

# Shortest path algorithms

Pramesh Kumar

IIT Delhi

August 7, 2025

# Shortest Path

- ▶ Fundamental problem with numerous applications.
- ▶ Appears as a subproblem in many network flow algorithms.
- ▶ Easy to solve.

## Shortest path problem

**Definition (Path cost).** The cost of a directed path  $\pi = (i_1, i_2, \dots, i_k)$  is the sum of cost of its individual links, i.e.,  $c^\pi = \sum_{i=1}^{k-1} t_{i,i+1}$ .

**Definition (Shortest Path Problem).** Given  $G(N, A)$ , link costs  $t : A \mapsto \mathbb{R}$ , and origin  $r \in N$ , the **shortest path problem** (also known as single-source shortest path problem) is to determine for every non-source node  $i \in N \setminus \{r\}$  a shortest cost directed path from node  $r$ .

OR

**Definition (Shortest Path Problem).** Given  $G(N, A)$ , link costs  $t : A \mapsto \mathbb{R}$ , and source  $r \in N$ , the **shortest path problem** is to determine how to send 1 unit of flow as cheaply as possible from  $r$  to each node  $i \in N \setminus \{s\}$  in an uncapacitated network.

## Types of shortest path (SP) problems

1. *Single-source shortest path*: SP from one node to all other nodes (if exists)
  - 1.1 with non-negative link costs.
  - 1.2 with arbitrary link costs.
2. *Single-pair shortest path* SP from between one node and another node.
3. *All-pairs shortest path* SP from every node to every node.
4. *Various generalizations of shortest path*:
  - Max capacity path problem
  - Max reliability path problem
  - SP with turn penalties
  - Resource-constraint SP problem
  - and many more

## Lemma (Subpaths of shortest path are shortest paths)

Let  $\pi = (r = i_1, \dots, i_h = k)$  be a shortest path from  $r$  to  $k$  and for  $1 \leq p \leq q \leq h$ , let  $\pi_{pq} = (i_p, \dots, i_q)$  be a subpath of  $\pi$  from  $p$  to  $q$ . Then,  $\pi_{pq}$  is a shortest path from  $i_p$  to  $i_q$ .

### Proof.

Decomposing path  $\pi$  into subpaths  $\pi_{rp}$ ,  $\pi_{pq}$ , and  $\pi_{qk}$ , so that

$c^\pi = c^{\pi_{sp}} + c^{\pi_{pq}} + c^{\pi_{qk}}$ . Assume that  $\pi'_{pq}$  be a path such that  $c^{\pi_{pq}} > c^{\pi'_{pq}}$ .

Then,  $\pi' = \pi_{sp} + \pi'_{pq} + \pi_{qk}$  has cost  $c^{(\pi')} = c^{\pi_{sp}} + c^{\pi'_{pq}} + c^{\pi_{qk}} < c^\pi$ , which contradicts that  $\pi$  is a shortest path from  $r$  to  $k$ . □

## LP formulation for a single pair shortest path

$$x_{ij} = \begin{cases} 1 & \text{if } (i, j) \in A \text{ is on shortest path} \\ 0 & \text{otherwise} \end{cases}$$

$$\min_{\mathbf{x}} \sum_{(i,j) \in A} t_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j \in FS(i)} x_{ij} - \sum_{j \in BS(i)} x_{ji} = \begin{cases} 1 & \text{if } i = r \\ -1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases}, \forall i \in N$$

$$1 \geq x_{ij} \geq 0, \forall (i, j) \in A$$

**Remark.** We can replace  $x_{ij} \in \{0, 1\}$  with  $1 \geq x_{ij} \geq 0$  due to a property whose discussion we are skipping here.

Let's write its KKT conditions ...

## Optimality conditions

### Theorem

For every node  $j \in N$ , let  $l(j)$  denote the cost of some directed path from source  $r$  to  $j$ . Then,  $l(j)$  represent the shortest path costs if and only if they satisfy the following optimality conditions:

$$l(j) \leq l(i) + t_{ij}, \forall (i, j) \in A \quad (\star)$$

### Proof.

$\implies$  Let  $l(j)$  represent the SP cost labels for  $j \in N$ . Assume that they do not satisfy the  $(\star)$ . Then, some link  $(i, j) \in A$  must satisfy  $l(i) > l(j) + t_{ij}$ . In this case, we can improve the cost of SP to node  $j$  by coming through node  $i$ , thereby contradicting the fact that  $l(j)$  represents the SP label of node  $j$ .



## Proof (contd.)

$\Leftarrow$  Consider labels  $l(j)$  satisfying  $(\star)$ . Let  $(r = i_1, i_2, \dots, i_k = j)$  be any directed path  $\pi$  from source  $r$  to node  $j$ . The conditions  $(\star)$  imply that

$$\begin{aligned}l(j) = l(i_k) &\leq l(i_{k-1}) + t_{i_{k-1}i_k} \\l(i_{k-1}) &\leq l(i_{k-2}) + t_{i_{k-2}i_{k-1}} \\&\vdots \\l(i_2) &\leq l(i_1) + t_{i_1i_2} = t_{i_1i_2}\end{aligned}$$

Adding above inequations, we get

$l(j) = l(i_k) \leq t_{i_{k-1}i_k} + t_{i_{k-2}i_{k-1}} + \dots + t_{i_1i_2} = \sum_{(i,j) \in \pi} t_{ij}$ . Thus  $t_j$  is a LB on the cost of any directed path from  $r$  to  $j$ . Since  $l(j)$  is the cost of some directed path from  $r$  to  $j$ , it is also an UB on the SP cost.

Therefore,  $l(j)$  is the shortest path cost from  $r$  to  $j$ . □



## Single-source shortest path

## Assumptions

1. Network is directed
2. Link costs are integers
3. There exists a directed path from  $r$  to every other node (can be satisfied by creating an artificial link from  $s$  to other nodes)
4. The network does not contain a negative cycle.

**Remark.** For a network containing a negative cycle reachable from  $r$ , the above LP will be unbounded since we can send an infinite amount of flow along that cycle.

### Can SP contain a cycle?

1. It cannot contain negative cycles.
2. It cannot contain positive cycles since removing the cycle produces a path with lower cost.
3. One can also remove zero weight cycle without affecting the cost of SP.

## Label setting and label correcting algorithms

- ▶ Shortest path algorithms assign tentative distance label to each node that represents an upper bound on the cost of shortest path to that node.
- ▶ Depending on how they update these labels, the algorithms can be classified into two types:
  1. Label setting
  2. Label correcting
- ▶ Label setting algorithms make one label permanent in each iteration
- ▶ Label correcting algorithms keep all labels temporary until the termination of the algorithm.
- ▶ Label setting algorithms are more efficient but label correcting algorithms can be applied to more general class of problems.

# Dijkstra's algorithm

## A label setting algorithm

- 1: **Input:** Graph  $G(N, A)$ , link costs  $\mathbf{t}$ , and source  $r$
- 2: **Output:** Optimal cost labels  $l$  and predecessors  $pred$
- 3: **procedure** DIJKSTRA( $G, \mathbf{t}, r$ )
- 4:      $SE = \{r\}$  ▷ Scan Eligible List
- 5:      $l(i) \leftarrow \infty, \forall i \in N \setminus \{r\}; l(r) \leftarrow 0$
- 6:      $pred(i) \leftarrow \text{NA}, \forall i \in N \setminus \{r\}; pred(r) \leftarrow 0$
- 7:     **while**  $SE \neq \emptyset$  **do**
- 8:         Choose a node  $i$  with minimum  $l(i)$  from  $SE$
- 9:         **for**  $j \in FS(i)$  **do**
- 10:             **if**  $l(j) > l(i) + t_{ij}$  **then**
- 11:                  $l(j) \leftarrow l(i) + t_{ij}$
- 12:                  $pred(j) \leftarrow i$
- 13:                  $SE \leftarrow SE \cup \{j\}$
- 14:             **end if**
- 15:         **end for**
- 16:     **end while**
- 17: **end procedure**

## Label correcting algorithm

```
1: Input: Graph  $G(N, A)$ , costs  $t$ , and source  $r$ 
2: Output: Optimal cost labels  $l$  and predecessors  $pred$ 
3: procedure LABELCORRECTING( $G, t, r$ )
4:    $SE = \{r\}$ 
5:    $l(i) \leftarrow \infty, \forall i \in N \setminus \{r\}; l(r) \leftarrow 0$ 
6:    $pred(i) \leftarrow \text{NA}, \forall i \in N \setminus \{r\}; pred(r) \leftarrow 0$ 
7:   while  $SE \neq \emptyset$  do
8:     Remove an element  $i$  from  $SE$ 
9:     for  $j \in FS(i)$  do
10:      if  $l(j) > l(i) + t_{ij}$  then
11:         $l(j) \leftarrow l(i) + t_{ij}$ 
12:         $pred(j) \leftarrow i$ 
13:        if  $j$  not in  $SE$  then
14:           $SE = SE \cup \{j\}$ 
15:        end if
16:      end if
17:    end for
18:  end while
19: end procedure
```

▷ Scan Eligible List

## Single pair shortest path

## A\* algorithm

- ▶ This algorithm requires a heuristic cost  $h(i)$  of reaching destination  $s$  from any node  $i$ .  $h(i)$  should be a lower bound on the value of cost of reaching from  $i$  to  $s$ . In highway networks,  $h(i)$  can be taken as the Euclidean distance between  $i$  and  $s$  divided by the highest speed possible in the network.
- ▶ The Dijkstra's algorithm can be slightly modified to convert it into A\* algorithm. Make the following changes in Line 8.  
*Choose a node  $i$  with minimum  $l(i) + h(i)$*   
Stop the algorithm if  $i = s$ .

## Shortest path in Directed Acyclic Graph (DAG)



## Directed acyclic graphs and topological ordering

**Definition (Directed acyclic graph (DAG)).** A directed graph is DAG if does not contain any directed cycle.

**Definition (Topological ordering).** We say that a labeling *order* of a graph is **topological ordering** if  $\forall (i, j) \in A$ , we have  $order(i) < order(j)$ . A network containing directed cycle cannot be topologically ordered.

Conversely, a directed acyclic graph can be topologically ordered.

```

1: Input: Graph  $G(N, A)$ 
2: Output: Topological ordering  $order$  of  $N$ 
3: procedure TOPOLOGICALORDERING( $G$ )
4:    $inDegree(i) \leftarrow 0, \forall i \in N$ 
5:    $order(i) \leftarrow \text{NA}, \forall i \in N$ 
6:    $count \leftarrow 1$ 
7:   for  $(i, j) \leftarrow A$  do
8:      $inDegree(j) \leftarrow inDegree(j) + 1$ 
9:   end for
10:   $Q \leftarrow \{n \in N : inDegree(n) = 0\}$ 
11:  while  $Q \neq \phi$  do
12:    Remove "next" node  $i$  from  $Q$ 
13:     $order(j) \leftarrow count$ 
14:     $count = count + 1$ 
15:    for  $j \in FS(i)$  do
16:       $inDegree[j] \leftarrow inDegree[j] - 1$ 
17:      if  $inDegree[j] == 0$  then
18:         $Q \leftarrow Q \cup \{j\}$ 
19:      end if
20:    end for
21:  end while
22:  if  $count < |N|$  then
23:     $G$  has cycle(s)
24:  else
25:     $G$  is acyclic and return  $order$ 
26:  end if
27:  return  $order$ 
28: end procedure

```

## Shortest path in acyclic networks

Remember that we can always order nodes in acyclic networks  $G(N, A)$  such that  $order(i) < order(j), \forall (i, j) \in A$  in  $O(|A|)$  time.

- 1: **Input:** Graph  $G(N, A)$ , costs  $\mathbf{t}$ , and source  $r$
- 2: **Output:** Optimal cost labels  $l$  and predecessors  $pred$
- 3: **procedure** SHORTESTPATHSDAG( $G, \mathbf{t}, s$ )
- 4:      $l(i) \leftarrow \infty, \forall i \in N \setminus \{r\}; l(r) \leftarrow 0$
- 5:      $pred(i) \leftarrow \mathbf{NA}, \forall i \in N \setminus \{s\}; pred(r) \leftarrow 0$
- 6:      $order \leftarrow \text{TOPOLOGICALORDERING}(G)$
- 7:     **for** each node  $i$  in  $order$  **do**
- 8:         **for**  $j \in FS(i)$  **do**
- 9:             **if**  $l(j) > l(i) + t_{ij}$  **then**
- 10:                  $l(j) \leftarrow l(i) + t_{ij}$
- 11:                  $pred(j) \leftarrow i$
- 12:             **end if**
- 13:         **end for**
- 14:     **end for**
- 15: **end procedure**

## Suggested reading

1. BLU Book Chapter 2
2. AMO Chapter 4 and 5

Thank you!