+} (\ell_i + c_{ij})$;

$\quad \ell_{j^*} := \ell_{i^*} + c_{i^*, j^*}$;

$\quad \pi_{j^*} := i^*$;

$\quad S := S \cup \{j^*\}$;

Return π;

$X := \emptyset;\ S := \{n_1\}$;

$\Delta_S^+ := \{(n_1, n_2), (n_1, n_3)\}$;

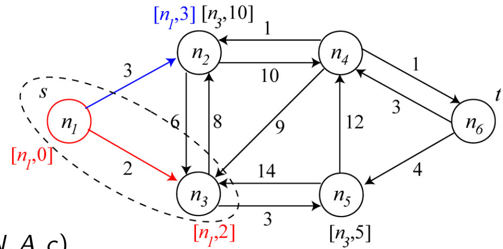
$(i^*, j^*) := (n_1, n_3)$;

$\ell_{j^*} := 0 + 2;\ (< 0 + 3)$

$\pi_{j^*} := n_1$;

$S := \{n_1, n_3\}$;

Application of Dijkstra's algorithm (2)



Dijkstra(N, A, c)

$S := \{s\};$

$\ell_s := 0;\ \pi_s := s;$

While $S \subset N$

$\quad (i^*, j^*) := \arg\min_{(i,j)\in\Delta_S^+} (\ell_i + c_{ij});$

$\quad \ell_{j^*} := \ell_{i^*} + c_{i^*,j^*};$

$\quad \pi_{j^*} := i^*;$

$\quad S := S \cup \{j^*\};$

Return $\pi;$

$\Delta_S^+ := \{(n_1, n_2), (n_3, n_2), (n_3, n_5)\};$

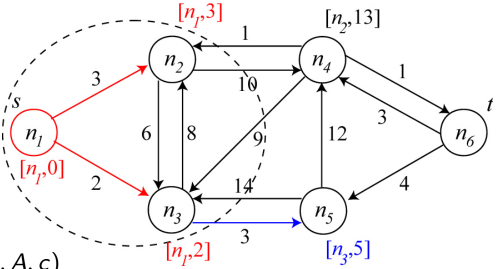
$(i^*, j^*) := (n_1, n_2);$

$\ell_{j^*} := 0 + 3;\ (< 2 + 3 \text{ and } < 2 + 8)$

$\pi_{j^*} := n_1;$

$S := \{n_1, n_2, n_3\};$

Application of Dijkstra's algorithm (3)



Dijkstra(N, A, c)

$S := \{s\};$

$\ell_s := 0;\ \pi_s := s;$

While $S \subset N$

$(i^*, j^*) := \arg\min_{(i,j)\in\Delta_S^+} (\ell_i + c_{ij});$

$\ell_{j^*} := \ell_{i^*} + c_{i^*,j^*};$

$\pi_{j^*} := i^*;$

$S := S \cup \{j^*\};$

Return $\pi;$

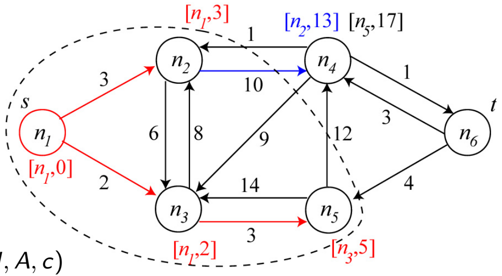
$\Delta_S^+ := \{(n_2, n_4), (n_3, n_5)\};$

$(i^*, j^*) := (n_3, n_5);$

$\ell_{j^*} := 2 + 3;\ (< 3 + 10)$

$\pi_{j^*} := n_3;$

$S := \{n_1, n_2, n_3, n_5\};$



Dijkstra(N, A, c)

$S := \{s\};$

$\ell_s := 0;\ \pi_s := s;$

While $S \subset N$

$(i^*, j^*) := \arg \min_{(i,j) \in \Delta_S^+} (\ell_i + c_{ij});$

$\ell_{j^*} := \ell_{i^*} + c_{i^*, j^*};$

$\pi_{j^*} := i^*;$

$S := S \cup \{j^*\};$

Return $\pi;$

$\Delta_S^+ := \{(n_2, n_4), (n_5, n_4)\};$

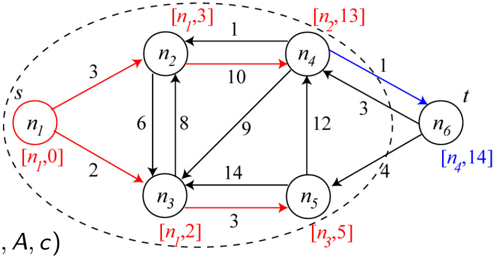
$(i^*, j^*) := (n_2, n_4);$

$\ell_{j^*} := 3 + 10;\ (< 5 + 12)$

$\pi_{j^*} := n_2;$

$S := \{n_1, n_2, n_3, n_4, n_5\};$

Application of Dijkstra's algorithm (5)



Dijkstra(N, A, c)

$S := \{s\};$

$\ell_s := 0; \; \pi_s := s;$

While $S \subset N$

$(i^*, j^*) := \arg \min_{(i,j)\in\Delta_S^+} (\ell_i + c_{ij});$

$\ell_{j^*} := \ell_{i^*} + c_{i^*,j^*};$

$\pi_{j^*} := i^*;$

$S := S \cup \{j^*\};$

Return $\pi;$

$\Delta_S^+ := \{(n_4, n_6)\};$

$(i^*, j^*) := (n_4, n_6);$

$\ell_{j^*} := 13 + 1;$

$\pi_{j^*} := n_4;$

$S := N;$ (termination condition)

Much similar to Prim's algorithm, but minimizing $\ell_i + c_{ij}$

Complexity of Dijkstra's algorithm

Dijkstra's algorithm consists of an initial step of complexity T_{in} and a certain number $i_{\max}$ of iterations of complexity $T_{\text{iter}}^{(i)}$

$$T = T_{\text{in}} + \sum_{i=1}^{i_{\max}} T_{\text{iter}}^{(i)}$$

Dijkstra(N, A, c)

$S := \{s\};$

$\ell_s := 0;\ \pi_s := s;$

While $S \subset N$

$\quad (i^*, j^*) := \arg \min_{(i,j)\in\Delta_S^+} (\ell_i + c_{ij});$

$\quad \ell_{j^*} := \ell_{i^*} + c_{i^*,j^*};$

$\quad \pi_{j^*} := i^*;$

$\quad S := S \cup \{j^*\};$

Return π;

$T_{\text{in}} \in O(1)$

$i_{\max} = n - 1$ (*one vertex at a time*)

$T_{\text{iter}}^{(i)} \in O(\alpha)$

Overall $T \in O(\alpha n)$

Minimum cost arc identification (1)

Possible implementations

1. Scan all the arcs and verify which ones belong to Δ_S^+: $T \in O(m)$
2. Maintain for each $j \in N \setminus S$ the cheapest path ending in $\Delta_S^+ \cap \Delta_{\{j\}}^-$

$$\tilde{\pi}_j = \arg \min_{i \in S : (i,j) \in \Delta_S^+ \cap \Delta_{\{j\}}^-} \ell_i + c_{ij}$$

- build $\tilde{\pi}_j$: $O(n)$
- find the minimum $\tilde{\pi}_j$: $O(n)$
- update $\tilde{\pi}_j$: $O(n)$

Dijkstra(N, A, c)

$S := \{s\}$;

$\ell_s := 0$; $\pi_s := s$;

While $S \subset N$

$(i^*, j^*) := \arg \min_{(i,j) \in \Delta_S^+} (\ell_i + c_{ij})$;
$\ell_{j^*} := \ell_{i^*} + c_{i^*,j^*}$;
$\pi_{j^*} := i^*$;
$S := S \cup \{j^*\}$;

Return π;

$\rightarrow$ the complexity of the first implementation is $T \in O(mn)$

Minimum cost arc identification (2)

Possible implementations

1. Scan all the arcs and verify which ones belong to Δ_S^+: $T \in O(m)$
2. Maintain for each $j \in N \setminus S$ the cheapest path ending in $\Delta_S^+ \cap \Delta_{\{j\}}^-$

$$\tilde{\pi}_j = \arg \min_{i \in S : (i,j) \in \Delta_S^+ \cap \Delta_{\{j\}}^-} \ell_i + c_{ij}$$

- build $\tilde{\pi}_j$: $O(n)$
- find the minimum $\tilde{\pi}_j$: $O(n)$
- update $\tilde{\pi}_j$: $O(n)$

Dijkstra(N, A, c)

$S := \{s\}$;

$\tilde{\ell}_s := 0$; $\pi_s := s$;

For each $i \in N \setminus \{s\}$ do

If $(s, i) \in A$
then $\tilde{\ell}_i := c_{si}$ else $\tilde{\ell}_i := +\infty$;
$\tilde{\pi}_i := s$;

While $S \subset N$

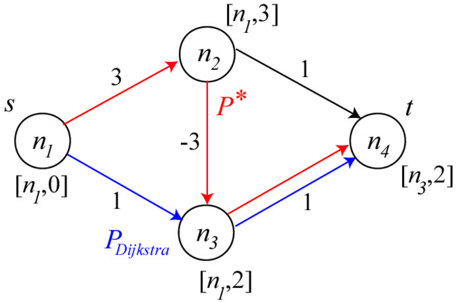
$j^* := \arg \min_{j \in N \setminus S} \tilde{\ell}_i$;
$S := S \cup \{j^*\}$;
For each $(j^*, h) \in \Delta_S^+$ do
If $\tilde{\ell}_{j^*} + c_{j^*h} < \tilde{\ell}_h$
then $\tilde{\ell}_h := \tilde{\ell}_{j^*} + c_{j^*h}$;
$\tilde{\pi}_h := j^*$;

Return π;

$\rightarrow$ the complexity of the second implementation is $T \in O(n^2)$

Negative costs

Dijkstra's algorithm can fail when there are arcs with negative cost



The first step of the algorithm sets $\ell_3 = 1$ and $\pi_3 = 1$

But the single arc (n_1, n_3) is not the shortest path from s to n_3:
$P^* = (n_1, n_2, n_3)$ achieves $\ell_3 = 0$ exploiting the negative cost arc (n_2, n_3)

Consequently, the shortest path computed from s to t is incorrect