The Societal Impact of Algorithms in Transport Optimization[1]

Christos D. Zaroliagis

Dept of Computer Eng & Informatics, University of Patras, 26500 Patras, Greece
R.A.Computer Technology Institute, Patras University Campus, 26500 Patras, Greece
Email: zaro@ceid.upatras.gr

Abstract

We review some recent advances for solving core algorithmic problems encountered in public transportation systems. We show that efficient algorithms can make a great difference both in efficiency and in optimality, thus contributing significantly to improving the quality and service-efficiency of public transportation systems.

1. Introduction

Mobility of passengers and goods has been increasing steadily over the past years, and it is likely that this trend will continue. Part of this growth in mobility is facilitated by public transportation systems, which are of great importance for economic growth and quality of life (especially within Europe). To achieve these vital socio-economic goals, public transportation systems have to improve the quality and efficiency of their management as well as the services they provide. This gives rise to a wealth of optimization problems, whose common characteristic is their high complexity and their sheer size.

One of the key problems in public transportation systems is the management of timetable information. In particular, the primary task is how to organize the information so that subsequent queries asking for optimal itineraries (best routes) can be efficiently answered. The main challenge is to cope with a typical situation in public transportation systems, where a vast number of on-line queries have to be processed as fast as possible. Such a query-intensive scenario is encountered, for instance, in (European) public railway transport, where a central server is directly accessible to any customer either through terminals in train stations or through a web interface, and has to answer a potentially infinite number of customer queries asking for their optimal itineraries (for example, the server of the German railways receives about 100 queries per second). Note that a similar situation is encountered in other

[1] This work was partially supported by the FET Unit of EC (IST priority – 6th FP), under contract no. FP6-021235-2 (ARRIVAL).

real-time application systems (e.g., car-traffic information systems, database queries, web searching, etc). The main goal in all such applications is to reduce the average response time for a query.

Efficient answering of timetable information is related to two core algorithmic problems: how to model timetable information so that itinerary queries can be answered fast and how to answer the query efficiently *and* optimally (i.e., correctly).

In this work, we review some recent advances regarding these two core algorithmic problems and show that efficient algorithms can make a great difference both in efficiency and in optimality. More specifically, we consider two approaches that model timetable information in public transportation systems as shortest path problems in weighted graphs. In the *time-expanded* approach every event at a station, e.g., the departure of a train, is modelled as a node in the graph, while in the *time-dependent* approach the graph contains only one node per station. Both approaches had been considered for a simplified version of the most frequently encountered timetable problem: the *earliest arrival problem*, where the goal is to find a train connection from a departure station *A* to an arrival station *B* that departs from *A* later than a given departure time and arrives at *B* as early as possible. So far, there were only theoretical arguments in favor of the time-dependent approach. In [Pyrga et al (2007)], the first extensive experimental comparison of the two approaches was carried out, along with new (theoretical and practical) extensions of them towards realistic modelling. In addition, several new, provably correct, heuristics are introduced to speedup query time. The evaluation was conducted on real-world data sets, and the experiments showed dramatic improvements in query time.

The most commonly used approach for answering shortest path queries employs Dijkstra's algorithm and/or variants of it. Consequently, the main challenge is how to reduce the algorithm's *search-space* (number of vertices visited), as this would immediately yield a better query time. In [Wagner et al. (2005)], the full exploitation of geometry-based algorithms was investigated using both street and railway networks. In that paper, it is shown that the search space of Dijkstra's algorithm can be significantly reduced (to 5% - 10% of the initial graph size) by extracting geometric information from a given layout of the graph and by encapsulating pre-computed shortest path information in resulted geometric objects, called *containers*. Moreover, the dynamic case of the problem was investigated, where edge costs are subject to change and the geometric containers have to be updated.

It is worth mentioning that the current commercial systems, e.g., the HAFAS system [HAFAS] used by the German and most European railways, provide itineraries that may not be optimal (given some optimization criterion). The main reason for such non-optimal itineraries is that the algorithms behind the systems employ heuristic methods to reduce the search space (in order to achieve an acceptable response time) that do not always guarantee optimal solutions.

2. Itinerary Problems and Timetable Information Modelling

A *timetable* consists of data concerning stations (or bus stops, ports, etc), trains (or busses, ferries, etc), connecting stations, departure and arrival times of trains at stations, and traffic days. A *timetable information problem*, or an *itinerary query*, defines a set of valid connections, and an optimization criterion (or criteria) on that set of connections. The problem is to find the optimal connection (or a set of optimal connections) with respect to the specific criterion or criteria. Typically, queries come in very large sequences and have to be answered on-line (i.e., in real-time). The two most important criteria are the earliest arrival and the minimum number of transfers.

2.1 Itinerary Problems

The most frequently encountered query is the so-called *earliest arrival problem*. In this problem, we are given a tuple (A, B, t_0) denoting a departure station *A*, an arrival station *B*, and a departure time t_0. Connections are *valid* if they do not depart before the given departure time t_0, and the optimization criterion is to minimize the difference between the arrival time and the given departure time. Additionally, one may ask among all connections that are solutions to such a query for the connection that departs as late as possible (latest departure problem).

Another important query concerns the so-called *minimum number of transfers problem*. In this problem, we are given a departure station *A* and an arrival station *B*. All connections from *A* to *B* are valid and the optimization criterion is to minimize the number of train transfers.

One can also consider bicriteria or Pareto-optimal problems with the earliest arrival (EA) and the minimum number of transfers (MNT) as the two criteria.

2.2 Basic Timetable Information Modelling

There are two main approaches for modelling timetable information: the time-expanded model and the time-dependent one.

The *time-expanded model* consists of the directed time-expanded graph, which is constructed as follows [Schulz et al (2000)]. There is a node for every time event (departure or arrival) at a station, and there are two types of edges. Let (Z, S_1, S_2, t_d, t_a) denote an elementary connection interpreted as train *Z* leaves station S_1 at time t_d, and the next stop of train *Z* is station S_2 at time t_a. For every such elementary connection in the timetable, there is a *train-edge* in the graph connecting a departure node, belonging to station S_1 and associated with time t_d, with an arrival node, belonging to station S_2 and associated with time t_a. In other

words, the endpoints of the train-edges induce the set of nodes of the graph. For each station *S*, all nodes belonging to *S* are ordered according to their time values. Let $v_1,\ldots,v_k$ be the nodes of *S* in that order. Then, there is a set of *stay-edges* $(v_i,v_{i+1}),1\le i\le k-1,$ and (v_k,v_1) connecting the time events within a station and representing waiting within that station. The length of an edge (*u*,*v*) is t_v-t_u (for edges over midnight the length is $1440+t_v-t_u$, respectively), where t_u and t_v are the time values associated with *u* and *v*, respectively.

A shortest path in the time-expanded digraph from the first departure node *s* at the departure station *A* with departure time later than or equal to the given start time t_0 to one of the arrival nodes of the destination station *B* constitutes a solution to the earliest arrival problem in the time-expanded model. The actual path can be found by Dijkstra's algorithm [Dijkstra (1959)].

The *time-dependent model* is also based on a digraph, called time-dependent graph [Brodal and Jacob (2003)]. In this graph there is only one node per station, and there is an edge *e* from station *A* to station *B* if there is an elementary connection from *A* to *B*. The length of an edge $e=(v,w)$ depends on the time at which this particular edge will be used during the algorithm. In other words, if *T* is a set denoting time, then the length of an edge (*v*,*w*) is given by $f_{(v,w)}(t)-t$, where *t* is the departure time at *v*, $f_{(v,w)}:T\to T$ is a function such that $f_{(v,w)}(t)=t'$, and $t'\ge t$ is the earliest possible arrival time at *w*. The time-dependent model is based on the assumption that overtaking of trains on an edge is not allowed, i.e., for any two given stations *A* and *B*, there are no two trains leaving *A* and arriving to *B* such that the train that leaves *A* second arrives first at *B*. A modification of Dijkstra's algorithm can be used to solve the earliest arrival problem in the time-dependent model. Let *D* denote the departure station and t_0 the earliest departure time. The differences, w.r.t. Dijkstra's algorithm, are: set the distance label of the starting node corresponding to the departure station *D* to t_0 (and not to 0), and compute the edge lengths “on-the-fly”. Since Dijkstra's algorithm is a label-setting shortest-path algorithm, whenever an edge $e=(A,B)$ is considered the distance label $\delta(A)$ of node *A* is optimal. In the time-dependent model, $\delta(A)$ denotes the earliest arrival time at station *A*. In other words, we indeed know the earliest arrival time at station *A* whenever the edge $e=(A,B)$ is considered, and therefore we know at that stage of the algorithm which train has to be taken to reach station *B* via *A* as early as possible: the first train that departs later than or equal to the earliest arrival time at *A*. The particular connection (among the possibly many between *A* and *B*) can be easily found by binary search if the elementary connections are maintained in a sorted array.

3. Realistic Timetable Information Modelling

The modelling mentioned in Section 2 makes two assumptions: (i) transfer time between stations takes zero time; and (ii) all trains operate daily (i.e., the timetable is identical for every day). Clearly, both assumptions are not realistic.

In [Pyrga et al (2007)], extensions to the aforementioned time-expanded and time-dependent modes have been proposed to take into account the non-negligible transfer times between trains at stations as well as to incorporate the different traffic days. In Section 3.1, we present the new models and discuss the solution of the earliest arrival problem. In Section 3.2, we discuss the solution to other problems.

3.1 Realistic Models and the Earliest Arrival Problem

To incorporate transfer times in the time-expanded model, [Pyrga et al (2007)] introduced the *realistic time-expanded graph*, which is constructed as follows. Starting from the original time-expanded graph, for each station, a copy of all departure and arrival nodes in the station is maintained which we call *transfer-nodes*. The stay-edges are now introduced between the transfer-nodes. For every arrival node there are two additional outgoing edges: one edge to the departure of the same train, and a second edge to the transfer-node with time value greater than or equal to the sum of the time of the arrival node and the minimum time needed to change trains at the given station. If the earliest arrival problem is to be solved, the edge lengths are defined as in the definition of the original model.

To incorporate transfer times in the time-dependent model, [Pyrga et al (2007)] extended the original model to include information on train routes, thus introducing the so-called *train-route graph*. Assume that we are given a set of train routes and their respective time schedules. The train-route graph is constructed as follows. We say that stations $S_0, S_1, \ldots, S_{k-1}$, $k > 0$, form a *train route* if there is some train starting its journey from S_0 and visiting consecutively $S_1, \ldots, S_{k-1}$ in turn. If there are more than one trains following the same schedule (with respect to the order in which they visit the above nodes), then we say that they all belong to the same train route. The node set of the train-route graph consists of the station nodes Σ representing the stations, and for each station *S* of one additional route-node per route that passes through the station *S*. There are three types of edges: (i) edges from each station-node to the route-nodes belonging to the same station that model the boarding of a train belonging to the specific route; (ii) edges from each route-node to the station-node that model the getting-off a train at that station; (iii) for each train route $S_0, S_1, \ldots, S_{k-q}$ edges that connect the corresponding route-nodes that model the actual train trips. To solve the earliest arrival problem with transfer times, edge lengths are defined as follows. The edges modelling boarding a train at a station *S* are

assigned the transfer time g_s, the edges modelling getting-off a train are assigned zero length, and the edges representing the train routes have time-dependent lengths as in the basic modelling. Given the query to solve, all internal edges are assigned zero length, and the modified version of Dijkstra's algorithm (see Section 2) is applied.

To take into account the traffic days in the time-expanded approach, edges representing elementary connections of trains that are not valid at a particular day can be simply ignored during Dijkstra's algorithm. Nevertheless, the algorithm has to be slightly modified because it may happen that an optimal connection stays more than a day at a station, and such connections would not be found otherwise (see [Pyrga et al (2007)] for the details).

In the time-dependent approach, the traffic days are considered in the computation of the time-dependent edge lengths.

3.2 Other Problems

The realistic time-expanded graph as well as the train-route graph can be used to solve a *minimum number of transfers* query with a similar method [Pyrga et al (2007)]: edges that model transfers are assigned a length of one, and all the other edges are assigned length zero. In the time-expanded case all incoming edges of transfer nodes have length one, whereas in the time-dependent case the edges that represent getting-off a train, except those belonging to the departure station, are assigned length one, and all other edges have length zero. Note that the edge lengths in the time-dependent train-route graph are all static. A shortest path in one of the graphs from a node belonging to (respectively representing) the departure station to a node belonging to (respectively representing) the arrival station provides a solution to the minimum number of transfers problem.

Determining a connection optimized for the *latest departure problem* combined with the earliest arrival can be done in the time-expanded case by introducing the latest departure as second criterion and determining the lexicographically first solution. In the time-dependent model the standard approach is to carry out a backward search from the destination station to the arrival station once the earliest arrival at the destination station is known.

Appropriate (non-trivial) extensions of the aforementioned algorithms can provide solutions to *bicriteria optimization problems*. In particular, the following bicriteria optimization problems have been considered and solved in [Pyrga et al (2007)] with the earliest arrival (EA) and the minimum number of transfers (MNT) as the two criteria: (i) finding the so-called *Pareto-curve*, which is the set of all undominated Pareto-optimal paths (the set of feasible solutions where the attribute-vector of one solution is not dominated by the attribute-vector of another solution); (ii) finding the solution that minimizes one criterion while retaining the second below a given

threshold; (iii) finding the lexicographically first Pareto-optimal solution (e.g., find among all connections that minimize EA the one with the minimum number of transfers).

4. The Impact of Efficient Algorithms

The above discussion reveals that the core procedure underlying all algorithms for the problems considered is Dijkstra's algorithm. It turns out, however, that a straightforward application of the classical Dijkstra's algorithm (even with the most efficient priority queue) can be much slower than variations of the algorithm enhanced with speedup techniques. For instance, the study in [Schulz et al (2000)] showed that Dijkstra's algorithm can be 60 times slower.

In [Pyrga et al (2007)], an improvement of the *goal-directed search* speedup technique was developed that is applied to both models, along with a host of model-specific speedup heuristics. Experiments were performed on several real-world railway networks exhibiting dramatic speedups in query times. For instance, in a part of the German railway timetable comprising of 6700 stations, 500000 departures and arrivals, and 100000 elementary connections (that correspond to the long-distance railway traffic in Germany) the following query times were achieved for the problems considered.

Problem	Realistic Time-Expanded Time [ms]	Realistic Time-Dependent Time [ms]
EA	78	50
MNT	125	38
Lex-First (MNT,EA)	161	83
All Pareto optima	287	181

The above experimental results demonstrate that: (i) the time-dependent approach is better (this is mainly due to the smaller in size resulted graphs); (ii) the fundamental earliest arrival problem can be solved in just 50 milliseconds; and (iii) finding all Pareto optimal solutions (which can be, in principle, exponentially large) takes less than 1/5 of a second.

Further speedup techniques were considered in [Wagner et al (2005)]. In that paper, a very fundamental observation on shortest paths was used: an edge that is not the first edge on a shortest path to the target can be safely ignored in any shortest path computation to this target. The main idea of the approach in [Wagner et al (2005)] is as follows. Assume that during preprocessing a set of nodes $S(e)$ is computed, for each edge e, containing all nodes that can be reached by a shortest path starting with e. Subsequently, when Dijkstra's algorithm is executed, those edges e for which the

target is not in $S(e)$ are ignored. As storing all sets $S(e)$ would require $O(nm)$ space, this prohibitive space requirement is relaxed by storing instead a geometric object, called *container*, for each edge that contains *at least* the nodes in $S(e)$. The shortest path queries are then answered by Dijkstra's algorithm restricted to those edges for which the target node is inside their associated geometric container. Note that this method still leads to a correct result (optimal path), although it may increase the number of visited nodes to more than the strict minimum (i.e., the number of nodes in the shortest path). In order to generate the geometric containers, we use the given layout of the graph.

In [Wagner et al (2005)], 12 different types of geometric containers were used. It turned out that the simplest container (bounding box) gives the fastest implementation, since it reduces the search space (number of nodes visited) of Dijkstra's algorithm to only 5% to 10% of the initial graph size. Moreover, efficient algorithms for dynamically updating the shortest path information encapsulated in the containers are also given in [Wagner et al (2005)].

5. Conclusions and Future Research

We reviewed some recent advances regarding the efficient answering of timetable itinerary queries, which is related to two core algorithmic problems: how to model timetable information so that itinerary queries can be answered fast and how to answer the query efficiently *and* optimally (i.e., correctly). We presented two new realistic models and showed that efficient algorithms can make a great difference both in efficiency and in optimality.

Currently, a fair amount of research is carried out in order to solve efficiently various optimization problems in railways, since they constitute the most complex and largest in scale transportation setting. Railway optimization deals with planning and scheduling problems over several time horizons (e.g., planning the train lines, constructing the timetable, scheduling the crew, etc). One of the most notable efforts is the EC-funded project ARRIVAL (Algorithms for Robust and online Railway optimization: Improving the Validity and reliAbility of Large scale systems), which is concerned with two important and actually unexplored facets of planning that pose even harder optimization questions: *robust planning* and *online (real-time) planning*. These two, tightly coupled, facets constitute a proactive and a reactive approach, respectively, to deal with disruptions to the normal operation over a short or medium time horizon. Robust planning is concerned with the development of an apriori plan that allows the absorption of disruptions to the best possible extend. Online planning is concerned with real-time decision making when, typically unpredictable, disruptions in daily operations occur, and before the entire sequence of disruptions is known.

The main goal of ARRIVAL is to develop the necessary foundational algorithmic research in order to provide ingenious and sound answers to the fundamental efficiency and quality issues encapsulated in robust and online planning of complex, large-scale systems as those in railways. More details on this research effort can be found in [ARRIVAL (2006)].

References

ARRIVAL Project (2006), funded by the FET Unit of EC (priority IST, 6th Framework Programme) under contract no. FP6-021235-2, URL: http://arrival.cti.gr/.

Brodal G.S., and Jacob R. (2003), *Time-dependent networks as models to achieve fast exact time-table queries*, In Proc. 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways – ATMOS 2003, Electronic Notes in Theoretical Computer Science, Vol.92 (2004), Elsevier.

Dijkstra E.W. (1959), *A note on two problems in connexion with graphs*, Numerische Mathematik, Vol.1, pp. 269-271.

HAFAS, *A timetable information system by HaCon Ingenieurgesellschaft mbH*, Hannover, Germany. URL: http://www.hacon.de/hafas/.

Pyrga E., Schulz F., Wagner D., and Zaroliagis C. (2007), *Efficient Models for Timetable Information in Public Transportation Systems*, ACM Journal of Experimental Algorithmics, Vol.12, No.2.4, pp.1-39.

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

Wagner D., Willhalm T., and Zaroliagis C. (2005), *Geometric Containers for Efficient Shortest Path Computation*, ACM Journal of Experimental Algorithmics, Vol. 10, No.1.3, pp.1-30.