Shortest Route in a Map

A mathematical formulation using Dijkstra's Algorithm

- Hair Albeiro Parra Barrera -

Hair Albeiro Parra Barrera
jair.parra@outlook.com
March 15, 2020

Contents

1 Problem Formulation

Problem 1. Consider a map, and two points: the **starting point** and the **end point**. Your task is to come up with a mathematical formulation for finding the fastest route between these two points. Assume the following constraints and simplifications:

- Assume you go by car, and the traffic is constant at any time of the day.
- Taking a highway is twice as fast as taking a road.
- Multiple routes may be output.
- For simplicity, let the time it takes to go from one point to another to be constant/proportional to the distance between those points (satisfying the constraint above).

2 Solution

Consider a situation as stated above.

- Let M be the map in question.
- Interpret M as a graph consisting of n nodes, $s_1, \ldots, s_n$ such that one can go from location $s_i \to s_j$ directly ($i, j \in \{1, \ldots, n\}$).
- Let $\vec{s_{ij}}$ be the vector starting from s_j and ending in s_j.
- Define $d(\vec{s_{ij}}) = d(s_i, s_j) = d_{ij}$, $d : M \to \mathbb{R}^+$ to be the distance between adjacent nodes s_i and s_j.
- Define $t(\vec{s_{ij}}) = t(s_i, s_j) := t_{ij}$, $t : M \to \mathbb{R}^+$ to be the time taken to go from point s_i to adjacent point s_j, and let $t_{ij} \propto d_{ij} \iff t_{ij} = \alpha * d_{ij}$ (that is, time is proportional to distance).
- If t_{day} is time of the day, then traffic is irrelevant by assumption, and so $t_{ij} = k \perp\!\!\!\perp t_{day}, k \ \ \forall t_{day}, k$ (i.e. time taken is the same at all times).
- Let $type(\vec{s_{ij}}) \in \{\text{highway}, \text{road}\}$. The constraint tells us that

$$t_{ij} = \begin{cases} 2\alpha * d_{ij} \ , \ type(\vec{s_{ij}}) = \text{"highway"} \\ \alpha * d_{ij} \ , \ type(\vec{s_{ij}}) = \text{"side road"} \end{cases} \tag{1}$$

  where $d_{ij}, \alpha \in \mathbb{R}$, and α is some constant factor.

- Let $s_a \xrightarrow{p} s_b = \{s_1 \to s_2, s_2 \to s_3, \ldots s_{k-1} \to s_k\}$ denote a path of length k between s_a and s_b , and let

$$t(s_a \xrightarrow{p} s_b) = \sum_{\substack{i,j \\ \vec{s_{ij}} \in s_a \xrightarrow{p} s_b}} t_{ij} = \sum_{\substack{i,j \\ \vec{s_{ij}} \in s_a \xrightarrow{p} s_b}} t(s_i, s_j) = \sum_{i=1}^{k-1} t(s_i \to s_{i+1}) \tag{2}$$

  i.e. the time taken from point a to point b is the sum of the time taken along every single two adjacent nodes in the path.

2.1 Algorithm

In order to obtain the shortest path, we use the following adaptation of **Dijkstra's algorithm** to find the shortest path:

Algorithm 1 Adapted Dijkstra's Algorithm

INPUT:
- Map $M = (S, E)$ be defined as above, where $S = s_1, \ldots, s_n$ is the set of points in the map, and E is the set of available roads/highways between each of the points;
- positive edge lengths $\{l_e : e \in E\}$; vertex $s \in V$

OUTPUT - For all vertices u reachable from s, $dist(u)$ is set to the distance from s to u, i.e. $dist(u) := d(s \xrightarrow{p} u)$,and $time(u)$ is set to be $t(s \xrightarrow{p} u)$.

PROCEDURE:

$$\text{Let} \quad col(s) = \begin{cases} white \text{ , if } s \text{ has not been visited} \\ grey \text{ , if } s \text{ has been visited, but not all it's descendents.} \\ black \text{ , if } s \text{ has been visited and is completely over} \end{cases}$$

for *all* $u \in V$ **do**
  $dist(u) = \infty$, $time(u) = \infty$
  $col(u) = white$, $prev(u) = nil$
end for
$dist(s) = 0$
$time(s) = 0$

$H =$ `makeheap(`V`)` (using time values as keys)
while H *is not empty:* **do**
  $(u, time(u)) =$ `deletemin(`H`)`
  $col(u) := grey$
  for *all edges* $(u, v) \in E$ **do**
    if $col(v) == white$ **then**
      Check the type of $u \to v$ and calculate $time(u \to v)$ accordingly,
      using equation (1).
      if $time(v) > time(u) + time(u \to v)$ **then**
        $time(v) := time(u) + time(u \to v)$
        $prev(v) := u$
        `decreasekey(`H, v`)`
      end if
    end if
  end for
  $col(u) := black$
end while

References

[1] "Dijkstra's Algorithm" `https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm`