Trip-Based Public Transit Routing

Sascha Witt

Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
sascha.witt@kit.edu

Abstract. We study the problem of computing all Pareto-optimal journeys in a public transit network regarding the two criteria of arrival time and number of transfers taken. We take a novel approach, focusing on trips and transfers between them, allowing fine-grained modeling. Our experiments on the metropolitan network of London show that the algorithm computes full 24-hour profiles in 70 ms after a preprocessing phase of 30 s, allowing fast queries in dynamic scenarios.

1 Introduction

Recent years have seen great advances in route planning on continent-sized road networks [2]. Unfortunately, adapting these algorithms to public transit networks is harder than expected [1]. On road networks, one is usually interested in the shortest path between two points, according to some criterion. On public transit networks, several variants of point-to-point queries exist. The simplest is the earliest arrival query, which takes a departure time as an additional input and returns a journey that arrives as early as possible. A natural extension is the multi-criteria problem of minimizing both arrival time and the number of transfers, resulting in a set of journeys. A profile query determines all optimal journeys departing during a given period of time.

In the past, these problems have been solved by modeling the timetable information as a graph and running Dijkstra’s algorithm or variants thereof on that graph. Traditional graph models include the time-expanded and the time-dependent model [14]. More recently, algorithms such as RAPTOR [10] and Connection Scan [11] have eschewed the use of graphs (and priority queues) in favor of working directly on the timetable.

In this work, we present a new algorithm that uses trips (vehicles) and the transfers between them as its fundamental building blocks. Unlike existing algorithms, it does not assign labels to stops. Instead, trips are labeled with the stops at which they are boarded. Then, a precomputed list of transfers to other trips is scanned and newly reached trips are labeled. When a trip reaches the destination, a journey is added to the result set. The algorithm terminates when all optimal journeys have been found.

A motivating observation behind this is the fact that labeling stops with arrival (or departure) times is not sufficient once minimum change times are introduced. Some additional information is required to track which trips can
be reached. For example, the realistic time-expanded model of Pyrga et al. [16] introduces additional nodes to deal with minimum change times, while Connection Scan [11] uses additional labels for trips. In contrast, once we know passengers boarded a trip at a certain stop, their further options are fully defined: Either they transfer to another trip using one of the precomputed transfers, or their current trip reaches the destination, in which case we can look up the arrival time in the timetable. In either case, there is no need to explicitly track arrival times at intermediary stops.

The core of the algorithm is similar to a breadth-first search, where levels correspond to the number of transfers taken so far. As a result, it is inherently multi-criterial, similar to RAPTOR [10]. Although a graph-like structure is used, there is no need for a priority queue. A preprocessing step is required to compute transfers, but can be parallelized trivially and only takes a few minutes, even on large networks (Section 4). By omitting unnecessary transfers, both space usage and query times can be improved at the cost of increased preprocessing time.

Section 2 introduces necessary notations and definitions, before Section 3 describes the algorithm and its variants. Section 4 presents the experimental evaluation. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Notation

We consider public transit networks defined by an aperiodic timetable, consisting of a set of stops, a set of footpaths and a set of trips. A stop $p$ represents a physical location where passengers can enter or exit a vehicle, such as a train station or a bus stop. Changing vehicles at a stop $p$ may require a certain amount of time $\Delta \tau_{ch}(p)$ (for example, in order to change platforms). Footpaths allow travelers to walk between two stops. We denote the time required to walk from stop $p_1$ to $p_2$ by $\Delta \tau_{fp}(p_1, p_2)$ and define $\Delta \tau_{fp}(p, p) = \Delta \tau_{ch}(p)$ to simplify some algorithms. A trip $t$ corresponds to a vehicle traveling along a sequence of stops $p(t) = \langle p_0^t, p_1^t, \ldots \rangle$. Note that stops may occur multiple times in a sequence. For each stop $p_i^t$, the timetable contains the arrival time $\tau_{arr}(t, i)$ and the departure time $\tau_{dep}(t, i)$ of the trip at this stop. Additionally, we group trips with identical stop sequences into lines such that all trips $t$ and $u$ that share a line can be totally ordered by

$$ t \preceq u \iff \forall i \in [0, |p(t)|) : \tau_{arr}(t, i) \leq \tau_{arr}(u, i) \quad (1) $$

and define

$$ t \prec u \iff t \preceq u \land \exists i \in [0, |p(t)|) : \tau_{arr}(t, i) < \tau_{arr}(u, i). \quad (2) $$

1 More fine-grained models, such as different change times for specific platforms, can be used without affecting query times, since minimum change times are only relevant during preprocessing (Section 3.1).

2 Line and route have both been previously used for this concept; we opted for line to avoid confusion with routing and the usage of route in the context of road networks.