Combinatorial optimization for undergraduate students

Lecture note 5. Shortest path-Dijkstra's algorithm and Train scheduling

Lecturer : O-joung Kwon
Spring, 2018

We discuss a system of equations, called Bellman's equation, which has to be satisfied by the distances $d(s, v)$ from a specified vertex s.

We assume that the given network (G, w) does not contain any cycles of negative length. Let $V(G) := \{1, 2, \ldots, n\}$, and let $w_{i,j} := w(ij)$ for an edge ij. Let u_i denote the distance $d(s, i)$ where $s = 1$ and it is a starting vertex. We also assume that every vertex is accessible from $s = 1$.

Proposition 1 (Bellman's equations)**.** *Let* $s = 1$. *Then*
(B)
$u_i = 0$ *and* $u_i = \min\{u_k + w_{ki} : i \neq k\}$ *for* $i = 2, \ldots, n$.

Unfortunately, even though we assumed that (G, w) does not contain any cycles of negative length, Bellman's equation may have two solutions.

Exercise 1. *Find a general construction providing infinitely many examples for this phenomenon.*

Bellman's equation have a unique solution under the stronger assumption that (G, w) contains only cycles of positive length.

Theorem 1. *If* 1 *is the root of* G *and if all cycles of* G *have positive length with respect to* w*, then Bellman's equations have a unique solution, namely, the distances* $u_j = d(1, j)$*.*

Bellman's equation in a network having all length non-negative can be solved by the algorithm of Dijkstra (1959), which is the most popular algorithm for finding shortest paths.

Let (G, w) be a network, where G is a graph and all lengths $w(e)$ are non-negative. The adjacency list of a vertex v is denoted by A_v. We want to calculate $d(s, v)$ for vertices v.

Algorithm 1 Dijkstra's algorithm

procedure DIJKSTRA$(G, w, s; d)$
    $d(s) \leftarrow 0$, $T \leftarrow V(G)$;
    for $v \in V(G) \setminus \{s\}$, assign $d(v) \leftarrow \infty$
    While $T \neq \emptyset$ do
        find some $u \in T$ such that $d(u)$ is minimal;
        $T \leftarrow T \setminus \{u\}$;
        for $v \in T \cap A_u$ do $d(v) \leftarrow \min\{d(v), d(v) + w(uv)\}$
end procedure

Theorem 2. *Algorithm 1 determines with complexity $\mathcal{O}(|V(G)|^2)$ the distances with respect to some vertex s in (G, w). More precisely, at the end of the algroithm, we have $d(s, t) = d(t)$ for each vertex t.*

Proof continue

Note that the algorithm of Dijkstra may fail if there are negative weights in the network, even if no cycles of negative length exist.

Exercise 2. *Modify Dijkstra's algorithm in such a way that it actually gives a shortest path from s to t, not just the distance $d(s,t)$. Construct an SP-tree for (G,w).*

An example of Dijkstra's algorithm

Train Schedules - using the algorithm of Dijkstra, we want to find optimal connections in a public transportation system. (Like we want to go from INU to Seoul Main Station using buses and subways...)

The state railway company of the Netherlands uses such a schedule information system based on Dijkstra's algorithm. See [Trains, an active time-table searcher. Siklóssy, Tulp, ECAI'89].

For the sake of simplicity, we restrict our interpretation to train lines and train stations, we have our trains begin their runs at fixed intervals.

We construct a digraph G as follows:

- $V(G)$ is the set of train stations,
- each edge from u to v has weight $f(uv)$ that represents the travel time when traveling from v station to w station.

A *line* is a path or a cycle (for example, the second 'line' of Incheon Subway).

For each line L, we associate a time interval T_L representing the amount of time between two consecutive trains of this line (like trains of this line comes every 15 minutes.).

For each station v on a line L, we define the time cycle $t_L(v)$ which specifies at which times the trains of line L leave station v; this is stated modulo T_L.

We assume for simplicity, the train does not take time in each station. So then for a line $v_0 - e_1 - v_1 - e_1 \cdots - e_n - v_n$, we have

- $t_L(v_0) := s_L(\mod T_L)$
- $t_L(v_i) = t_L(v_{i-1}) + f(e_i)(\mod T_L)$ for $i = 1, \ldots, n$.

We calculate the wating times involved in changing trains.

We want to change the train at station v. Consider $e = uv$ and $e' = vw$ that are edges in different lines L and L', respectively. A train of line L leaves the station v at the times

$$t_L(v), t_L(v) + T_L, t_L(v) + 2T_L, \dots$$

and a train of line L' leaves the station v at the times

$$t_{L'}(v), t_{L'}(v) + T_{L'}, t_{L'}(v) + 2T_{L'}, \dots$$

To simplify matters, we assume all time cycles are actually the same as T. Then the waiting time at station v is given by $w(v_{LL'}) = t'_L(v) - t_L(v)(\mod T)$.

Exercise 3. *Reduce the case of different time cycles to the special case where all time cycles are equal.*

We now construct a new graph G' so that we can apply Dijkstra's algorithm to obtain the shortest path.