The Shortest Path Problem

Jesper Larsen & Jens Clausen

jla,jc@imm.dtu.dk

Department of Management Engineering

Technical University of Denmark

The Shortest Path Problem

- Given a **directed** network $\mathcal{G} = (V, E, w)$ for which the underlying undirected graph is connected.
- Furthermore, a **source vertex** r is given.
- **Objective:** Find for each $v \in V$ a shortest directed path from r to v (if such one exists).
- Let n denote the number of nodes and m the number of edges in $\mathcal{G}$.

Integer Programming Formulation

- Suppose r is the source vertex. Look at the **number** of paths leaving a vertex vs. the **number** of paths entering a vertex.
  - For r $n-1$ paths have to leave r.
  - For any other vertex, the number of paths entering the vertex must be exactly 1 larger than the number of paths leaving the vertex.
- Let x_e denote the **number** of paths using each edge $e \in E$.

Mathematical Model

This gives the following mathematical model:

$$\begin{array}{lll}
\min & \sum_{e \in E} w_e x_e & \\
s.t. & \sum_{i \in V} x_{ir} - \sum_{i \in V} x_{ri} = -(n-1) & \\
& \sum_{i \in V} x_{ij} - \sum_{i \in V} x_{ji} = 1 & j \in \{2, \ldots, n\} \\
& x_e \in \mathcal{Z}_+ & e \in E
\end{array}$$

Feasible potentials

- Consider an n-vector $y = y_1, \ldots, y_n$.
- If y satisfies that $y_r = 0$ and

$$\forall (i,j) \in E : y_i + w_{ij} \geq y_j$$

then y is called a **feasible** potential.

Feasible potentials II

- If P is a path from r to $v \in V$, then if y is a feasible potential, $w_P \geq y_v$:

$$\begin{aligned} w_P &= \textstyle\sum_{i=1}^{k} w_{e_i} \\ &\geq \textstyle\sum_{i=1}^{k} (y_{v_i} - y_{v_{i-1}}) \\ &= y_{v_k} - y_{v_0} \\ &= y_v \end{aligned}$$

Basic algorithmic idea

- Start with the potential with $y_r = 0$ and $y_i = \infty,\ i \in V \setminus \{r\}$
- Check for each edge if the potential is feasible
- If YES - Stop - the potentials identify shortest paths
- If an edge (i,j) violates the feasibility condition, update y_j - this is sometimes called “correct (i,j)” or “relax (i,j)”

Ford’s Shortest Path Algorithm

$y_r \leftarrow 0, y_i \leftarrow \infty$ for all other i
$p_r \leftarrow 0, p_i \leftarrow$ NIL for all other i
while *an edge exists* $(i,j) \in E$ *such that* $y_j > y_i + w_{ij}$ **do**
    $y_j \leftarrow y_i + w_{ij}$
    $p_j \leftarrow i$

Problem with Ford's Algorithm

- **Complexity !** Beware of *negative length circuits* - these may lead to infinite computation.
- **Solution:** Use the same sequence for the edges in each iteration.

Bellman-Ford's Shortest Path Algorithm

$y_r \leftarrow 0; y_i \leftarrow \infty$ for all other i
$p_r \leftarrow 0; p_i \leftarrow$ Nil for all other i
$k \leftarrow 0$
while $k < n$ *and* $\neg(y$ *feasible*$)$ **do**
    $k \leftarrow k + 1$
    for $(i, j) \in E$ **do** /* correct (i, j) */
        if $y_j > y_i + w_{ij}$ **then**
            $y_j \leftarrow y_i + w_{ij}$
            $p_j \leftarrow i$

Complexity of Bellman-Ford's Algorithm

A worst-case time complexity analysis leads to the following conclusions:

- Initialization: $O(n)$.
- Outer loop: $(n - 1)$ times.
- In the loop: each edge is considered one time - $O(m)$.
- All in all: $O(nm)$.

Correctness of Bellman-Ford's Algorithm

- Proof is based on induction.
- The induction hypothesis is: After iteration k of the main loop, y_i contains the length of any shortest path with *at most* k edges from 1 to i for any $i \in V$.
- For the base case $k = 0$ the induction hypothesis is trivially fulfilled. ($y_r = 0 =$ shortest path from r to r).

Correctness of Bellman-Ford's Algorithm II

- **For the inductive step**, we assume that $y_{v_{i-1}}$ = shortest path from r to v_{i-1} after the $(i-1)$'st pass. Then (v_{i-1}, v_i) is relaxed in the i'th iteration. So y_{v_i} = shortest path from r to v_i.
- **If all distances are non-negative**, a shortest path containing at most $(n-1)$ edges exists for each $v \in V$. If negative edge lengths are present, the algorithm still works. If a *negative length circuit* exists, this can be discovered by an extra iteration in the main loop. If at least on y_i changes, there is a negative length circuit.

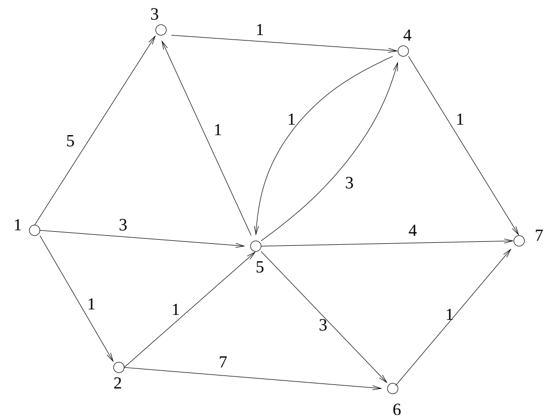
Introduction to Dijkstra's Algorithm

- If we assume all edge weights are non-negative we can derive a more efficient algorithm.
- A new algorithm for a general graph could therefore be: Find the most negative edge weight w_e and add $|w_e|$ to all edge weights. Now all $w_f \geq 0$ for all $f \in E$. Use the Dijkstra algorithm. Does that work????

Dijkstra's Algorithm for Shortest Path

$S \leftarrow \{r\}, T \leftarrow \emptyset$
$p_r \leftarrow 0, y_r \leftarrow 0$
$p_i \leftarrow \textsc{Nil}, y_i \leftarrow \infty$ for all other i
while $S \neq \emptyset$ **do**
  select a $i \in S$ minimizing y_i
  for $\{j \in V : j \notin T \wedge (i,j) \in E\}$ **do**
    if $y_j > y_i + w_{ij}$ **then**
      $y_j \leftarrow y_i + w_{ij}, p_j \leftarrow i$
      $S \leftarrow S \cup \{j\}$
  $S \leftarrow S \setminus \{i\}, T \leftarrow T \cup \{i\}$

Example



Complexity of Dijkstras Algorithm

- The only difference to Prim's algorithm for Minimum Spanning Trees is the update step in the inner loop, and this step takes - like in the MST algorithm - $O(1)$.
- Hence the complexity of the algorithm is $O(n^2)$ if a list representation of the y vector is used, and a complexity of $O(mlogn)$ can be obtained if the `heap` data structure is used for the representation of y vector.

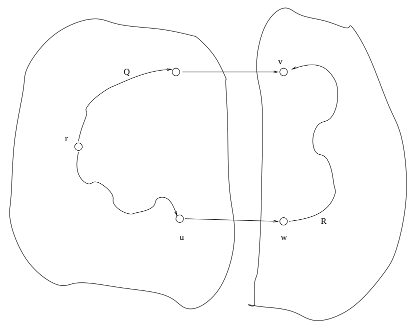
Correctness of Dijkstras Algorithm

- This proof is made by induction:
- Suppose that before an iteration of the `while` loop it holds that
  1. for each vertex $u \in T$, the shortest path from r to u has been found and is of length y_u, and
  2. for each vertex $u \notin T$, y_u is the shortest path from from r to u with all vertices except u belonging to T.
- This is obviously true initially.

Correctness of Dijkstras Algorithm II

- Let v be the element with least y value picked initially in the inner loop of iteration k.
- y_v is the length of a path Q from r to v passing only through vertices in T.
- Suppose that this is not the shortest path from r to v - then another path R from r to v is shorter.

Correctness of Dijkstras Algorithm III



$y_w >= y_v (= \text{length}(Q)) \Rightarrow \text{length}(R) \geq \text{length}(Q)$

Correctness of Dijkstras Algorithm IV

- Look at R: R starts in r, which is in T. Since v is not in T, R has an edge from a vertex in P to a vertex not in T.
- Let (u, w) be the first edge of this type. w is a candidate for vertex choice in the current iteration, where v was picked. Hence $y_w \geq y_v$.
- If all edge lengths are non-negative, the length of the part of R from w to v is non-negative, and hence the total length of R is at least the length of the path Q.

Correctness of Dijkstras Algorithm V

- This is a contradiction – hence Q is a shortest path from r to v.
- Furthermore, the update step in the inner loop ensures that after the current iteration it again holds for u not in P (which is now the “old” P augmented with v) that y_u is the shortest path from from r to u with all vertices except u belonging to P.

Floyd-Warshall’s all-to-all Algorithm

$y_{ij} \leftarrow w_{ij}, p_{ij} \leftarrow i$ for all (i, j) with $w_{ij} \neq \infty$, $p_{ij} \leftarrow 0$ otherwise.
for $k \leftarrow 1$ *to* n **do**
    for $i \leftarrow 1$ *to* n **do**
        for $j \leftarrow 1$ *to* n **do**
            if $i \neq k \wedge j \neq k \wedge y_{ij} > y_{ik} + y_{kj}$ **then**
                $y_{ij} \leftarrow y_{ik} + y_{kj}$
                $p_{ij} \leftarrow p_{kj}$

Complexity of Floyd-Warshall’s Algorithm

- In addition to the initialisation, which takes $O(n^2)$, the algorithm has three nested loops each of which is performed n times.
- The overall complexity is hence $O(n^3)$.

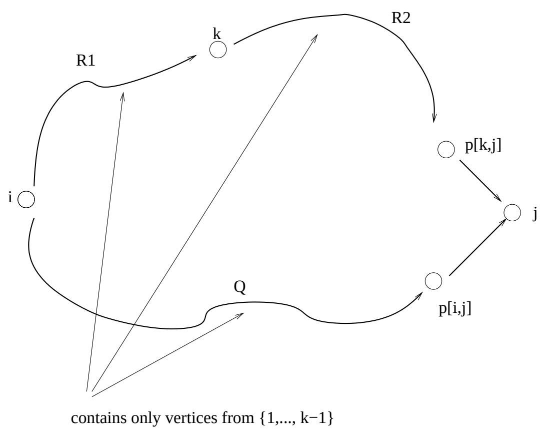
Correctness of Floyd-Warshall's Algorithm

- This proof is made by induction:
- Suppose that prior to iteration k it holds that for $i, j \in v$ y_{ij} contains length of the shortest path Q from i to j in G containing only vertices in the vertex set $\{1, ..., k-1\}$, and that p_{ij} contains the immediate predecesor of j on Q.
- This is obviously true after the initialisation.

Correctness of Floyd-Warshall's Algorithm II

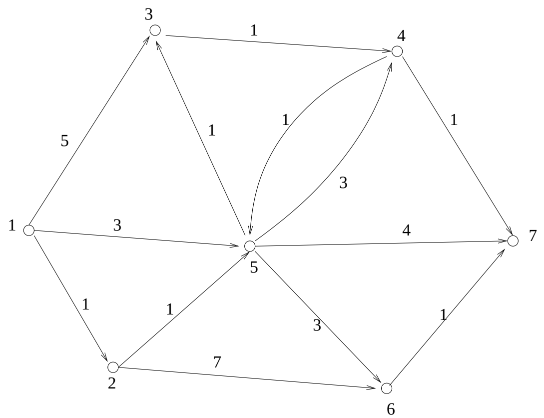
- In iteration k, the length of Q is compared to the length of a path R composed of two subpaths, $R1$ and $R2$.
- $R1$ is a path from i to k path with "intermediate vertices" only in $\{1, ..., k-1\}$, and $R2$ is a path from k to j path with "intermediate vertices" only in $\{1, ..., k-1\}$. The shorter of these two is chosen.

Correctness of Floyd-Warshall's Algorithm III



Correctness of Floyd-Warshall's Algorithm III

- The shortest path from i to j in G containing only vertices in the vertex set $\{1, ..., k\}$ either
  1. does not contain k - and hence is the one found in iteration $k-1$, or
  2. contains k - and then can be decomposed into a path from i to k followed by a path from k to j, each of which has been found in iteration $k-1$.
- Hence the update ensures the correctness of the induction hypothesis after iteration k.


1
2
3
4
5
6
7
5
1
1
3
1
3
4
1
1
1
7
3
1