Multimodal Transportation Route Planning in Private Transport Mode Network

Careelika Liisi Kuik
University of Tartu
Institute of Computer Science
Email: clk@ut.ee

***Abstract* — the aim of this seminar paper is to present the design for the algorithm for multimodal transportation route planning in private transport mode.**

I. INTRODUCTION

The aim of this seminar paper is to present the network and algorithm design for multimodal transportation route planning in private transport mode. In designing and implementing multimodal transportation solutions, different aspects need to be taken into account. The paper consists of two parts: finding unimodal trajectories for each mode separately and, secondly, combining the trajectories from the previously found unimodal trajectories for constructing multimodal trajectories.

The paper is structured as follows: first, the structure of the road network is described, next, the design of both algorithms is presented and the used tools are described. Third, the results of initial testing are presented. Lastly, conclusions are made based on the results and finally, the conclusion is done.

II. THE DESCRIPTION OF THE STRUCTURE OF THE ROAD NETWORK

Firstly, the structure of the road network is described. In multimodal transportation different means of transport, both public (e.g. bus, train, ferry) and private (e.g. bicycle, car, pedestrian) [1] can be described as separated network mode layers. In this work we focus on finding multimodal paths in private transportation modes. The route data can be extracted from source database separately for each mode or all modes together. Extracting the data for each mode separately as different layers, makes data easily manageable and accelerates computing the result using the path finding algorithm. This way candidate trajectories for each mode are found separately and later compared and combined to formulate multimodal trajectory result. While this approach makes finding unimodal trajectories easier and faster, it also makes constructing multimodal trajectories more complex and time consuming. This is caused by the fact that there might not be links between different nodes that belong to different modes. The corresponding siblings from other modes have to be found in the process of constructing multimodal trajectories. On the other hand, the amount of the nodes to be processed during the construction of the multimodal trajectory is notably smaller than the amount of nodes that would have to be processed while finding the best trajectories from the multimodal network. For this reason, the approach of finding the unimodal trajectories separately was used in this work.

III. EXTRACTING AND PREPROCESSING THE SOURCE DATA

The source data for the program was extracted from OpenStreetMap (OSM). OSM is an open source project created and updated by volunteers and its data is free for use [2]. As the data is updated by volunteers, it is not as complete as the data from non-free sources like Google Maps. There can be missing or faulty nodes and links in the data.

The data is extracted using Overpass API. Overpass API is a read-only API that serves as a database over the web returning a required dataset in response of the query made by the user [3]. This data is later processed with a tool named osm2po. The tool osm2po is a free open source tool that converts OSM files to routing database tables [4]. As the next step, the data is imported in the PostgreSQL database, that has PostGIS and pgRouting extensions. PostGIS is a spatial database extender for PostgreSQL that adds support for geographic objects and running location queries in SQL [5]. PgRouting is an PostGIS extension that adds geospatial routing functionality [6]. As a result of these steps a database with routing data is created and the data can be used by the program to query for the necessary source data.

IV. THE ALGORITHM FOR UNIMODAL PATH FINDING

A. The algorithm

To solve the previously mentioned unimodal best path finding problem, a modification of A Star algorithm is used. A Star algorithm searches the right path taking into account travel time from both nodes – the source node and destination node – simultaneously. It does so, by calculating at each intermediate point the average travel time from the start point and the remaining estimated travel time for the endpoint for each candidate trajectory. [7] The pseudo code of A Star algorithm is shown on Figure 1[8].

```
A* (start, goal)
Closed set = the empty set
Open set = includes start node
G[start] = 0, H[start] = H_calc[start, goal]
F[start] = H[start]
While Open set ≠ Ø
    do CurNode ← EXTRACT-MIN- F(Open set)
    if ( CurNode == goal ), then return BestPath
    For each Neighbor Node N of CurNode
        If ( N is in Closed set ), then Nothing
        else if ( N is in Open set ),
            calculate N's G, H, F
            If ( G[N on the Open set] > calculated G[N] )
                RELAX(N, Neighbor in Open set, w)
                N's parent=CurNode & add N to Open set
        else, then calculate N's G, H, F
                N's parent = CurNode & add N to Open
```

Figure 1. The pseudo code of A Star algorithm[8]

For calculating the remaining estimated travel time, the average speed for the mode and the Haversine distance between the current point and the endpoint of the trajectory are used in calculating the estimated remaining travel time. To speed up the calculation, the obtained estimated total travel time is used as cost in priority list where trajectory candidates found until the current point, are stored. As a modification, the priority list only allows a limited amount of entries for the same cost. On each iteration the candidate trajectory with the smallest cost - the smallest estimated travel time is continued. As the costs can be the same for endless amount of trajectories, a constant can be configured to define, how many candidates with the same cost are preserved in the priority list. If this amount is exceeded, the oldest candidate with the cost will be thrown out of the list. This decreases the probability of finding the overall fastest path. But due to the nature of A Star algorithm, the original algorithm itself does not ensure finding the best path with 100 percent probability. Taking this into account, adding this kind of constraint does not degrade the probability of finding the best path in a great amount. To increase the probability of finding the fastest trajectory, another parameter is introduced.

This parameter marks the amount of the results that are computed by the algorithm.

As the user can select any latitude, longitude point as the start location and any latitude, longitude as the end location, the node with exactly the same latitude, longitude cannot always be found. For this reason, the missing start and end segment from the start and end location selected by the user, is added in freestyle mode, as there exists no suitable defined path in the source data.

Among this amount of possible trajectories between the start and the endpoint the fastest one is selected for the multimodal comparison. This algorithm is executed for each mode separately and the fastest trajectories of each mode are used in the multimodality algorithm.

B. Testing, results and analyzes

For testing the algorithm, the data of the city of Tallinn was used. Two random points were selected as the start point and the end point. For each mode the route from the start point to the end point was found using the program. Figure 2 shows an example of the best route extracted in car mode, figure 3 shows an example of the best route extracted in bicycle mode and figure 4 shows an example of the best route extracted in pedestrian mode.

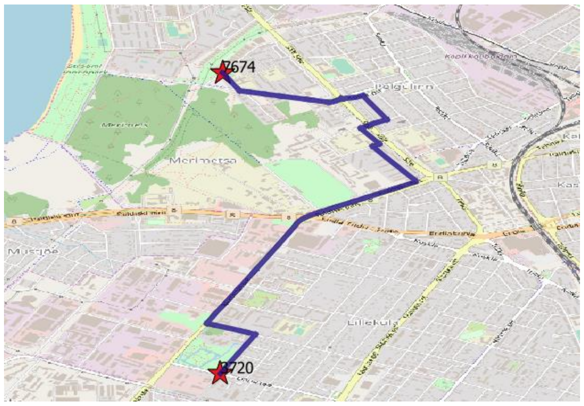


Figure 2. An example of the best route extracted in car mode. Red stars represent the start and end points of the route and blue line represents the trajectory between the start and end point.

It can be noticed that the extracted trajectories are probably not the best ones. This occurs due to the allowed inaccuracy for the algorithm. As mentioned before, the algorithm does not ensure finding the best path. Furthermore, some road segments might be missing from the source data and thus not considered in the selection. The best trajectories form each of the three modes can be seen on figure 5. As the result, from the computed trajectories in three modes, the trajectory in car mode is the trajectory with

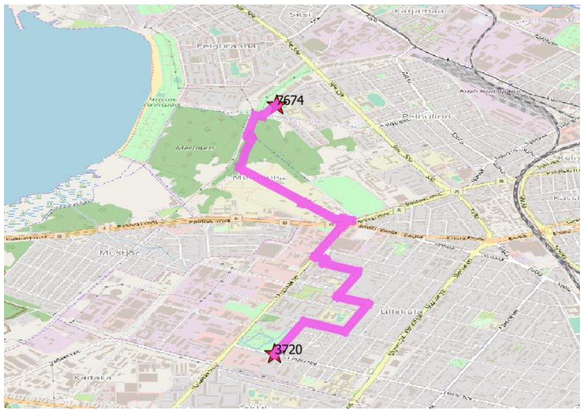


Figure 3. An example of the best route extracted in bicycle mode. Red stars represent the start and end points of the route and violet line represents the trajectory between the start and end point.

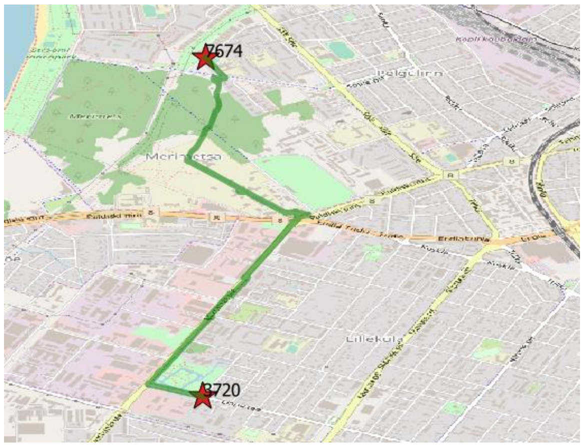


Figure 4. An example of the best route extracted in pedestrian mode. Red stars represent the start and end points of the route and green line represents the trajectory between the start and end point.

the smallest overall travel time. And the best trajectory in pedestrian mode is the slowest one among the three. Therefore, if choosing the best path among unimodal trajectories, the trajectory in car mode would be the fastest one.

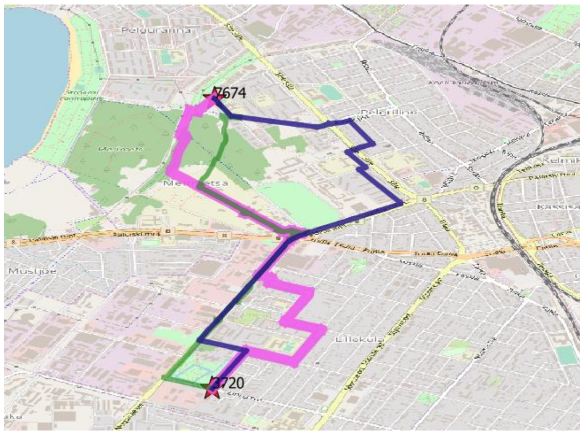
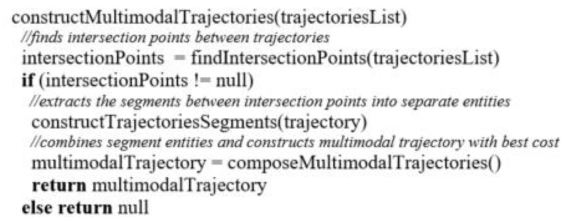


Figure 5. The best routes extracted in each of the three modes. Red stars represent the start and end points of the route. The blue line represents car mode trajectory, the pink line represents bicycle mode trajectory and the green line represents the pedestrian mode trajectory.

V. THE ALGORITHM FOR CONSTRUCTING MULTIMODAL TRAJECTORIES

1) *The algorithm*

The algorithm for multimodality takes as input the candidate trajectories from all selected modes. For the nodes in every trajectory, it finds the closest nodes from other trajectories. To decide, which points are intersection points and which not, Haversine distance between the nodes is calculated and compared to the distances between other nodes. Additionally, as the intersection points might not be at exactly the same coordinates a parameter for the biggest allowed distance between two intersection points is defined. After extracting the intersection points, the segments between each pair of intersection points are added a separate segment entity. Next, the segments entities of the trajectories for each mode are used for combining multimodal trajectory. For each intersection points pair the corresponding segment pair, one from one mode, the other from another mode, are compared and the segment with smallest travel time is selected. For switching between different modes, the travel time between the two intersection points is included in the cost. As a result, if a trajectory composed from different modes is faster than a unimodal trajectory, the trajectory will be returned to the user with the estimated travel time. The pseudocode of the multimodal algorithm is shown on figure 6.



```
constructMultimodalTrajectories(trajectoriesList)
  //finds intersection points between trajectories
  intersectionPoints  = findIntersectionPoints(trajectoriesList)
  if (intersectionPoints != null)
    //extracts the segments between intersection points into separate entities
    constructTrajectoriesSegments(trajectory)
    //combines segment entities and constructs multimodal trajectory with best cost
    multimodalTrajectory = composeMultimodalTrajectories()
    return multimodalTrajectory
  else return null
```

Figure 6. The pseudo code for multimodality algorithm.

A. *Testing, results and analyzes*

For testing the algorithm, the data of the city of Tallinn was used. Two random points were selected as the start point and the end point. For each mode the route from the start point to the end point was found using the previously described algorithm for unimodal path finding. The unimodal trajectories are shown in figures 7,8 and 9.

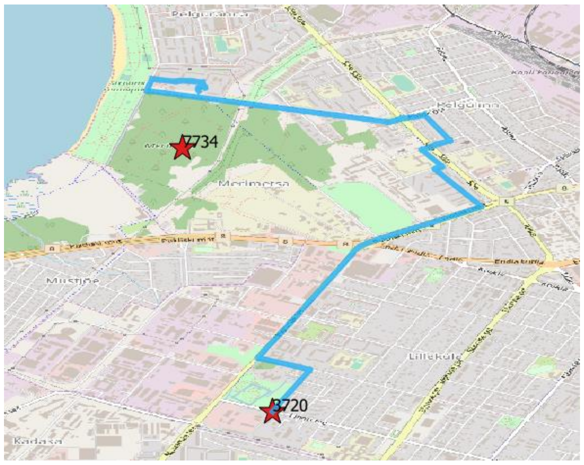


Figure 7. The best route extracted in car mode. Red stars represent the start and end points of the route and the light blue line represents the trajectory in car mode.

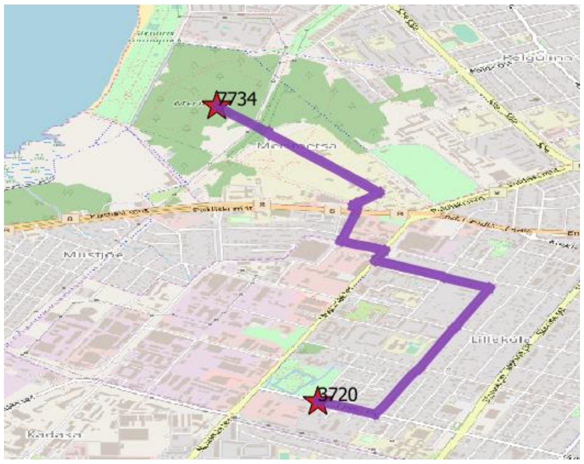


Figure 8. The best route extracted in bicycle mode. Red stars represent the start and end points of the route and the violet line represents the bicycle mode trajectory.

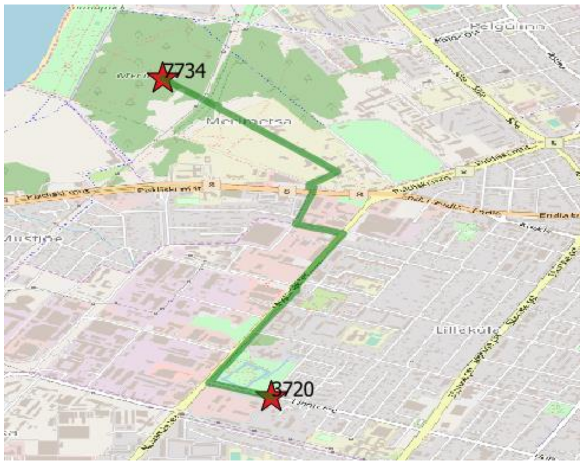


Figure 9. The best route extracted in pedestrian mode. Red stars represent the start and end points of the route and green line represents the pedestrian mode trajectory.

The estimated travel time for the best route extracted in car mode, is 15min 24s. The estimated travel time for the best route extracted in bicycle mode, is 15min 8s. The estimated travel time for the best route extracted in pedestrian mode, is 39min 40s. All the trajectories given as a parameter to the multimodal algorithm, are shown on figure 10.

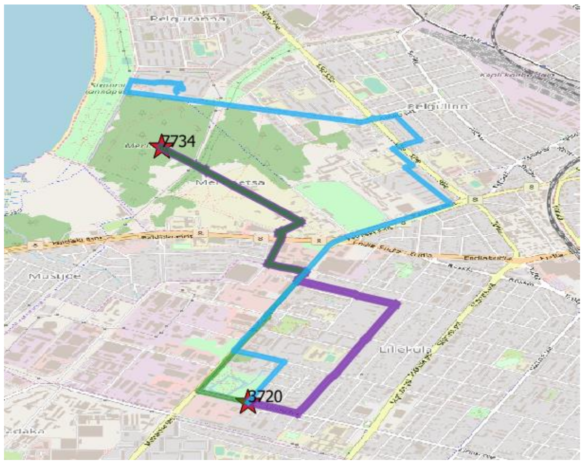


Figure 10. The best routes extracted in each of the three modes. Red stars represent the start and end points of the route. The light blue line represents car mode trajectory, the violet line represents car mode trajectory and the green line represents the pedestrian mode trajectory.

The result found by the multimodal algorithm is shown of figure 11. The estimated travel time for this trajectory is 11min 20s. As can be seen from the figure 11, the best trajectory between the start point and end point is a multimodal trajectory combined from bicycle mode and car mode. Additionally, the missing start and end segment from the start and end location selected by the user, is added. This example shows, that the result of selecting the best trajectory among the unimodal trajectories differs from the result computed by the multimodal algorithm. If in the first case car trajectory was the trajectory with the least travel time, then in this case the multimodal trajectory gives faster trajectory as a result.

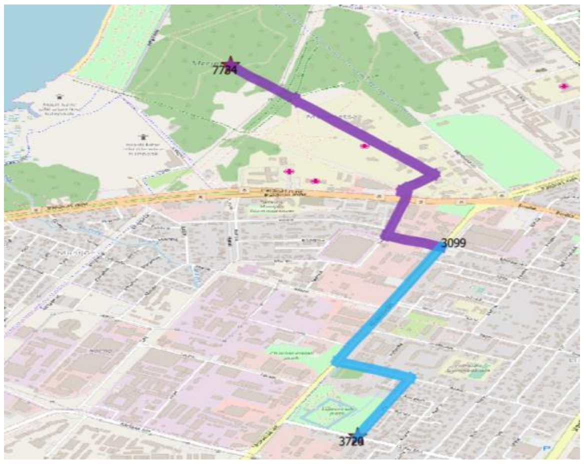


Figure 11. The multimodal trajectory with the smallest travel time constructed by the algorithm. Red stars represent the start and end points of the route. The violet line represents the trajectory travelled by bicycle and the light blue line represents the trajectory travelled by car.

VI. CONCLUSION

In this paper an algorithm based on A star algorithm for unimodal best path finding and another algorithm for combining the found paths in to formulate multimodal trajectory were introduced. The algorithms were tested with the data of the city of Tallinn. Testing showed that, due to a large amount of car routes in the city, car mode will almost always be the fastest. However, other modes can be used if car is not in the selection or there is a larger car route free area on the trajectory, e.g. a park, a river with a bridge only for pedestrians and bicycles, a forest, etc. In this case multimodally combined trajectories can outperform the speed of the unimodal trajectory, even the trajectory in car mode.
Additionally, using user set speed can increase the usage of multimodality in case of bicycle and pedestrian mode as in these modes the speed can be largely affected by the user. In car mode the average speed is dictated by the traffic regulations and traffic condition, thus user cannot affect it in a significant and feasible way.

REFERENCES

[1] Zhang, J., Liao, F., Arentze, T., Timmermans, H., A multimodal transport network model for advanced traveler information systems Procedia Social and Behavioral Sciences 20, 313–322 (2011)
[2] OpenStreetMap https://www.openstreetmap.org/about Last visited 21.05.2018
[3] Overpass API https://wiki.openstreetmap.org/wiki/Overpass_API Last visited 21.05.2018
[4] osm2po http://osm2po.de/ Last visited 21.05.2018
[5] PostGIS https://postgis.net/ Last visited 21.05.2018
[6] pgRouting https://pgrouting.org/ Last visited 21.05.2018
[7] P. E. Hart, N. J. Nilsson, B. Raphael, "A formal basis for the heuristic determination of minimum cost paths",
IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, 1968.
[8] W. J. Seo, S. H. Ok, J. H. Ahn, S. Kang and B. Moon, "An Efficient Hardware Architecture of the A-star Algorithm for the Shortest Path Search Engine," 2009 Fifth International Joint Conference on INC, IMS and IDC, Seoul, 2009, pp. 1499-1502.