On Determining the Shortest Path through a Number of Intermediate Points

S. Demeyer, P.. Audenaert, and M. Pickavet

Ghent University, Department of Information Technology (INTEC),
sofie.demeyer@intec.ugent.be

Abstract

In this article, an algorithm is presented which determines in a transportation (road) network the shortest path from an origin to a destination that passes by a number of predefined intermediate points, all given by their geographic coordinates. In general, these points do not coincide with nodes of the graph and we may assume that a point is visited as soon as one of the nodes closest to it is visited. The algorithm searches for the shortest path iteratively from one intermediate point to another. It can be proven that, in particular cases, this algorithm realizes a speedup in comparison with a point-to-point algorithm.

Keywords: Shortest Path Algorithm, Intermediate Points, Transport Networks

In this article we will present a solution to a specific route planning problem, which was presented to us by an industrial company. Consider the case where a user wants to get from one point to another in a road network, while on its route stopping by a number of predefined locations. Since the user has no notion of the underlying routing graph, all points are given by their geographic locations. We want to find the shortest route in the graph and assume that a location is visited as soon as one of the nodes close to it is visited. The problem description then formulates as follows: In a given graph, find the shortest path between a node close to the origin point p_0 to a node close to the destination point p_k, which passes by a list of intermediate points $[p_1, p_2, ..., p_{k-1}]$, i.e. passes through nodes close to these points.

The most common way to route in a graph between two geographic coordinates which do not coincide with nodes of the graph, is to find the n closest nodes of these coordinates, and add unidirectional links from the origin point to each of its closest nodes and links from the closest nodes of the destination to the destination point. Subsequently the shortest path can be found between the origin node and the destination node. This has no impact for other users routing on the network, since by making use of unidirectional links other paths cannot pass through these newly created nodes. Applying this technique for intermediate points is no option, since we need both incoming as outgoing links, which

would allow users to pass through nodes created by other users. Therefore, we have developed an algorithm which avoids adding these links.

The algorithm is based on the well-known algorithm of Dijkstra [1], which is a label-setting algorithm. In contrast with Dijkstra, our algorithm allows multiple labels per node. Additionally, we will add a level to the labels which indicates the iteration in which this label has been assigned. These measures deal with the fact that depending upon the location of the intermediate points, nodes can be visited multiple times in the same route. The algorithm can be sketched as follows. First determine for both the origin, destination and each of the intermediate points the n closest nodes in the network. Initialize the level to 0 and the temporary set to the set of nodes closest to the origin point. Similar to the algorithm of Dijkstra, build the shortest path tree from these nodes, until all the closest nodes to the first intermediate point are found. Increase the level by 1 and re-initialize the temporary set with the nodes closest to this intermediate point, keeping the labels from the previous iteration. Again, build the shortest path tree from these nodes to the closest nodes of the next intermediate point. Repeat these steps until the labels of the nodes closest to the destination point are found. The smallest label of these nodes represents cost of the shortest path and this path can be reconstructed by tracking back in each level to the previous 'intermediate node'.

The algorithm has one variable parameter, namely the number of closest nodes n. Experiments have shown that, in a transportation network, $n = 5$ is a good value for this parameter. The performance of the algorithm is comparable with that of the algorithm of Dijkstra. Moreover, the calculation time is strongly dependent upon the relative position of the intermediate points. If they are laying criss-cross, there are parts of the network which will be investigated multiple times. The calculation time then is more or less proportional to the size of the resulting route. If the intermediate points are situated in order near the shortest path between the origin and the destination, the presented algorithm can realize a speed-up, known as the sub-goal method [2]. In this case, we can even drop the concept of levels and have a single label for each node.

In conclusion, we have presented a novel algorithm which finds the shortest route from an origin point passing by an ordered list of intermediate points to a destination point, in which each point can be represented by its n closest nodes. From each of these points it builds a shortest path tree to the next point. It can be proven that the algorithm has a calculation time which is similar to that of the algorithm of Dijkstra [1] and, in some particular cases, even realizes a speed-up.

References

[1] Dijkstra EW., A Note on Two Problems in Connexion with Graphs, Numerische Mathematik, 1, 269-271 (1959)

[2] Fu L et al., Heuristic shortest path algorithms for transportation applications: State of the art, Computers and Operations Research, 33, 3324-3343 (2006)