Acta Cybernetica **23** (2018) 773–782.

Regional Multicriteria and Multimodal Route Planning System for Public Transportation: A Case Study*

József Békési[a]

Abstract

Nowadays, the use of computer-based route planners is popular among private and public transportation passengers. A large range of websites and GPS navigation devices provide such services for their users. Here, we present an algorithm used by a route planning system which operates on a complete public transport network of two regions from two countries, namely Hungary and Serbia. The algorithm can handle the pedestrian traffic between stops not too far from each other. It can take into account individual user preferences like walking distances and modes of transport. The graph representing the transport network was very large, but with the help of some speed-up techniques, we managed to create an effective search algorithm that is able to handle user requirements.

Keywords: public transportation, route planning, algorithms

1 Introduction

Nowadays, the use of computer-based route planners is widespread among passengers. Individual passengers have a large choice of websites and GPS navigation devices. Generally speaking, with these kind of systems the available road network is modeled by a graph. The vertices of the graph represent the points of contact of the roads, while the edges represent the road sections connecting the points. If we assign the length of the road section to the edges, we get a weighted graph. We can use the well-known Dijkstra algorithm to calculate the shortest distance between two points. The efficiency of the Dijkstra algorithm is quite good, but the size of the graphs describing real-world road networks can be very large, especially in the case of a larger geographical area like the road network of a continent. As passengers generally expect an almost immediate response to their searches, even the Dijkstra

*This research work was supported by the EU-funded Hungarian grant EFOP-3.6.2-16-2017-00015.

[a]Department of Applied Informatics, Gyula Juhász Faculty of Education, University of Szeged, E-mail: bekesi@jgypk.szte.hu

DOI: 10.14232/actacyb.23.3.2018.3

algorithm with polynomial running time is not good enough for large graphs. Over the last two decades, many potential speed-ups have been investigated to address the problem. More details about this can be found in [6].

Routing services are available not only for private passengers, but for passengers travelling by public transportation as well. In this case, usually the road network does not need to be described by the graph. The reason for this is that the route is pre-determined, or the journey is often not on a conventional road network but on a bound track (e.g. rail) or in the air, or even on water. The nodes of public transport routing graphs generally represent the vehicle stops, and the edges provide travel possibilities between stops. The weight of the edges may represent the travel time, but as different users may look at travel from different aspects, other models should be considered. Public transport networks may generally be larger than those of road networks, given that travel is time-dependent, and handling this is also required in the model. Therefore, in this case, the opportunities for speeding-up searches are particularly important, and they have also been widely investigated. Some of the speed-up options available on road graphs can be used here, but not those that use the special features of road networks. For more details on models and speed-up options, see [4, 11].

In practice, it is usual for public transportation companies to provide route planners for their own service area. This usually includes the services of the company or other companies closely related to that city, region or country. In most cases, however, these services provide only route planning for one kind of journey; for example, a route planner of a rail company can be searched for rail routes, while route planners of bus companies can be used for bus routes. In the case of local transport, it is common that there are search engines that cover different modes of transport in a given city, such as buses, trams and metro, trains, but they are usually not intended for long-distance transport. So, for instance, if a passenger wants to get from a point in a city to a point in another city, using local and then several long-distance modes of transport, he or she usually will not find a search engine that offers such a travel option, not even in a country or in a region. And the graphs of road networks can handle the same problem for larger areas such as continents. This suggests that finding a route to public transport is a more difficult issue than planning private trips.

In this paper, we present a route planning system and its search algorithm, where the latter operates on a complete public transport network of two regions from two countries, namely Hungary and Serbia. The databases on which the model is based include long-distance trains, buses and complete local transport of the major cities. The databases are based on timetables taken from service providers operating in the regions. The system can also model the pedestrian traffic between stops not too far from each other. It can take into account individual user preferences, like walking distances, modes of transport, and properties of the objective function. The graph representing the transport network was very large, but with the help of some speed-up techniques, we managed to create an effective search algorithm that is able to handle user requirements.

The algorithm was developed under the EU-funded Hungary-Serbia Interreg-

IPA CBC Program as part of a complex route planning application. This study presents the most common methods used in the search algorithm, as well as the experiences and results obtained during the practical running.

2 Preliminaries and literature review

Next, we will summarize the key results of public transport route algorithms described in the literature. We will mainly refer to the technology used by the method we use, and in the following discuss the topic in more detail [6, 12].

The simplest modelling of public transport networks can be achieved using the so-called Station Graph [15]. In this case, the graph's nodes represent the stops and edges only exist between two stops if one of the stops can be reached directly from the other. Generally speaking, the Station Graph is a relatively small graph, since each stop has exactly one node and there is at most one edge between any two stops. However, this graph contains limited information as it does not represent the time of the routes at all. Hence it is not suitable for precise route planning, and it can only provide a poor estimate for the length of the trip or for the minimum required transfers.

The Time-Dependent model may be regarded as an extension of the Station Graph [5], in which the edges may have multiple weights. During a route search, the current weight is calculated using the given time, taking into account the bus lines departing from the stop. In this case, the size of the graph does not increase, but the determination of the weight requires more complex calculations, which may slow down the search. The model has often been studied on railway networks and has been developed, for instance, for the correct modelling of transfers [13].

The most common model used is the Time-Expanded model [14]. Here, the starting and arrival times of the vehicles are represented by special nodes. The nodes belonging to a station can be sorted by time and the waiting can be represented by edges. Trips between the different stations can be expressed by edges between the appropriate departure and arrival times. Therefore the model can handle transfers, and pedestrian traffic between two specific stops can also be modelled. Initially, due to the large size of the graph, searches in this model were not sufficiently fast, but due to technological developments and improvements, we can now consider it a competitive method [6].

While the aim of search engines is normally to find the shortest route between departures and arrivals, this is not always the case for public transport networks. Passengers may wiew things from different aspects; some may want to minimize the number of passes against the earliest arrival, while others may want to keep their travel costs as low as possible. What is more, we usually choose one mode of transport on the route; for example, like that for a motorist or pedestrian. With public transport, someone may want to use a variety of vehicles or does not even have the opportunity to reach the goal otherwise; for example, he or she needs to combine buses and railways, and has to reach the stop on foot. Hence special models have also been developed for public transport networks. This is why besides

the classical earliest arrival problem, [14] multiobjective optimization methods were applied [11].

The research work of recent years has focused mainly on developing complex multiobjective, multimodal route-planning algorithms [3, 9], which include appropriate speed-up techniques [8] and they can be applied in practice [1].

3 Modeling

All three types of graphs described above were used to model our transport network. Only the Station Graph and the Time-Dependent Ggraph built on it were stored permanently in the memory. The number of edges of the Time-Dependent Graph was already quite large as there were seven different timetable versions in use during that period. This meant that searches for different days and periods were subject to different schedules. For example, the Sunday schedule was different from the weekday one. In the Time-Dependent Graph there were different edges for the different lines, and the departure and arrival times of the lines were stored in separate structures. It also included possible walking routes. And the Station graph and the Time Dependent graph did not have any direct search, but only had roles in the preprocessing of searches. In the actual search, the Dijkstra algorithm was implemented on a Time Expanded Graph. The current Time Expanded Graph was not stored permanently in memory as the many different timetable versions dynamically changed it. The generation of the current graph was always related to the actual query.

Table 1: Sizes of the graphs

Graph type	Number of nodes	Number of edges
Station	12014	416163
Time Dependent	12014	6786662
Time Expanded	≈ 562000	≈ 4616000

Table 1 lists the sizes of different graphs.
Figure 1 depicts the modelling of a line with the Station Graph.

Figure 2 shows a part of the Time Expanded Graph. The horizontally positioned nodes are part of a stop's timeline, and the pink nodes represent the departure times, the green ones represent the arrival times. The orange edges are the waiting edges of the timeline. The black edges model the lines between the stops, while the red edges show the walking options. The departure timeline is in the upper line, the arrival is in the lowest one. Thickly marked paths indicate the travel possibilities chosen by the system.

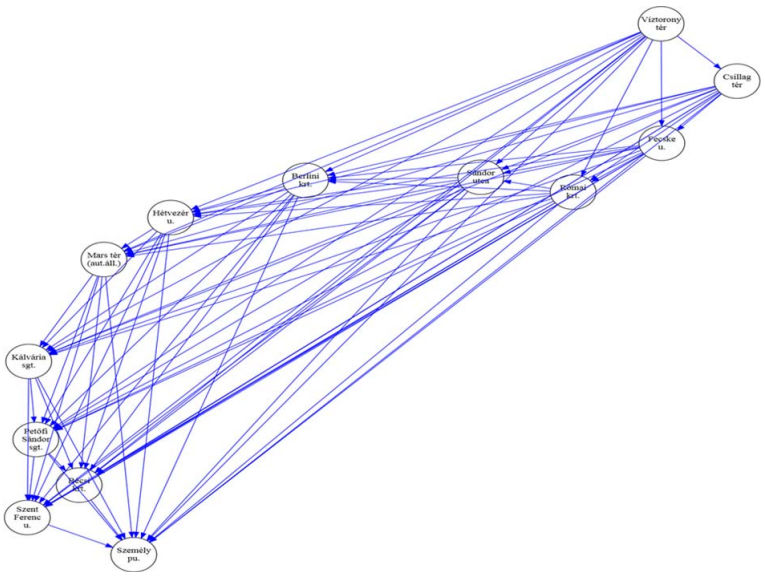

Figure 1: The Station Graph of a line

4 Search algorithm

After producing the Time Expanded Graph and weighing the edges appropriately, with the help of the Dijsktra algorithm we can find the shortest path. In our case, we came up against a difficulty caused by the search dependence of the Time Expanded Graph. Its structure was influenced by the time specified in the search and by the modes of transport that the user wanted to use, or by other parameters such as the maximum allowed walking distance. Despite the large size of the graph, its generation was relatively fast, but after including the search time, in some cases we could not achieve the desired speed with this method. In addition, the structure of the entire graph resulted in excessive memory usage during a search, which was a real problem, because multiple clients wanted to use the services of the server simultaneously.

To overcome the problems listed above, we performed a preprocessing step before producing the Time Expanded Graph and the search. The aim was to determine the nodes which include some of the shortest paths corresponding to the user parameters and the objective function. The method is similar to the TRANSIT algorithms described in the literature [1, 2, 9]. In the end, the Time Expanded Graph contained far fewer nodes, and it led to a significant speed-up of the search.

The purpose of the preprocessing was to determine the nodes that lie on the shortest paths from the source to the target. Here, we used the Station Graph for the prepocessing part. Our aim was to determine all the nodes that lie on the maximum k-length paths between the source and destination. In this case, the

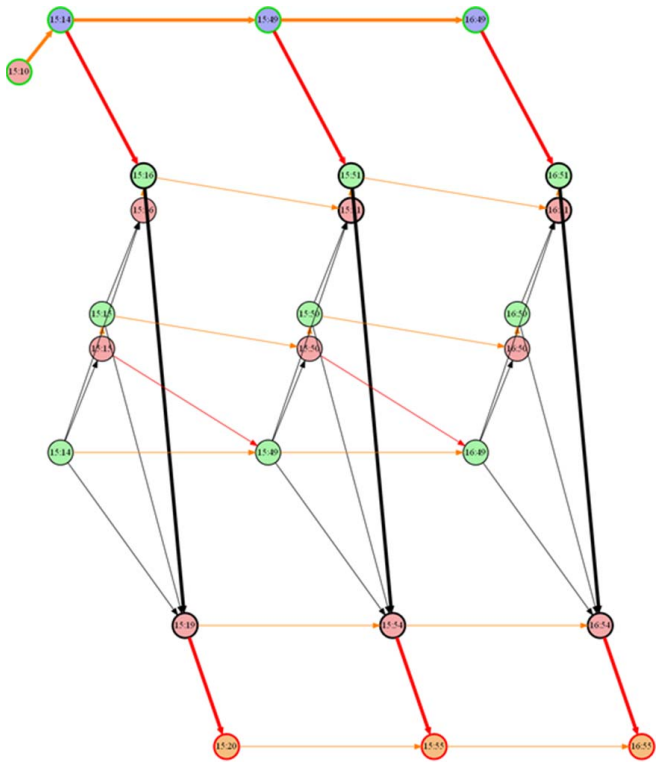

Figure 2: Part of the Time Expanded Graph

parameter k means the maximum number of transfers that can be made during the journey. Here, we used a modified version of the depth first search algorithm that did not take into account the nodes farther from the source than the desired length. Given that in some cases the execution of the procedure might take longer, the value of the parameter k and the execution time were also limited in the search. Experience has shown that these heuristics worked well in practice so, in almost every case, the Time Expanded Graph built on these nodes contained the shortest path of the entire graph. For a more detailed analysis of this, see Section 6.

5 The objective function

In general, different objective functions are used for public transport route planning. One of the most common is the earliest arrival, but it does not always fully meet the user's preferences. Finding different Pareto optima is also a common problem. We used an objective function that included a weighted sum of different goals. And the weights were determined based on information provided by the users, using preset patterns.

The weight of a travel edge is the following: $t_r w_r + f_r$, where t_r is the time of travel, w_r is the actual weight of the journeys, and f_r is the additive factor for the travel cost.

The weight of a waiting edge is the following: $t_i w_i$, where t_i is the waiting time, and w_i is the actual weight of the waiting.

The weight of a pedestrian edge is the following: $t_a w_a + f_a$, where t_a is the time of walking, w_a is the actual weight of the walking, and f_a is the additive factor for pedestrian cost.

In the search, there were three options available to the users regarding the search query. These three options are the fastest route, the minimum number of connections and the minimum walking distance. And the parameters of the weights for each goal are given in Table 2.

Table 2: Parameters of the weights

Options	w_r	f_r	w_i	w_a	f_a
Fastest	1	30	0.8	1.5	30
Less transfer	1	1000	0.8	1.5	10
Less walking	1	30	0.8	30	1000

6 Results and analysis

We investigated the proportion of the investigated instances such that the reduced graph contained all nodes of the shortest path. This ensured that the search on the reduced Time Expanded Graph would give an optimal solution. In the preprocessing heuristics we modified two parameters. One was the previously mentioned k parameter, while the other was the search time. For the possible values of k, we selected s+1 and s+2, where s is the shortest path between two points in the Station Graph.

Here, we used two types of test queries. They both contained 1000 queries, one of which concerned questions regarding local traffic (T1) and the other concerned long-distance traffic (T2). In the table below, we summarize how efficient the preprocessing heuristics was for the two types of test data, for different values of k. In the table out of the 1000 questions we see how many times the reduced-size Time Expanded Graph gave a solution that differed from the optimum, which we calculated using the complete Time Expanded Graph.

Tables 4 and 5 contain the average and maximum search times in milliseconds on the above-mentioned query lists for the reduced and for the complete Time Expanded Graphs. The notations used for the columns are the following:
RB: Reduced graph build time
RS: Reduced graph search time
RC (=RB+RS): Reduced graph complete search time

Table 3: Non-optimality statistics of the prepocessing heuristics

Input type	$k = s + 1$	$k = s + 2$
T1	13	0
T2	54	11

FB: Full graph build time
FS: Full graph search time
FC (=FB+FS): Full graph complete search time

Table 4: Average running times in milliseconds

	RB	RS	RC	FB	FS	FC
$T1, k = s + 1$	308	18	327	5327	166	5493
$T1, k = s + 2$	1394	406	1801			
$T2, k = s + 1$	256	55	312	5427	1137	6564
$T2, k = s + 2$	1186	316	1502			

Table 5: Maximal running times in milliseconds

	RB	RS	RC	FB	FS	FC
$T1, k = s + 1$	1478	446	1529	5744	4130	9502
$T1, k = s + 2$	4959	2458	5453			
$T2, k = s + 1$	2491	788	3279	6290	10216	15650
$T2, k = s + 2$	3119	3938	6469			

7 Conclusions

We presented an algorithm used by a route planning system for a complete public transport network of two regions from Hungary and Serbia. The graph representing the transport network was very large, but with the help of some speed-up techniques, we managed to create an effective search algorithm that is able to handle user requirements.

8 Acknowledgments

This study was carried out in cooperation with The Public City Transport Enterprise of Novi Sad, Department of Applied Informatics, University of Szeged, University of Novi Sad, DAKK Ltd., Szeged, Newline Ltd., Szeged and TIAC Ltd., Novi Sad.

References

[1] L. Antsfeld, T. Walsh, Finding Optimal Paths in Multi-modal Public Transportation Networks using Hub Nodes and TRANSIT algorithm, In.: L. Frommberger,K. Schill, B. Scholz-Reiter (eds.), Proceedings of the 3rd Workshop on Artifical Intelligence and Logistics, Montpellier, France, 7-13, 2012.

[2] H. Bast, S. Funke, D. Matijevic. Transit ultrafast shortest-path queries with linear-time preprocessing. In 9th DIMACS Implementation Challenge, 2006.

[3] H. Bast, E. Carlsson, A. Eigenwillig, R. Geisberger, C. Harrelson, V. Raychev, F. Viger, Fast routing in very large public transportation networks using transfer patterns, Proceedings of the 18th Annual European Symposium on Algorithms (ESA'10), Lecture Notes in Computer Science, Vol. 6346, Springer, 290-301, 2010.

[4] H. Bast et al., Route Planning in Transportation Networks. In: L. Kliemann, P. Sanders (eds) Algorithm Engineering. Lecture Notes in Computer Science, Vol. 9220. Springer, Cham, 2016.

[5] G. Brodal, R. Jacob. Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries. Electronic Notes in Theoretical Computer Science, Vol. 92, 3-15, 2004.

[6] D. Delling, Engineering and augmenting route planning algorithms, Ph.D. thesis, Universität Karlsruhe, Fakultät für Informatik, 2009.

[7] D. Delling, T. Pajor, D. Wagner, Engineering time-expanded graphs for faster timetable information, Robust and Online Large-Scale Optimization, Lecture Notes in Computer Science, Vol. 5868, Springer, 182-206, 2009.

[8] D. Delling, T. Pajor, D. Wagner, Accelerating multimodal route planning by access-nodes, Proceedings of the 17th Annual European Symposium on Algorithms (ESA'09), Lecture Notes in Computer Science, Vol. 5757, Springer, 587-598, 2009.

[9] J. Koszelew, Two Methods of Quasi-Optimal Routes Generation in Public Transportation Network, International Conference on Computer Information Systems and Industrial Management Applications,231-236, 2008,

[10] M. Müller-Hannemann, K. Weihe, Pareto shortest paths are often feasible in practice, Proceedings of the 5th International Workshop on Algorithm Engineering (WAE'01), Lecture Notes in Computer Science, Vol. 2141, Springer, 185-197, 2001

[11] M. Müller-Hannemann, F. Schulz, D. Wagner, C. Zaroliagis, Timetable Information: Models and Algorithms. In: F. Geraets, L. Kroon, A. Schoebel, D. Wagner, C.D. Zaroliagis (eds) Algorithmic Methods for Railway Optimization. Lecture Notes in Computer Science, Vol. 4359. Springer, Berlin, Heidelberg, 2007

[12] T. Pajor, Algorithm Engineering for Realistic Journey Planning in Transportation Networks, Ph.D.thesis, Universität Karlsruhe, Fakultät für Informatik, 2013.

[13] E. Pyrga, F. Schulz, D. Wagner, C. Zaroliagis, Efficient models for timetable information in public transportation systems, *ACM Journal of Experimental Algorithmics* Vol. 12 No. 2.4, 1-39., 2008

[14] F. Schulz, D. Wagner, K. Weihe, Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport, *ACM Journal of Experimental Algorithmics*, Vol. 5, No. 12, 2000.

[15] F. Schulz, D. Wagner, C. Zaroliagis, Using multi-level graphs for timetable information in railway systems, Proceedings of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02), Lecture Notes in Computer Science, Vol. 2409, Springer, 43-59, 2002.

Received 4th April 2018