Shortest-path problem and Dijkstra's algorithm

G. Ferrari Trecate

Dipartimento di Ingegneria Industriale e dell'Informazione
Università degli Studi di Pavia

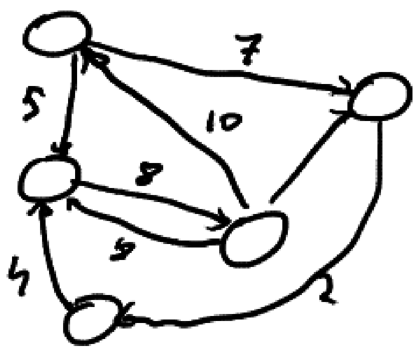
Industrial Automation

Shortest-path problems

Problem 5.1: Let $G=(V,E,c)$ be a directed network. Given $v_1, v_2 \in V$, find a $\underbrace{\text{simple path with minimal cost}}_{\text{"shortest path"}}$ from v_1 to v_2

Example

Vertices = cities
Edges = transportation links
Weights = distances
↳ shortest-path problem
〃
optimal path finding (e.g. Google maps)

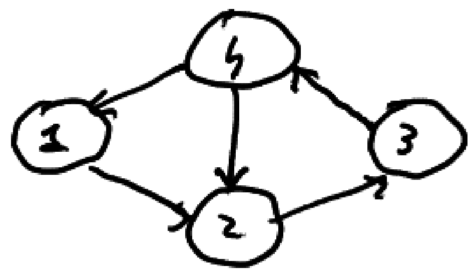


Computational complexity

Thm. Problem (S1) is NP-hard

Proof. Simple path $p \rightarrow$ at most $n = |V|$ edges in p.
Simple path with n edges = Hamiltonian cycle

Consider the instance of (S1) given by
$c(e) = -1, \ \forall e \in E$ and $v_1 = v_2$.



Solutions, if any, are Hamiltonian cycles

Therefore (Hamiltonian cycle problem) $\propto$ (S1) that implies
(S1) is NP-hard.

Rmk. • (S1) is not NP-complete because it is the optimization form of the Hamiltonian cycle problem

• the proof uses negative weights

Solution to (S1) through binary LP

Assumption: $v_1 \neq v_2$

Variables: $x_{ij} = \begin{cases} 1 & \text{if the edge } (i,j) \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases}$

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

$$\sum_{(i,j)\in \delta^+(\{i\})} x_{ij} - \sum_{(k,i)\in \delta^-(\{i\})} x_{ki} = \begin{cases} +1 & \text{if } i = v_1 \\ -1 & \text{if } i = v_2 \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \dots, n \qquad \text{(V1)}$$

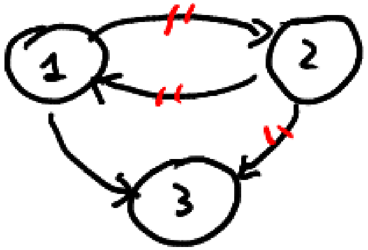
$$\sum_{(i,j)\in E(U)} x_{ij} \leq |U| - 1, \quad \forall U \subseteq V \qquad \text{(V2)}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V \qquad \text{(V3)}$$

Constraint (V1):

for all nodes in the path, there is the same number of incoming and outgoing edges, except for v_1 and v_2

↳ Otherwise variables $x_{ij}=1$ do not describe a path



$v_1 = 1$

$v_2 = 3$

Selected edges do not define a path from 1 to 3 because 2 does not verify (V1)

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

$$\sum_{(i,j)\in\delta^+(\{i\})} x_{ij} - \sum_{(k,i)\in\delta^-(\{i\})} x_{ki} = \begin{cases} +1 & \text{if } i = v_1 \\ -1 & \text{if } i = v_2 \\ 0 & \text{otherwise} \end{cases} \quad i = 1,\dots,n \qquad (V1)$$

$$\sum_{(i,j)\in E(U)} x_{ij} \le |U|-1, \quad \forall U \subseteq V \qquad (V2)$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V \qquad (V3)$$

Constraint (V2):

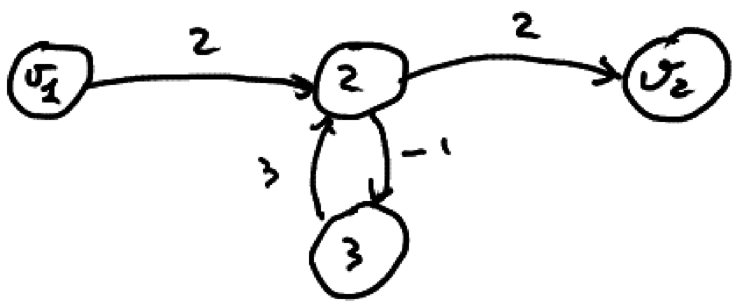
subtour elimination (necessary for obtaining simple paths)

Pbl: n° of (V2) constraints grows as $O(e^n)$ if G is complete

↳ the formulation is computationally inefficient!

Rmk. If G does not contain cycles with negative cost, then (V2) are redundant because optimal solutions will never contain cycles

Ex.



$v_1 \, 2 \, v_2 \to$ cost 4

$v_1 \, 2 \, 3 \, 2 \, v_2 \to$ cost 6

However, the complexity of binary LP is still prohibitive, even without (V2)

↳ there are polynomial LP formulations but even dedicated and more efficient algorithms!

Next: •) Dijkstra's algorithm when weights are ≥ 0

•) Floyd-Warshall algorithm when there are no cycles with negative cost

Dijkstra's algorithm

Problem (S2). Given the directed network $G=(V,E,c)$ with $c(e)\geq 0, \forall e\in E$, compute shortest paths from v_1 to every other vertex

Algorithm split in four steps

1) Init. Set the vertex labels

$$\ell(v)=\begin{cases} 0 & \text{if } v=v_1 \\ +\infty & \text{otherwise}\end{cases}$$

When the algorithm stops, $\ell(v)<+\infty$ means there is a path of cost $\ell(v)$ from v_1 to v

Define $L\subset V$ as the set of permanent vertices (i.e their labels cannot be further changed) and set $L=\emptyset$

Define pred(v) as the predecessor of v in a path and set pred$(v)=\phi$, $\forall v \in V$
When the algorithm stops, pred$(v)=q$ means that the edge (q,v) is in the computed shortest path from v_1 to v (pred$(v)=\phi$ means there is no path from v_1 to v)

2) Choice of the permanent vertex.

Pick $\bar{v} \in V \setminus L$ that verifies

$$\ell(\bar{v}) = \min_{v \in V \setminus L} \ell(v) \longrightarrow \text{minimal label}$$

and add $\bar{v}$ to L $\longrightarrow$ $\bar{v}$ becomes permanent

3) Update of $\ell(v)$ and pred (v). Let

$$R(\bar{v}) = \{ v \in V \setminus L : (\bar{v}, v) \in E \} \longrightarrow \text{non permanent successors of } \bar{v}$$

For all $v \in R(\bar{v})$, if $\ell(\bar{v}) + c(\bar{v}, v) < \ell(v)$, then

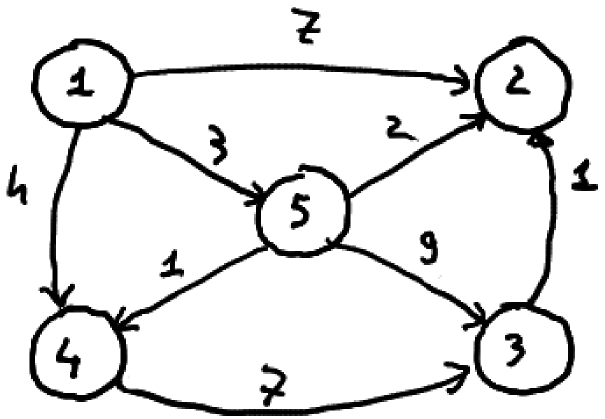
$$\cdot) \quad \ell(v) = \ell(\bar{v}) + c(\bar{v}, v)$$

$$\cdot\cdot) \quad \text{pred}(v) = \bar{v}$$

4) Stop the algorithm if $L = V$. Otherwise go to step 2

↳ the algorithm ends in n iterations

Example

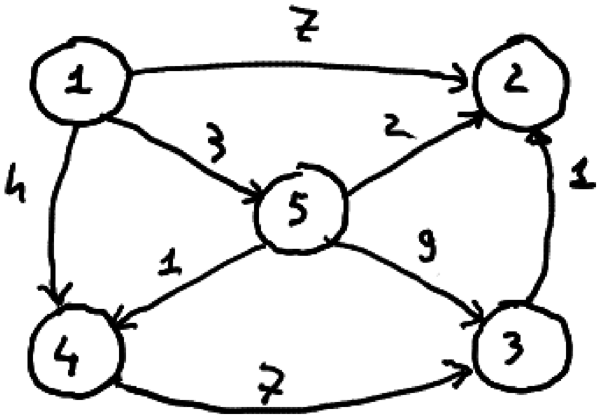


Compute all shortest paths from 1 and a shortest path from 1 to 3

Set $v_1 = 1$

$\bar{v}$	L	$\ell(1)$	$\ell(2)$	$\ell(3)$	$\ell(4)$	$\ell(5)$
/	$\emptyset$	0	∞	∞	∞	∞
1	$\{1\}$	$\underline{0}$	$7_{(1)}$	∞	$4_{(1)}$	$3_{(1)}$
5	$\{1,5\}$	$\underline{0}$	$5_{(5)}$	$12_{(5)}$	4	$\underline{3}$
4	$\{1,5,4\}$	$\underline{0}$	5	$11_{(4)}$	$\underline{4}$	$\underline{3}$
2	$\{1,5,4,2\}$	$\underline{0}$	$\underline{5}$	11	$\underline{4}$	$\underline{3}$
3	$L = V$	$\underline{0}$	$\underline{5}$	$\underline{11}$	$\underline{4}$	$\underline{3}$

← INIT



red = updates

(v) = updated predecessor

↓

no update until the end means

pred (v) = $\emptyset$

Reading the results

$\bar{v}$	L	$\ell(1)$	$\ell(2)$	$\ell(3)$	$\ell(4)$	$\ell(5)$
/	ϕ	0	∞	∞	∞	∞
1	$\{1\}$	$\underline{0}$	$7_{(1)}$	∞	$4_{(1)}$	$3_{(1)}$
5	$\{1,5\}$	$\underline{0}$	$5_{(5)}$	$12_{(5)}$	4	$\underline{3}$
4	$\{1,5,4\}$	$\underline{0}$	5	$11_{(4)}$	$\underline{4}$	$\underline{3}$
2	$\{1,5,4,2\}$	$\underline{0}$	$\underline{5}$	11	$\underline{4}$	$\underline{3}$
3	$L = V$	$\underline{0}$	$\underline{5}$	$\underline{11}$	$\underline{4}$	$\underline{3}$

Final predecessors (pick the last update)

pred (1) = ϕ

pred (2) = 5

pred (3) = 4

pred (4) = pred (5) = 1

Shortest path from 1 to 3

Start from 3 and move backwards

pred (3) = 4

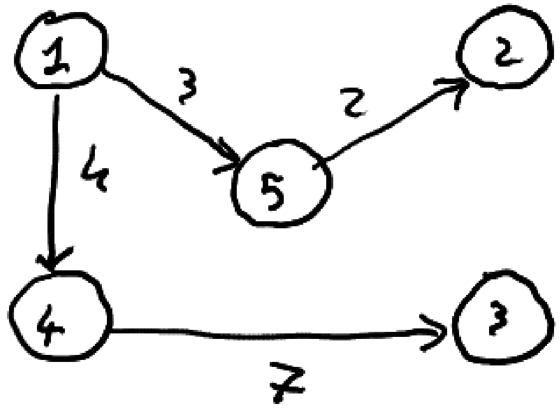
pred (4) = 1 = v_1 → STOP

↳ the path is 1 4 3

Reading the results

v	L	$\ell(1)$	$\ell(2)$	$\ell(3)$	$\ell(4)$	$\ell(5)$
/	$\emptyset$	0	∞	∞	∞	∞
1	$\{1\}$	$\underline{0}$	$7_{(1)}$	∞	$4_{(1)}$	$3_{(1)}$
5	$\{1,5\}$	$\underline{0}$	$5_{(5)}$	$12_{(5)}$	4	$\underline{3}$
4	$\{1,5,4\}$	$\underline{0}$	5	$11_{(4)}$	$\underline{4}$	$\underline{3}$
2	$\{1,5,4,2\}$	$\underline{0}$	$\underline{5}$	11	$\underline{4}$	$\underline{3}$
3	$L=V$	$\underline{0}$	$\underline{5}$	$\underline{11}$	$\underline{4}$	$\underline{3}$

Graph of shortest paths

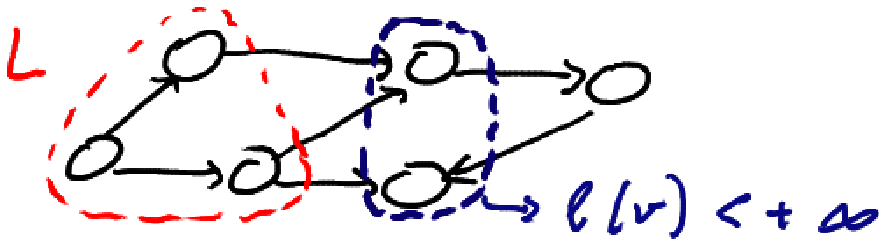


Remarks

1) At the end of each iteration

$$v \notin L \text{ and } \ell(v) < +\infty \iff \exists\, i : (i,v) \in \delta^+(L)$$

i.e. v is the successor of some permanent node



Therefore, in step 2 $\bar{v}$ is the successor of some permanent node

2) $v_1 \in L$ in every iteration after the first one

3) There is an implementation of the algorithm with complexity $O(n^2)$, $n = |V|$

↳ Intuition: . the "boundary" of L advances at each iteration ↳ at most n iterations

. every iteration has $O(n)$ complexity

Correctness of Dijkstra's algorithm

Thm. If $c(i,j) \geq 0$, $\forall (i,j) \in E$, at every iteration the label $\ell(\bar{v})$ of the new permanent node $\bar{v}$ is the cost of the shortest path from v_s to $\bar{v}$

Proof by induction....