Speeding Up Shortest Path Search in Public Transport Networks*

Vladislav Martínek and Michal Žemlička

Charles University, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
VladaM@seznam.cz, Michal.Zemlicka@mff.cuni.cz

Abstract. The searching for the shortest path in public transport networks can take more time than is acceptable for given situation. We have therefore searched for methods that speed up the given calculation. The approach, when the calculation is not performed on the original network but on the simplified one, seems to be very promising. The path found in the simplified network can be easily mapped to a corresponding path in the original network. In the case of the Prague public transport the simplified network has several times less nodes and the computation is speeded up correspondingly.

Keywords: shortest path search in public transport networks, network simplification

1 Introduction

Searching for the optimal connection between two various places is a frequent task solved in public transport networks. The path duration is an important criterion of the searched connection. From the view of graph theory the problem can be seen as a shortest path search. It is possible to keep in mind other criteria derived from user preferences or restrictions when choosing the connection. Traditional approaches to the shortest path search (recent optimizations compared in [1] expect searching in a static network). Most of these approaches are not applicable on dynamic network without additional modifications[1]. The method introduced in [2] is performing data reduction to simplify train network. Such reduction is generally an NP-hard problem. Fortunately on real data the problem is usually solvable within acceptable time [3]. Our paper introduces similar approach to the graph reduction aimed at the urban public transport and at the practical use of this reduced data in mobile devices application.

* This paper was partially supported by the Program "Information Society" under project 1ET100300517 and by the Czech Science Foundation by the grant number 201/09/0983.

[1] For example a bi-directional search would be difficult to introduce in public transport networks, if we do not know the arrival time.

1.1 Basic Solution Approaches

Two basic approaches can be considered when solving the problem. The first approach is to find the path between all vertices and then only return results on query. The situation is complicated in mass transport networks by the fact, that the edge value is determined according to the actual time. The precomputation of the results would mean to find the shortest paths between all vertices for certain time interval. This approach is suitable when the number of queries is relatively high and there is sufficient memory and computation power for reaction to data changes in appropriate time. The hardware requirements may be hard to satisfy for the large networks as described in [4].

The second approach is to find the shortest path directly according to the given parameters. This approach is suitable, if the number of queries is relatively low and if it is possible to find the answer in acceptable time. The advantage against the previous case is that the variable value of the edges does not mean serious complication in our case.

Algorithms used for direct computing of the shortest path derived from the algorithm published by Dijkstra in [5]. Their complexity is typically superlinear with respect to the number of vertices and edges. If we succeed in reducing the size of input graph, the computation speed will be increased significantly.

1.2 Scheduled and Real Traffic

Timetables determine prescript departure times of individual connections which may vary from actual times of departure. There typically occur two types of irregularities. The first ones may occur relatively frequently and may be relatively small. They may be caused by the current density of traffic, weather, road conditions or other relatively predictable effects. One of the user's preferences could be a requirement for reliability of the connection.

The second irregularity type is caused by extraordinary events of a larger impact. They cannot be predicted, and cause relatively large irregularities from the timetables. Typically can cause temporary interference of the carrier into the timetables. These temporary changes in the timetables would be difficult to handle in the case of precomputed results.

To choose the connection so it meets all user preferences is the case of the search algorithm. This paper is focused on the reduction of input data. For the selection of a connection we will consider only one criterion – the path length (duration).

2 Mass Transport Network Representation

Mass transport system can be seen as an oriented multigraph with valued edges.

Pseudoline is a representation of the certain line of the mass transport system or a walk. Every line of the mass transport has its own timetable and route. In order to separate various line directions, it is suitable to represent them as separate pseudolines.

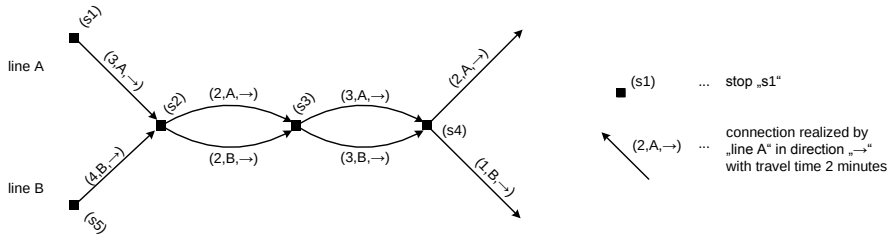

Fig. 1. Graph example

Vertex represents the street refuge of the stop or a platform of the station. Every stop can have several various refuges. In order to count time between refuges in a walk, it is appropriate to represent them as separate vertices.

Edge represents direct connection between two stops, or more precisely refuges. The connection is realized by one of the pseudolines, this information is marked inside the edge structure. If several direct connections are available between two vertices, then every single one of them is represented by the separate edge.

Edge value is expected travel time of the pseudoline between the stops connected by the edge. Travel time may vary according to the time of a day[2].

Waiting time is expected time spent on waiting for the service arrival. This value is added to the edge value in case of transfer between services. It is determined on the basis of actual time and valid timetable of the given line.

There could be various exceptions in the timetables – for example the service has a variable route or is avoiding some stops in certain moments. This situation can be handled by creation of new pseudolines for each type of the exception. The exceptions can be excluded from the original pseudoline and delegated into the new pseudoline. Several new pseudolines can be created on the basis of one line. New pseudoline will be marked in the same way as the original pseudoline for the user.

When searching for the shortest path in mass transport network, it is necessary to count the edge value and also the waiting time in the path length. The waiting time is counted only in the case of transferring between the services or getting in a service. In order to detect the transfers it is necessary to remember the pseudoline, which has been used to get to the vertex. If a new edge is added to the current path and this edge is realized by other pseudoline than the one

[2] For example a “shorted” travel time – B and a normal travel time – A is set for the trams in weekdays according to the departure time from the stop: 0:00-6:59 - B, 7:00-18:59 - A, 19:00-23:59 - B.

used to get to the last vertex in the current path, then the waiting time is to be added to the value of new edge.

The value of the edge realized by the walk should be derived from walk time. Value of the walk edge can markedly vary according to the user preferences.

Recognition of the resulting path Classical algorithms for searching for the shortest path between two vertices in the graph terminates the search at the moment the target is processed. In each vertex there is stored a pointer to the predecessor, from which the vertex was reached. In the case of a multigraph this information is insufficient to recognize the resulting path and it is necessary to remember also the edge leading from the predecessor to the vertex.

Correctness The resulting path found in the reduced graphs must be equal to the path found in the original graph for the equal search parameters. Introducing pseudolines does not change the results presented to the user. Each of the pseudolines holds the identification of the real line, which is presented to the user. The pseudoline is equal to the subset of services of the original line. Transfers between pseudolines are equal to the transfers between original lines. However, the transfers between pseudolines identified by the same line could be redundant[3]. The redundant transfer can occur in the real situation as well, without influence to the resulting path length.

3 Graph Reduction

To reduce the size of input data, we will try to reduce certain edges and vertices in the input graph using appropriate adjustments. Reduction of edges is achieved by a replacement of several original edges with a single new one. The new edge will fully represent all of the original edges when searching for the shortest path. The reduction of vertices occurs so that after reduction of edges some isolated vertices remain in the graph, which are not reasonable to hold for the shortest path search. When the shortest path is found in the reduced graph, it is important to be able to reconstruct the appropriate path in the original graph easily. If the mapping of the path to the original network is too complicated, the advantage of searching in reduced graph could be eliminated. Moreover, the edge values must be maintained, otherwise the condition of the shortest path could be violated.

The following adjustments are intended to be used in the way described. The separate usage of adjustments is possible, but with weaker effect. The result of their use in an opposite order is unsure. The results compared with the previous adjustment are listed in the table below each of them.

3.1 Edge Aggregation

The first adjustment is based on the following heuristics: To get from one stop to the next one in the shortest possible time, it is necessary to get on the service,

[3] An additional transfer may occur if the pseudoline represents the shortened route of the certain line. This can be easily solved by postprocessing of the path found.

which will arrive there first. If several services take the same time to run between those stops, the service with the shortest waiting time should be taken. To find out how long to wait for each connection, we are forced to view timetables of all these connections.

The above situation is in the graph indicated by the two vertices which are connected by several edges with the same value. These edges differ only by pseudoline. We will create a new pseudoline which is a combination of all pseudolines of the edges mentioned above. These edges can be all replaced by a single new edge. The new edge connects the same two vertices as the original edges does and has the same value as the original edges, but it is labeled by a new pseudoline:

Merge-line is a new pseudoline, which is created as a combination of several original pseudolines. The merge-line has its own timetable, which is a combination of the timetables of original pseudolines. The merge-line leaves the stop every time, when one of the original pseudolines leaves the stop. In order to map the path found in new graph to the original graph, it is necessary to remember from which pseudolines the merge-line was created.

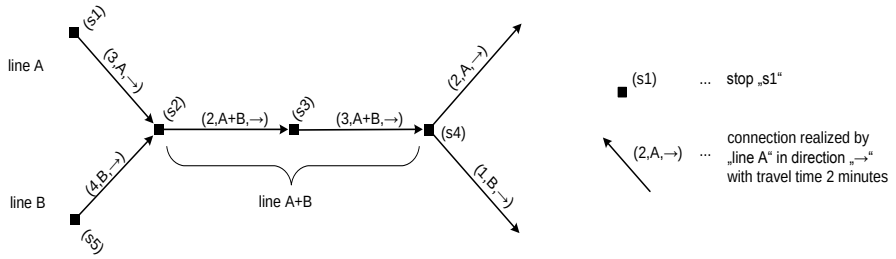

Fig. 2. Aggregation of the edges with the same travel time

Mapping the Path to the Original Network The vertices of the new graph directly correspond to the vertices of the original graph. The problem occurs in the case of edges, where the information about the original pseudoline used to travel between stops is lost due to creation of merge-line. To be able to determine the edge in the original graph, it is necessary to remember the identity of the original pseudoline that is used to travel through the aggregated edge[4]. When searching the new graph, the transfer detection should be changed as followed:

[4] There can be more such pseudolines; we will therefore remember a list of applicable pseudolines.

Getting in While getting in a pseudoline, it is necessary to detect which of the original pseudolines from the merge-line is just used. First, the departure time of the next service is found – in the aggregated timetable. This is the departure time of merge-line. Now the original pseudoline needs to be found. At this point it is necessary to view the original timetables to determine which original pseudolines are leaving at the found time[5].

Transfer between aggregated pseudolines If the identification of the merge-line on the arrival edge and on the leaving edge is equal, then this is not a transfer. In this case, the waiting time is not counted in.

Real transfer If the identification of the merge-line on the arrival edge and on the leaving edge are not equal, then this could be a transfer. This is not a real transfer if the pseudoline of arrival edge **is contained** in the list of original pseudolines of the merge-line on the leaving edge. In other case, this is a real transfer and the waiting time must be counted in.

Correctness The vertices in the new graph are equal to the vertices in the original one. Every aggregated edge represents the edge in the original graph with equal value. The changed transfer mechanism above ensures, that the transfers between merge-lines are equal to the transfers between original pseudolines. The pseudolines in the original graph can be labeled by the identification of merge-line into which it is aggregated. The path found in the reduced graph will be created by merge-lines corresponding the labels on pseudolines, which creates the path in the original graph.

	# of nodes	# of edges	memory usage	search time
before	1096	8473	429 721B	4s
after	1096	2927	1 182 226B	1.70s
decrement	0%	65%	-175%	58%

Table 1. Comparison of computation over original data on Prague Public Transport Network and after simple network compaction

3.2 Path Aggregation

The latter adjustment builds on the results of the first one. Based on a direct connection between the two stops longer stretch of several consecutive stops. The condition is that only one certain line runs in this sequence of stops, and no other line is connecting or leaving this sequence. Such a sequence of stops can be aggregated into a sort of “pipelines”.

[5] This situation is simpler then in the original graph. The departure time needs to be found only once – in the aggregated timetable. Then this time is searched in the original timetables for the match.

Pipeline is an aggregation of a number of services that ensures the same travel time between all the stops in the sequence. In order to guarantee this property, the order of services entering the pipeline must be the same like the order of the services leaving the pipeline. For trams or trolley-buses this property is guaranteed. The problem might occur on buses.

Node is a stop, where the passenger can take a relevant transfer to another service. The transfer is considered to be relevant, if the service can get the passenger to other stop that he was not before. The example of not relevant transfer is a transfer to the same line, but opposite direction. This typically gets the passenger to the stop where he was[6]. By excluding these loops the resulting path will always get shorter.

Starting with a graph where the first adjustment has been already made simplifies the initial situation. In the following, we assume the first adjustment is made. The second adjustment may be done in two phases.

The first phase Pipeline is the edge of the new graph. Vertices, which are outside the pipeline, will form the vertices of the new graph – nodes. Vertices, which are intside the pipeline must hold the following two conditions:

1. The vertex may not have more than two different adjacent vertices[7].
2. Edges that go into the vertex must also go outside. Corresponding input and output edges must bear the same identification of merge-line.

The second phase The nodes are connected by edges representing the pipelines: An edge representing a pipeline between two nodes corresponds to a sequence of vertices in the original graph. This sequence must be of the same merge-line and can not be interrupted by any other node what ensures that there will be no transfer inside the sequence. This sequence is replaced by the new single graph edge. The value of new edge is set to the sum of the values of edges in the original sequence. Merge-line of the new edge is the merge-line of the original edges.

Connection of the path search There will be a reduction of vertices in the new graph. This means that if the shortest path search starts in a vertex, which is not a node in the new graph, it is necessary to find the path to the nearest nodes. This path can be found in the original graph quite easily because of the properties of vertices within pipeline which means that the path can lead up to a maximum of two directions. After a very short search two peripheral nodes will be encountered. Both peripheral nodes will be taken as starting points for search in the new graph. For starting points the initial value of path length estimation will be set to the length of the path from the original vertex to the nearest node – to the starting point. Similarly, if the target is not node.

[6] The opposite direction of the link can get the passenger to the stop, where he was not yet; therefore the nodes should be chosen carefully.

[7] The vertices available in the opposite direction of the oriented edge are also considered to be adjacent here.

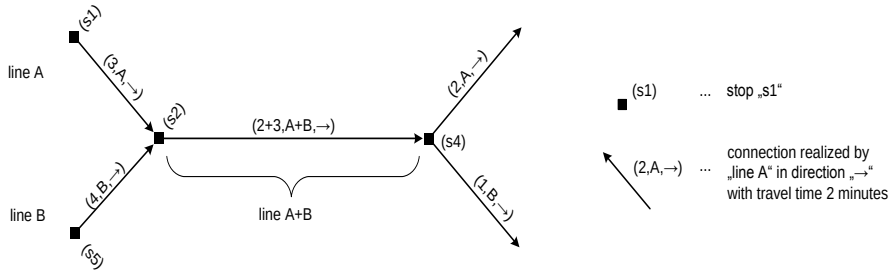

Fig. 3. Creation of pipeline between the nodes.

Mapping the path to the original network Since the latter adjustment is based on the first one, we need the same procedure used for detection of transfers as the first adjustment. The path found in this new graph is made up of nodes and pipelines.

To overcome the reduction of vertices to nodes we can return back to the vertices. To get detailed path in the 2nd level graph it is sufficient to search for path between vertices corresponidng to the nodes being neighbours in the 3rd level graph.

Finally, we need to add the initial segments into the resulting path which we used to get from initial vertex to the initial nodes.

Correctness The sequences of vertices and edges in the graph after first reduction were replaced by the single edge. The value of this edge is equal to the sum of values of the original edges. The mechanism of choice of the vertices inside the pipeline ensures, that the transfer in the vertex inside the pipeline is not relevant for searching the shortest path. Therefore the value of pipeline is equal to the value of the part of any shortest path leading through the sequence of vertices and edges creating the pipeline. The mechanism for detection of transfers is equal to the mechanism in previous reduction step.

	# of nodes	# of edges	memory usage	search time
before	1096	2927	1 182 226B	1.70 s
after	549	1987	1 225 810B	1.03 s
decrement	50%	32%	-4%	39%

Table 2. Comparison of computation over modified data on Prague Public Transport Network and after advanced network compaction

4 Prague Public Transport

As we have only data set the Prague public transport, the example is based on part of this data. The time necessary for the graph search on a sample mobile device (hardware details are mentioned below) is included in the tables in the text. For benchmarking purposes the terminating condition was excluded from the search algorithm. So the values in Tab. 2 are maximal – represent searching entire graph.

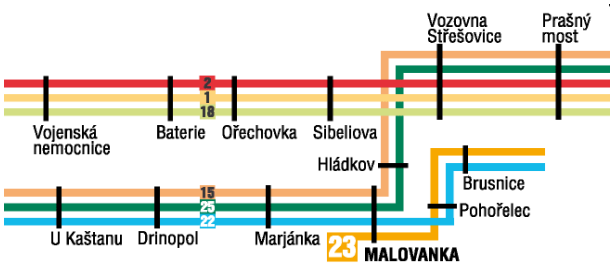

Fig. 4. Sample cutout of Prague public transport (taken from [6])

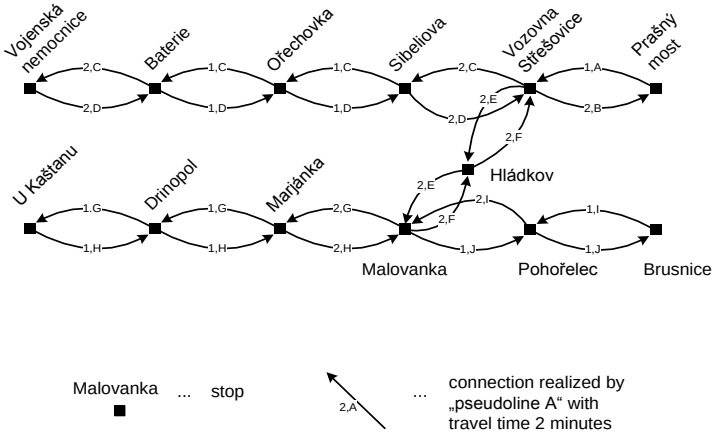

Fig. 5. Sample of Prague public transport after first adjustment.

In our sample case the number of vertices decreased more then two times and number of edges almost seven times. The density of individual lines crossing

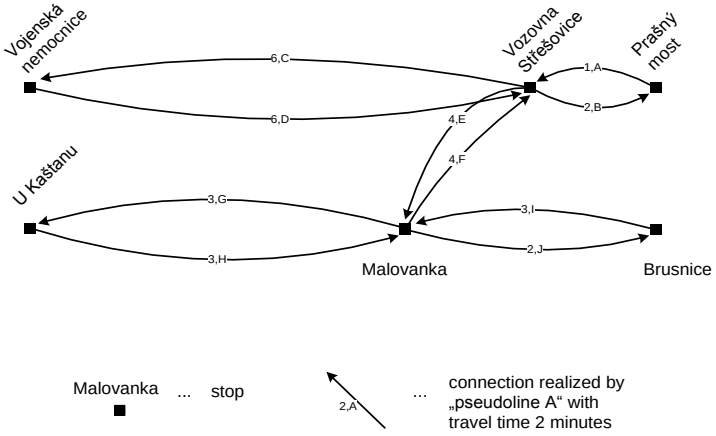

Fig. 6. Sample cutout of Prague public transport after both adjustments.

and irregular placement of street refuges prevent from such significant reduction in full-scale.

	# of nodes	# of edges	memory usage	search time
original graph	1096	8473	429 721B	4s
first adjustment	1096	2927	1 182 226B	1.70s
second adjustment	549	1987	1 225 810B	1.03s
decrement	50%	77%	-185%	74%

Table 3. Comparison of computation over original data on Prague Public Transport Network and after simple network compaction

5 More Opportunities for Graph Reduction

The advantage of the adjustments referred to in this paper is that they do not alter the substance of problem but they only reduce the size of input data. It does not prevent the application of other techniques for reducing the search complexity.

5.1 Highway Hierarchy

One of interesting processes, which could build on referred adjustments is “highway hierarchy”[7]. This method could be used in two ways. We could further reduce the public transport network from the second adjustment to achieve a

further acceleration. To bring a noticeable improvement, we must be able to find areas in the graph, with relatively dense traffic inside. These areas should be connected together in a relatively small number of vertices. This approach brings a problem, how to find areas compliant to the "highway hierarchy". In the case of urban public transport we consider finding of compliant areas to bring the significant improvement very difficult. The other approach is to link urban public transport and national or interstate transport networks together using "highway hierarchy".

6 Results

6.1 Memory Consumption

There are three levels of the graph – the original graph, graph after the first and after both adjustments. For the search itself we need to hold the two highest levels in memory. The initial segments are searched in the second level graph when the initial or target vertex is not a node. Otherwise we start directly in the third level. The path between nodes is searched in the third level graph. It is not necessary to hold the original graph in main memory. The size of the graph levels is decreasing. So the memory consumed by the graphs themselves would not exceed the double of original value.

The data of the original timetables are needed for the changed transfer detection in the modified graph. The aggregated timetables are needed to determine the waiting time. The resulting memory consumption depends on the representation of timetables. If the time tables are maintained only for the initial stop of the pseudoline, the number of merged time tables will depend on the number of pipeline. Each pipeline has its own merge-line and its own timetable.

In the current implementation all three levels of the graph are kept in the main memory. This reflects to the referred memory consumption. Our aim in future is to choose an appropriate representation of time tables and to minimize the memory needed to store the additional structures.

6.2 The Hardware

Our reference hardware is HTC X7500 having Intel XScale 624MHz processor and 128 megabytes RAM, 65 megabytes of free. Portable devices, on which the current implementation is mostly targeted, used to be equipped by secondary Flash-type memory. The writing to this kind of memory is usually several times slower then the reading. The advantage of the mentioned adjustments is that they do not require frequent writing into the secondary memory. Another specific feature is that the access to various locations in the memory is not as complicated as for example for hard drives. For this reason it is possible to hold the original graph in the secondary memory without slowdown noticeable to the user.

7 Conclusions

The methods introduced here speed up searching for optimal path between two given points in public transport network. They are intended to be used on mobile hardware where the original computation took several seconds what is for many users not acceptable. Currently the computation takes in the worst case only one second what is acceptable for most users. We are currently considering other options of improvement of the search algorithm.

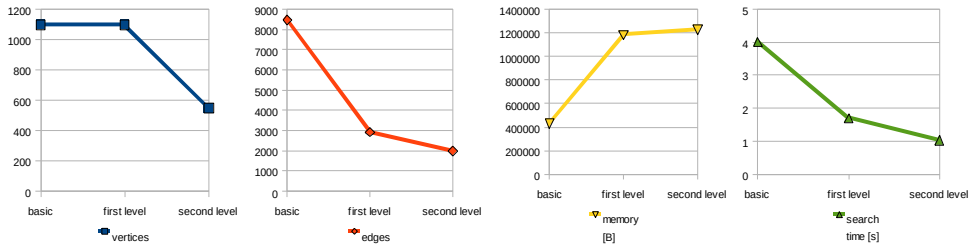

Fig. 7. Shift in the number of vertices, edges, memory consumption and computing speed.

References

1. Sanders, P., Schultes, D.: Engineering fast route planning algorithms. In Demetrescu, C., ed.: WEA. Volume 4525 of Lecture Notes in Computer Science, Springer (2007) 23–36
2. Weihe, K.: Covering trains by stations or the power of data reduction. In Battiti, R., Bertossi, A.A., eds.: Proceedings of "Algorithms and Experiments" (ALEX98). (1998) 1–8
3. Liebers, A., Weihe, K.: Recognizing bundles in time table graphs - a structural approach. In Näher, S., Wagner, D., eds.: Algorithm Engineering. Volume 1982 of Lecture Notes in Computer Science., Springer (2000) 87–98
4. Demetrescu, C., Italiano, G.F.: Experimental analysis of dynamic all pairs shortest path algorithms. ACM Transactions on Algorithms **2**(4) (2006) 578–601
5. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1** (1959) 269–271
6. Dopravní podnik hlavního města Prahy: (Užitečná dopravní schémata) `http://www.dpp.cz/uzitecna-dopravni-schemata/`.
7. Delling, D., Sanders, P., Schultes, D., Wagner, D.: Highway Hierarchies Star. Technical report, ARRIVAL Project (2006) work presented at 9th DIMACS Challenge on Shortest Paths.