Abstract — the aim of this paper is to present the initial design for the algorithm used in my thesis on the topic Multimodal Transportation Route Planning for Navigation Purposes.

I. INTRODUCTION
The aim of this paper is to present the initial network and algorithm design used in my thesis on the topic Multimodal Transportation Route Planning for Navigation Purposes. In designing and implementing multimodal transportation solutions, different aspects need to be taken into account. Thus, when designing and implementing a feasible solution, as the first thing an abstract overview of the multimodal network structure is needed. The structure of the network affects the software design in a great way and in case the features of the network have not been taken into account, the overall solution can be erroneous and inefficient. For the same reason it is necessary to design the algorithm taking into account both the network and the characteristics of the data at as early stage as possible.

The paper is structured as follows: first, the multimodal transportation network is introduced, next, the initial design of the algorithm is discussed, third, the conclusions about the data are made based on the previous discussion and finally, the future work is introduced.

II. MULTIMODAL TRANSPORTATION NETWORK
Firstly, the structure of the transportation network is introduced. In multimodal transportation different means of transport, both public (e.g. bus, train, ferry) and private (e.g. bicycle, car, foot) [1] can be described as separated network mode layers. Describing different means of transport as different layers offers more flexibility and additional information for the route planning algorithm. The layers can only be used together when the data and structure of the network is the same for all layers. Each layer is a graph of nodes that are connect edges. A node represents a place where one has to choose between continuing with the current mode or changing the mode [2]. In subnetworks edges exist between the nodes of the same mode layer. By adding possible edges between the different mode network layers, we connect the subnetworks into a multimodal graph. These additional edges an only be added if the transfer is possible between these nodes. By doing this, we enable routes that contain nodes from different modes of transportation. The cost of changing from one subnetwork to another has to be taken into account by the cost algorithm. Elaborating the network taking into account the navigation side, there shall also be a type of edges that specify a range of travel from source node instead of uniting the source node with another node.
Moreover, depending on the transportation mode, the position of the nodes can change. For example, in case of temporary transport solution one node can only be used during a specified time period. Thus, as an additional feature, the structure of the network would benefit from the ability of adding or removing nodes and edges either temporarily or permanently.

III. THE ALGORITHM

The trivial solution for shortest path finding, would be to use some general shortest path algorithm, like Dijkstra’s algorithm. Dijkstra’s algorithm (Figure 1.) finds the single-source shortest path on a weighted direct graph \( G = (V, E) \). It takes into account the cost of each edge and considers every possible path. For Dijkstra’s algorithm all edge weights have to be nonnegative.\[^3\]

![Figure 1. Dijkstra’s algorithm. where G represents the graph, u and v are the vertices connected by the edge under inspection, source is the source vertex, v.d is shortest-path estimate, v.pi is v’s predecessor attribute, S is the set of vertices whose final shortest-path weights from the source s have already been determined, Q = V-S. \[^3\]](image)

However, using Dijkstra’s algorithm has a number of downsides. Firstly, finding the suitable path in transportation network requires a vast amount of computations. The search begins from the source node and traverses nodes from the queue until finding the destination node. In the worst case, this could mean passing through all the graph and computing all the edges. One way to avoid that, would be using an alternative algorithm like A Star. A Star algorithm searches the right path from both nodes – the source node and destination node – simultaneously. However, using A Star algorithm makes it difficult to take into account the nature of the network and the data, for example earliest possible arrival and the feasibility of the chosen road. In this context, also the problem of time dependent traffic network needs to be solved. Instead, in order to avoid recomputing all costs at every iteration, it is possible to index and precompute the costs for edges. It could involve either precomputing all or only a set of more time consuming calculations. In any case the computed values need to be stored and the access time shall be smaller than the time needed for the computation.

Another problem that is faced when using Dijkstra’s algorithm, is the problem of the feasibility of the route. With respect to comfort, it is not pragmatic to switch transportation modes too often unless it is unavoidable. To avoid such undesirable sequences of routes, label constrained shortest path problem has to be solved. In order to do that, additional data from the edges can be used in weight computing. \[^4\]

Furthermore, taking into account the estimated time for the start of the trip, we can avoid unnecessary computations by excluding the edges that are not available for usage at that particular time. Moreover, the decision making algorithm can be elaborated using additional parameters, for example time limit or maximum distance.

As a result, the algorithm becomes a modified version of Dijkstra’s algorithm.

The algorithm follows the next steps:
1) determining suitable set of nodes and edges taking into account the input parameters
   a. pre-processing costs
      The most popular trajectories can be pre-processed to reduce the amount of time consumed on the queries.
2) finding the shortest path in the defined multimodal graph using modified version of Dijkstra’s algorithm
   a. computing costs for route candidates
   b. comparing route candidates based on costs

For pre-processing the same Dijkstra’s algorithm is used and the results are saved in the database.

In order to increase the speed of the computation, parallel computing approaches could be introduced.

IV. THE CONCLUSIONS ABOUT THE DATA
Firstly, the data of each transportation mode has to be converted to suitable nodes and edges. If possible, additional information should be added to nodes. The more precise is the information about every node, the more precise will be the cost computation. The particular additional data fields depend on the cost function. The precision required for the additional data fields depends on the cost function as well.

It is visible from the network structure that each edge has to have temporal aspect included. For example, in the case of the edges that do not end with a target node. Instead of target nodes they specify a range of travel that can either be a constant or computed using additional data from the source node and/or the input parameters.

V. FUTURE WORK
After the initial network structure and algorithm design has been determined, the next step is to modify the source data adding the necessary missing nodes between different transportation layers. Next, the initial algorithm shall be tested and improved using the data with added nodes. In case there are better alternatives for accomplishing the same result faster or more efficiently, the algorithm shall be modified accordingly. After that, additional input parameters shall be added to the algorithm to improve cost computation and the selection of the route. Finally, the ability to cope with the changing data shall be enhanced as well as the performance.

REFERENCES