Journal of Discrete Algorithms ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Engineering graph-based models for dynamic timetable information systems[☆]

Alessio Cionini[a], Gianlorenzo D'Angelo[b], Mattia D'Emidio[b], Daniele Frigioni[a], Kalliopi Giannakopoulou[c,d], Andreas Paraskevopoulos[c,d], Christos Zaroliagis[c,d,*]

[a] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Via Vetoio, I-67100 L'Aquila, Italy
[b] Gran Sasso Science Institute (GSSI), Viale Francesco Crispi, I-67100 L'Aquila, Italy
[c] Department of Computer Engineering and Informatics, University of Patras, 26504 Patras, Greece
[d] Computer Technology Institute and Press "Diophantus", 26504 Patras, Greece

A R T I C L E I N F O

Article history:
Received 23 October 2015
Received in revised form 19 March 2017
Accepted 29 September 2017
Available online xxxx

Keywords:
Journey planning
Public transportation systems
Timetable models
Shortest paths
Experimental evaluation
Dynamic updates

A B S T R A C T

In the last years we have witnessed remarkable progress in providing efficient algorithmic solutions to the problem of computing best journeys (or routes) in schedule-based public transportation systems. We have now models to represent timetables that allow us to answer queries for optimal journeys in a few milliseconds, also at a very large scale. Such models can be classified into two types: those representing the timetable as an array, and those representing it as a graph. Array-based models have been shown to be very effective in terms of query time, while graph-based ones usually answer queries by computing shortest paths, and hence they are suitable to be combined with the speed-up techniques developed for road networks.

In this paper, we study the behavior of graph-based models in the prominent case of *dynamic* scenarios, i.e., when delays might occur to the original timetable. In particular, we make the following contributions. First, we consider the graph-based *reduced time-expanded model* and give a simplified and optimized routine for handling delays, and a re-engineered and fine-tuned query algorithm. Second, we propose a new graph-based model, namely the *dynamic timetable* model, natively tailored to efficiently incorporate dynamic updates, along with a query algorithm and a routine for handling delays. Third, we show how to adapt the ALT algorithm to such graph-based models. We have chosen this speed-up technique since it supports dynamic changes, and a careful implementation of it can significantly boost its performance. Finally, we provide an experimental study to assess the effectiveness of all proposed models and algorithms, and to compare them with the array-based state of the art solution for the dynamic case. We evaluate both new and existing approaches by implementing and testing them on real-world timetables subject to synthetic delays.

Our experimental results show that: (i) the dynamic timetable model is the best model for handling delays; (ii) graph-based models are competitive to array-based models with respect to query time in the dynamic case; (iii) the dynamic timetable model compares favorably with both the original and the reduced time-expanded model regarding space;

☆ Research partially supported by the Italian Ministry of University and Research under Grant 2012C4E3KT PRIN 2012 (AMANDA), as well as by EU FP7/2007–2013 under grant no. 288094 (eCOMPASS), no. 609026 (MOVESMART), and no. 621133 (HoPE). A preliminary version of this work appeared in [10].

* Corresponding author.

E-mail addresses: alessio.cionini@gmail.com (A. Cionini), gianlorenzo.dangelo@gssi.it (G. D'Angelo), mattia.demidio@gssi.infn.it (M. D'Emidio), daniele.frigioni@univaq.it (D. Frigioni), gianakok@ceid.upatras.gr (K. Giannakopoulou), paraskevop@ceid.upatras.gr (A. Paraskevopoulos), zaro@ceid.upatras.gr (C. Zaroliagis).

https://doi.org/10.1016/j.jda.2017.09.001

(iv) combining the graph-based models with speed-up techniques designed for road networks, such as ALT, is a very promising approach.

1. Introduction

Computing the *best route* in a *schedule-based public transportation system* (consisting, e.g., of trains, buses, trams, etc) is a problem that has been faced at least once by everybody who ever traveled. In particular, the so-called *journey planning* problem is defined as follows: given an input *timetable* associated to the transportation system, one wants to answer queries like: "What is the best route from some station A to some other station B if I want to depart at time t?".

Despite its simple formulation, the journey planning problem is much more challenging with respect, for instance, to the classic route planning problem in road networks. In fact, schedule-based transportation systems exhibit an inherent time-dependent component that requires more complex modeling assumptions in order to obtain meaningful results. For this reason, nowadays, public transportation companies have invested a lot of effort in developing on-line software, called *journey planners*, which are able to efficiently answer to such kind of queries and to provide best routes with respect to some desired metrics.

Depending on the considered metrics and modeling assumptions, the journey planning problem can be specialized into a plethora of optimization problems. A first set of problems can be defined by considering different objective functions. If, for instance, the best route (a.k.a. *optimal*) is the one that minimizes the traveling time required to complete the journey, then the problem is called the *earliest arrival time problem*. Another, yet simpler and less considered, variant of the problem is to find the best route that minimizes the number of times that a passenger has to move from one vehicle of the system to another, during the journey. This version of the problem is called the *minimum number of transfers problem*. Sometimes, these two optimization criteria might be considered in combination (*bi-criteria problem*), possibly alongside further criteria (like, e.g., monetary cost). In this case, the problem is generally referred as the *multi-criteria problem*.

Further specializations can be obtained according to the level of abstraction at which the problem has to be solved. If, for instance, one wants to model (and optimize) the time required by a passenger for moving from one vehicle to another one within a station (i.e., *transfer time*), then the problem is referred to as *realistic* [31] while, when such an issue is ignored, the problem is referred to as *ideal* or *basic*. In this paper, we focus only on the realistic journey planning problem. Furthermore, if the departure time is within a time interval rather than be a single value, then the problem is called *profile* or *range* problem. We refer to the very recent survey of Bast et al. [3] for a comprehensive overview on journey planning.

1.1. Related work

The great variety of models proposed in the literature to solve the mentioned variants of the journey planning problem can be broadly classified into two categories: those representing the timetable as an *array*, and those representing it as a *graph* (see e.g., [3]).

Two of the most successful (and effective) examples of the array-based model are the Connection Scan Algorithm (CSA) [20] and the Round-bAsed Public Transit Optimized Router (RAPTOR) [17]. In CSA all the *elementary connections* of a timetable are stored in a single array which is scanned only once per query. The acyclic nature of some timetables is exploited to solve the earliest arrival problem. In RAPTOR the timetable is stored as a set of arrays of trips and routes which are used by a dynamic programming algorithm to solve the bi-criteria problem. Recently, some faster approaches in terms of query time with respect to the two aforementioned methods, have been proposed in [18] and in [37].

The graph-based models, instead, store the timetable as a suitable graph and execute known adaptations of Dijkstra's shortest path algorithm to compute optimal routes. There exist two main approaches of this kind: the *time-expanded* and the *time-dependent* model [31]. The former model explicitly represents each time event (departure or arrival) in the timetable as a node. The arcs represent elementary connections between two events or waiting within stations, and their weights usually represent the time difference between the corresponding events. The latter model represents each station as a node and there is an arc between two nodes if there exists at least one elementary connection between the two stations represented by such nodes. The weight of an arc is time-dependent, i.e., it is a function that depends on the time at which a particular arc is scanned during the shortest path search. The time-expanded model produces a graph with a larger number of nodes and arcs with respect to the time-dependent model, and thus larger query times. A variant of the time-expanded model having a smaller number of nodes and arcs (the so called *reduced time-expanded model*) has been proposed in [31]. Graph-based models can in general be easily adapted to solve the multi-criteria problem as well as RAPTOR [3].

Experimental results have shown so far that the array-based approaches are faster in terms of query time than graph-based ones [3,19,20], as they are able to better exploit data locality and do not rely on priority queues. On the other hand, array-based approaches have been tested on metropolitan-size instances that span a time period of one day, while graph-based approaches can be used in periodic timetables (i.e., that span more than one day), and therefore the latter are more suitable to handle queries in large-scale networks. Moreover, during the last years, a great research effort has been devoted to devise many so-called *speed-up techniques*, which heuristically speed up Dijkstra's algorithm for shortest