A Graph- and Monoid-Based Framework for Price-Sensitive Routing in Local Public Transportation Networks

Ricardo Euler
Zuse Institute Berlin, Germany
euler@zib.de

Ralf Borndörfer
Zuse Institute Berlin, Germany
borndoerfer@zib.de

Abstract

We present a novel framework to mathematically describe the fare systems of local public transit companies. The model allows the computation of a provably cheapest itinerary even if prices depend on a number of parameters and non-linear conditions. Our approach is based on a *ticket graph* model to represent tickets and their relation to each other. Transitions between tickets are modeled via transition functions over partially ordered monoids and a set of symbols representing special properties of fares (e.g. surcharges). Shortest path algorithms rely on the subpath optimality property. This property is usually lost when dealing with complicated fare systems. We restore it by relaxing domination rules for tickets depending on the structure of the ticket graph. An exemplary model for the fare system of Mitteldeutsche Verkehrsbetriebe (MDV) is provided. By integrating our framework in the multi-criteria RAPTOR algorithm we provide a price-sensitive algorithm for the earliest arrival problem and assess its performance on data obtained from MDV. We discuss three preprocessing techniques that improve run times enough to make the algorithm applicable for real-time queries.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Applied computing → Transportation

Keywords and phrases shortest path, public transit, optimization, price-sensitive, raptor, fare, operations research

Digital Object Identifier 10.4230/OASIcs.ATMOS.2019.12

Funding Our research was supported by the Federal Ministry of Transport and Digital Infrastructure(BMVI) under the project no. 19E17001C.

Acknowledgements We thank Niels Lindner and Pedro Maristany de las Casas for many fruitful discussions on the subject as well as MDV and InfraDialog GmbH for providing the data for this study.

1 Introduction

Recent progress in the field of routing algorithms for public transportation has led to several very fast algorithms [2]. Usually, these algorithms determine the best itinerary with respect to travel time in mere milliseconds. This led to the desire to optimize additional criteria such as the number of transfers or the reliability of the connection. For most users of public transportation systems the price of a journey is one of the most important criteria for assessing its quality. Unfortunately, public transportation fare systems are notoriously complex and therefore algorithmically hard to deal with. The ticket price can depend on a variety of variables such as the set of fare zones, the distance traveled, the number of stops visited, surcharges for night buses or ferries, etc. To reduce this complexity, previous research

has usually focused only on specific aspects such as zone- or distance-based prices and/or dealt with them heuristically. In this study, we present a novel and flexible framework to model the price structures of (regional) public transportation companies that is able to take all of the aforementioned criteria into account. The framework can be used to compute price-optimal journeys by applying typical multi-criteria shortest path algorithms such as RAPTOR or Dijkstra. When comparing itineraries by price, the subpath optimality principle is usually lost. An example could be taking a detour into a new fare zone to avoid paying a special connections surcharge (e.g. for using ferries). It is possible that, at a later point in the journey, the surcharge has to be paid anyway (e.g. the target station can only be reached via a ferry). In that case, the detour was a suboptimal decision. To avoid this problem, we base dominance relations between labels not on price, but on paths in a directed *ticket graph* modeling the relations between tickets. Transitions between different tickets are modeled as directed arcs and usually depend on a number of additional *fare attributes* such as fare zones or the distance traveled. We model these fare attributes as (positive) partially ordered monoids.

1.1 Related Literature

For an exhaustive overview of shortest path algorithms in road and public transportation networks please refer to [2]. It has previously been observed that shortest path problems can be generalized to ordered monoids [10] and semirings [8]. In the study of public transit routing, prices are taken into account to varying degrees. Müller-Hannemann and Schnee study fare systems that entail distance- and relation-based prices [9], i.e., fare systems that in Germany are usually associated with long-distance public transportation. They approximate fares by assigning a fixed price to every edge. Their approach, however, does not take into account fares based on fare zones and short-distance city tickets. Both of these are usually more prominent in local public transportation. Delling et al. [6][5] use their RAPTOR algorithm to compute itineraries that touch the smallest number of fare zones. The idea of restoring subpath optimality by relaxing rules for label domination is discussed in a different context in [3]. The ticket graph approach bears similarity to the finite automata used to find language-constrained shortest paths (Barrett et al [1]). It is different, however, in that it serves to evaluate paths instead of restricting the set of feasible paths. Furthermore, our approach also covers fares based on numerical attributes that are not expressed as part of a formal languages. We originally presented the idea of using a ticket graph in [4, German language].

1.2 Our Contribution

In this paper, we make two novel contributions. The first is a framework based on graphs and monoids to mathematically model public transportation fare systems. The aspects of fare systems that can be modeled include (but are not limited to): zone-based fares, distance-based fares, surcharges for special vehicles or daytime, and discounted short-distance tickets that do not allow transfers. Our second contribution is a formal definition of label domination rules for public transportation tickets while retaining the subpath optimality property. We prove that these rules do in fact yield lowest-price itineraries.

1.3 Overview

In Chapter 2, we present a framework for public transportation fare systems and show how it can be used to model several aspects of fare systems. The algorithmic treatment of fares is laid out in Chapter 3. Chapter 4 discusses how our framework can be used in the

context of the RAPTOR algorithm. We propose three different speed-up techniques and an evaluation of the framework's performance for the network of MDV, a public transit company in Saxony, Germany. Chapter 5 concludes the paper with some final remarks. The proofs of all propositions can be found in the appendix.

2 A Formal Framework for Fare Systems

We are given a (directed) routing graph $G = (V, A)$, in which arcs represent either public transport connections, footpaths or transfers between lines and/or modes of transportation. Every path p in G is associated with a ticket t and every ticket has a corresponding price $\pi(t) \in \mathbb{Q}^+$. In order to efficiently compute cheapest paths, we need a model that is able to locally describe the development of a path's ticket once arcs are added to it as well as a provably correct way of comparing those tickets. This is done in the following way: We denote the set of all available tickets by T and define a *ticket graph* $\mathcal{F} = (T, E)$ where an edge $e \in E$ models how the ticket changes when following an arc in the routing graph. Each edge in E carries a Boolean function determining the conditions under which the path transitions to a different ticket. For example, when visiting the fifth stop on a path, a short-distance ticket t_0 could be lost and a more expensive ticket t_1 would be applicable. The edge (t_0, t_1) would then carry a condition on the number of stops visited.

The transition along an edge $e \in E$ usually depends on multiple factors, e.g., the set of fare zones visited thus far, the distance traveled (in meters), surcharges for special trains, etc. These factors are picked up when relaxing an arc in the routing graph. We generalize them in two kinds of mathematical objects: abstract symbols from a symbol set S and elements of a partially ordered positive monoid $(H, +, \leq)$.

▶ **Definition 1** (Partially ordered monoid). *Let $(H, +)$ be a monoid and let $\leq$ be a partial order on H that is translation-invariant with respect to the monoid operation $+$, i.e., $h_1 \leq h_2 \Rightarrow h_1 + x \leq h_2 + x \,\forall\, h_1, h_2, x \in H$. We call $(H, +, \leq)$ a partially ordered monoid. We call $(H, +, \leq)$ positive, if $e \leq h \,\forall\, h \in H$ where e is the neutral element of $(H, +)$.*

A common example for fare systems is the power set 2^Z of a set of fare zones Z combined with k numerical fare attributes in $\mathbb{Q}^k$. In this case, $H = 2^Z \times \mathbb{Q}^k$. For $h_1 = (z_1, r_1)$, $h_2 = (z_2, r_2) \in H$, we define $h_1 + h_2 = (z_1 \cup z_2, r_1 + r_2)$ and $h_1 \leq h_2$ if and only if $z_1 \subseteq z_2$ and $r_1 \leq r_2$. Translation invariance in $(H, +, \leq)$ is inherited from the translation invariance in $(2^Z, \cup, \subseteq)$ and $(\mathbb{Q}^k, +, \leq)$.

Every arc $a \in A$ of G is annotated with a *fare attribute* $\mathcal{A}(a) = (\mathcal{A}^s(a), \mathcal{A}^h(a)) \in S \times H$ that is picked up when relaxing the arc. Every vertex $v \in V$ is annotated with a *start state* $\mathcal{S}(v) = (\mathcal{S}^t(v), \mathcal{S}^h(v)) \in T \times H$ giving an initial ticket and element from the monoid.

Using this notation we can define the *fare state* of a path.

▶ **Definition 2** (Fare State). *We call an element $f \in T \times H$ a fare state. The set of all fare states is denoted by $F := T \times H$. We use f^t and f^h to denote its components.*

A fare state contains all information necessary for calculating the price of a journey (the ticket associated with the path) as well as all information necessary to decide domination between paths. We now want to enable the tracking of the fare states along paths in G. To do so, we formalize the notion of the *fare transition function* on arcs in the ticket graph. The definition is intentionally left rather general to capture a large number of possible ticketing conditions.

▶ **Definition 3** (Fare Transition Function). *We call a function* $Tr : E \times S \times H \longrightarrow \{0,1\}$ *a fare transition function for the ticket graph* $\mathcal{F}$ *if*

$$\forall\, e = (t_1, t_2) \in E : \forall\, s \in S : \forall\, h \in H : \sum_{e \in \delta^+(t_1)} Tr(e, s, h) \in \{0,1\}.$$

Hence, a fare transition function allows only one well-defined transition from any fare state $f \in F$. We use the notion of fare transition functions to define the update of tickets when relaxing an arc of the routing graph.

▶ **Definition 4** (Ticket Update Function). *We introduce a ticket update function* $Up : F \times A \longrightarrow F$ *in the following way. Let* $f = (f^t, f^h) \in F, a \in A$. *Then, we define* $\tilde{f} = Up(f)$ *by*

$$\tilde{f^h} := f^h + \mathcal{A}^h(a) \tag{1}$$

$$\tilde{f^t} := \begin{cases} head(e) \text{ if } \exists\, e \in \delta^+(f^t) \text{ with } Tr(e, \mathcal{A}^s(a), \tilde{f^h}) = 1 \\ f^t \text{ otherwise.} \end{cases} \tag{2}$$

We now have a tool at our disposal to track the tickets along a path in G in the ticket graph $\mathcal{F}$. Each path in G can be associated with a sequence of fare states in F.

▶ **Definition 5** (Path-Induced Fare Sequence). *We call a sequence of fare states* $(f_1, \ldots, f_n)$ *path-induced if there is a path* $p = (v_1, \ldots, v_n)$ *with the following properties:*

1. $f_1 = \mathcal{S}(v_1)$
2. $f_i = Up(f_{i-1}, (v_{i-1}, v_i))\, \forall\, i = 2, \ldots, n$

We call the fare state $f(p) := f_n$ *the fare state of the path* p.

Combining all the above definitions we arrive at the notion of *conditional fare networks* which can precisely capture a fare system.

▶ **Definition 6** (Conditional Fare Network). *Let* $G = (V, A)$ *be a routing graph and let the following be given:*

1. *the space of fare attributes* $S \times H$ *as product of a set of symbols* S *and a partially ordered, positive monoid* $(H, +, \leq)$,
2. *a set of tickets* T,
3. *a cycle-free ticket graph* $\mathcal{F} = (T, E)$ *with transition function* $Tr : E \times S \times H \longrightarrow \{0,1\}$,
4. *arc attributes* $\mathcal{A}(a) \in S \times H \quad \forall\, a \in A$,
5. *start fare states* $\mathcal{S}(v) \in T \times H \quad \forall\, v \in V$ *and*
6. *a price function* $\pi : T \longrightarrow \mathbb{Q}_+$ *that is monotonously non-decreasing along directed paths in* $\mathcal{F}$, *i.e., if there is a directed* $t_1 - t_2$*-path in* $\mathcal{F}$ *for* $t_1, t_2 \in T$, *then* $\pi(t_1) \leq \pi(t_2)$.

We call the six-tuple $(\mathcal{F}, \mathcal{A}, \mathcal{S}, S, Tr, \pi)$ *a conditional fare network* $\mathcal{N}$ *of* G.

Note that cycle-freeness in $\mathcal{F}$ and the monotonicity condition on π as well as the positivity of H ensure that no price-decreasing cycles exist in G. We consider those assumptions natural enough that most reasonable price system should satisfy them.

Soon, we will see that dominance relations between paths need to be based on their fare states instead of their price. Thus, we can drop the monotonicity condition on π while still retaining optimality. In this case, however, price-based target pruning (confer Section 4.2), which proved essential in ensuring acceptable performance for our shortest path search, cannot be applied.

Having introduced conditional fare networks, we now define the price-sensitive earliest arrival problem.

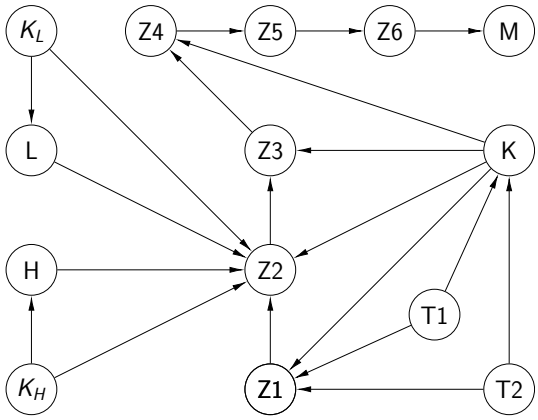


Figure 1 Ticket graph for MDV. There is a total number of fourteen different tickets.

▶ **Definition 7** (Price-Sensitive Earliest Arrival Problem). *Let a public transportation network be given as a graph $G = (V, A)$ and let $(\mathcal{F}, \mathcal{A}, \mathcal{S}, S, Tr, \pi)$ be a conditional fare network of G. Let for all $a \in A$ a time-dependent FIFO travel time function $c(a) : I \to I$ be given, where I is the set of time points. Finally, let $P_{s,t}$ be the set of all s,t-paths in G for some $s, t \in V$. Then, the price-sensitive earliest arrival problem (PSEAP) is defined as finding a Pareto-set of s,t-paths $P^*_{s,t} \subset P_{s,t}$ in G, such that*

$$\forall\, p^* \in P^*_{s,t} \nexists\, p \in P_{s,t} : \pi(p) \leq \pi(p^*) \wedge c(p) \leq c(p^*) \wedge (\pi(p) < \pi(p^*) \vee c(p) < c(p^*)). \tag{3}$$

In our following theoretical discussion, we will ignore the arrival time aspect of PSEAP and focus only on the fare framework. The correctness results carry over to the full version of PSEAP.

▶ **Example 8** (The Fare System of MDV). The fare systems of MDV divides its operational area into 67 fare zones Z. A ticket Z_i is applicable if the itinerary touches i zones from Z. Once a total of seven fare zones have been touched a maximum price M is reached that does not change. Two fare zones, Halle and Leipzig, are cities and are more expensive than other zones. They offer special tickets H and L. They, however, count as normal zones for all itineraries that pass through multiple fare zones. Several smaller cities are part of larger fare zones, but allow for discounted tickets (T_1 and T_2) when traveling in the city only. They do not count as fare zones of their own. For Halle and Leipzig there are discounted tickets for short trips (K_H and K_L), which do not allow for transfers and are valid for a maximum number of four stations on the itinerary. Discounted tickets exist also for other zones (K), but these are cheaper and depend on the length of the itinerary (4 km maximum). Figure 1 depicts the associated ticket graph $\mathcal{F}$. The monoid $(H, +_H, \leq_H)$ is defined by $H := (2^Z, \mathbb{N}^2, \{0,1\})$, where Z captures the fare zones, $\mathbb{N}^2$ the distance traveled in meters and the number of stations and $\{0,1\}$ the existence of transfers. Addition $+_H$ and partial order $\leq_H$ are induced from the respective operations in 2^Z and $\mathbb{N}^2$ and the logical or $\vee$ on $\{0,1\}$. Note that there is no connection between, e.g., Z_2 and Z_4. This is due to the fact that we can touch only one fare zone at any station. Also, there are no discounted trips that cover more than three fare zones and hence there is no arc between K and Z_5. This highlights that the structure of the routing graph influences the structure of the ticket graph.

A list of all fare transition functions can be found in the appendix.

Transfer Penalties, Footpaths and Surcharges

Footpaths are modeled as arcs with the arc attribute (S_0, e), where $S_0 \in S$ is a symbol that cannot activate a ticket transition and $e \in H$ is the neutral element of the monoid. The fare attribute is set to e so as to not modify the current fare state. The transition from a footpath to a public transportation vehicle requires some care. Assume we walk from station v_0 to v_1 along arc $a_0 = (v_0, v_1)$ to take a vehicle along $a_1 = (v_1, v_2)$ to reach v_2. Some fare systems use the number of stations a path touches to calculate prices. Here, this number would obviously be two. Counting a station when relaxing a_0 is a mistake if the optimal journey would be to continue on foot. Counting both v_1 and v_2 when relaxing a_1 is also wrong since this would overcount the number of stations for every itinerary that reaches v_1 via a vehicle. Hence, the graph model needs to be extended by splitting up stations into vertices for every route and a vertex that is connected to footpaths. These vertices are then connected via transfer arcs and boarding arcs. We can also have arc attributes different from (S_0, e) on transfer arcs. This allows us to make the applicable ticket dependent on the number of transfers. Arc attributes on arcs representing boarding can be used to model surcharges for the route boarded. For more details on how to build these expanded graphs, we refer to [7].

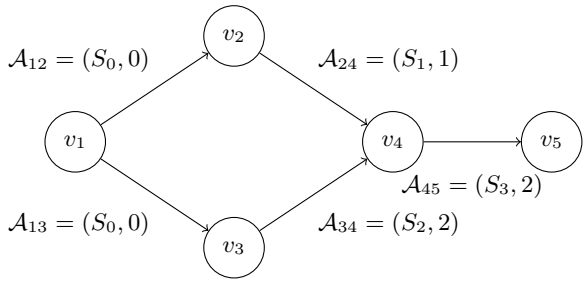
Neutral Zones

Some fare systems that are based on fare zones contain neutral zones. Stations in a neutral zone can be counted as part of either of its neighboring zones, whichever is cheapest for the costumer. This is meant to mitigate sharp price increases at fare zone borders. MDV uses them as well as several other German railway companies (e.g. Verkehrsverbund Bremen/Niedersachsen GmbH). Neutral zones can be incorporated by label duplication: Assume a neutral zone neighbors n fare zones. We associate each arc a whose $head(a)$ represents a station in the neutral zone with n fare attributes, one for each fare zone it could possibly be part of. When settling the vertex in a shortest path search, the current fare state is updated once for each fare attribute thereby creating n new labels.

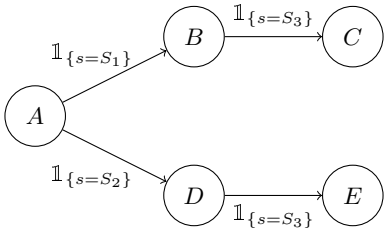
3 The Fare Framework in Routing Algorithms

Classical shortest path algorithms rely on dynamic programming and the subpath optimality condition [3]. That is, every subpath of an optimal s,t-path is in itself an optimal path. When comparing paths in G naively by means of the price function π, the subpath optimality condition is usually violated. Think about taking a local detour to avoid a fare zone: Later on, travelers may be forced to cross the zone due to the infrastructure, turning the locally dominant detour into a suboptimal choice. On the other hand, a locally dominated subpath might still lead to an optimal s,t-path. This type of problem persists in our framework: the transition between tickets depends on the fare attributes already collected, but also on the structure of the reachable ticket graph and the transition functions of reachable fare arcs. Example 9 highlights that problems can already arise even with simple examples of ticket graphs.

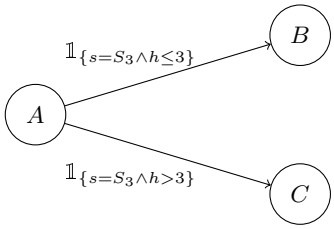
▶ **Example 9** (Label Dominance in Figure 2)**.** Consider the routing graph **(a)** together with the conditional fare network **(b)**. Examining the paths $p_1 = (v_1, v_2, v_4)$ and $p_2 = (v_1, v_3, v_4)$, we find their respective fare states are $f(p_1) = (B, 1)$ and $f(p_2) = (D, 2)$. Extending them by v_5 to p_1' and p_2' yields $f(p_1) = (C, 3)$ and $f(p_2) = (E, 4)$. Comparing fare states by price would indicate that p_1 could be cut off at v_4 since $\pi(B) > \pi(D)$. This is a suboptimal choice



(a) Routing Graph.



(b) Ticket Graph.



(c) Ticket Graph with Divergence.

Figure 2 Example of a routing graph **(a)** with two possible conditional fare networks **(b)** and **(c)**. For both, the underlying partially ordered monoid is $(\mathbb{R}, +, \leq)$, the symbol set is $S = \{S_0, S_1, S_2, S_3\}$ and the start state for all vertices v_i with $i = 1, \ldots, 5$ is $\mathcal{S}(v_i) = (A, 0)$. We give prices for the tickets as $\pi(a) = 0$, $\pi(B) = 2$, $\pi(C) = 3$, $\pi(D) = 1$ and $\pi(E) = 5$. Transition functions are displayed as indicator functions on fare arcs. Using the ticket graph **(b)**, the upper v_1, v_5-path yields ticket C, while the lower path yields ticket E. Using ticket graph **(c)**, the upper path yields ticket B, the lower path yields ticket C.

as p_1' dominates p_2' since $\pi(C) < \pi(E)$. Hence, price cannot be used as dominance criterion for fare states. A natural alternative would be to use the partial order defined by paths in the ticket graph, instead. A ticket t_1 then dominates a ticket t_2 if there is a t_1, t_2-path. This would render the tickets B and D and the tickets C and E mutually incomparable. The idea, however, comes with problems of its own. To see this, consider now the conditional fare network **(c)**. At v_4, we have $f(p_1) = (A, 1)$ and $f(p_2) = (A, 2)$ and hence both paths are equivalent and it would be sensible to keep only one of them based on the relation between $f^h(p_1)$ and $f^h(p_2)$. By relaxing (v_4, v_5), we obtain $f(p_1') = (B, 3)$ and $f(p_2') = (C, 4)$, which are incomparable, i.e., the fare states of p_1' and p_2' diverged from comparable to incomparable. Consequently, any dominance ruling cutting off either p_1 or p_2 would be defective.

To mitigate these and similar problems, we might assume a general incomparability of fare states. This comes down to enumerating all s, t-paths and simply comparing them by price. However, in a sensibly designed fare system it is usually clear which ticket is better and taking a cheaper subpath should usually not turn out more expensive overall. In the remainder of this chapter, we propose a more tailored approach. It generally bases domination rules on path relationships but adds exceptions to cover cases in which it is not safe to do so.

3.1 Dominance for Fare States

We want to define a comparison operator for fare states that restores subpath optimality while not relaxing dominance too strongly.

To do so, we partition the ticket set T into three disjoint *comparability groups*: C_F (full comparability), C_P (partial comparability), C_N (no comparability). Based on the partition $C = (C_F, C_P, C_N)$, we define a comparison operator for fare states.

▶ **Definition 10** (Comparability of Fare States). *Let $f_1 = (t_1, h_1)$, $f_2 = (t_2, h_2)$ be fare states. We say $f_1 \leq_C f_2$ if and only if $t_1 \notin C_N$, $h_1 \leq h_2$ and*

$$t_1 = t_2 \quad \text{if } t_1 \in C_P \tag{4}$$
$$\exists\, t_1, t_2\text{-path in } \mathcal{F} \quad \text{if } t_1 \in C_F. \tag{5}$$

If and only if $f_1 \leq_C f_2$ and either $h_1 < h_2$ or $t_1 \neq t_2$, we say that f_1 is strictly lesser than f_2, i.e., $f_1 <_C f_2$.

We denote by $P^f_{s,t}$ the set of all paths Pareto-optimal with respect to $\leq_C$, i.e.,

$$p^* \in P^f_{s,t} \Rightarrow \nexists\, s,t\text{-path } p : f(p) <_C f(p^*). \tag{6}$$

Note that $P^f_{s,t}$ is not equal to $P^*_{s,t}$. Proposition 16 shows that it is in fact a superset of the set of all price-optimal paths, which we denote by $P^\pi_{s,t}$. We call paths in $P^f_{s,t}$ *state-optimal* and paths in $P^\pi_{s,t}$ *price-optimal*.

The partition C has to be defined in a way that monotonicity of the update function along all arcs $a \in A$ is not violated[1], i.e.,

$$\forall\, f_1, f_2 \in F : f_1 \leq_C f_2 \Longrightarrow \forall\, a \in A : Up(f_1, a) \leq_C Up(f_2, a). \tag{7}$$

This condition is enough to ensure that a weaker form of subpath optimality holds.

▶ **Proposition 11** (Weak Subpath Optimality). *Let $G = (V, A)$ be a routing network and $\mathcal{N} = (\mathcal{F}, \mathcal{A}, \mathcal{S}, S, Tr, \pi)$ be its conditional fare network. Let $p^* \in P^f_{s,t}$ be a state-optimal s,t-path in G for some $s, t \in V$. Then, there is a path $p' = (s = v_0, v_1, \ldots, v_{n-1}, v_n = t) \in P^f_{s,t}$ with $f_{p^*} = f_{p'}$, such that every subpath $p'' = (v_0, \ldots, v_l), l < n$ of p' is a state-optimal v_0, v_l-path.*

Note that Proposition 11 doesn't imply that every subpath of a state-optimal path is state-optimal. It, however, implies that we can discard all state-optimal paths without this property since a path with equal fare state still remains in $P^f_{s,t}$. Hence, all classical algorithms relying on subpath-optimality can still be applied. Note also that Equation 7 need not hold for all $f_1, f_2 \in F$ but only for those that might occur on paths in G.

3.2 Comparability Partitions

In choosing C_F, C_P and C_N, there is some degree of freedom. We want C_F to be as big and C_N as small as possible while still fulfilling Equation 7. It is clear that the choice does not only depend on $\mathcal{F}$ and Tr but also on the structure of G and its arc attributes $\mathcal{A}$. Choosing the partition depending on G and $\mathcal{A}$ would require some preprocessing of G. We propose a solution that depends only on $\mathcal{F}$ and Tr and needs less recomputation when changes in the network occur. To deal with this dependency, we use the notion of a vertex's *reach*.

▶ **Definition 12** (Reach). *Let $\mathcal{F} = (T, E)$ be a directed graph. We define the reach $R(t)$ of a vertex $t \in T$ as the subgraph induced by all vertices reachable from t, i.e.,*

$$R(t) := \mathcal{F}[\{\tilde{t} \in T : \exists\, t, \tilde{t}\text{-path}\}]. \tag{8}$$

[1] Shortest path algorithms on graphs with weights from partially ordered monoids require the monoid operation to be translation-invariant with respect to the partial order. Since the fare states and arc attributes do not belong to the same structure, the notion of translation invariance is relaxed to a monotonicity formulation.

To simplify our notation, we introduce operators that represent path relations. If there is a path in $\mathcal{F}$ between $t_1, t_2 \in T, t_1 \neq t_2$, we write $t_1 \longrightarrow t_2$. We write $t_1 \xrightarrow{=} t_2$ if either $t_1 \longrightarrow t_2$ or $t_1 = t_2$.

▶ **Definition 13** (No-overtaking Property). *Let $\mathcal{F}^*$ be a vertex-induced subgraph of $\mathcal{F}$. We say it has the no-overtaking property if for all $e = (t_1, t_2) \in \mathcal{F}^*$, $(s, h) \in S \times H$ and $\tilde{t}_1 \in T$ with $t_1 \xrightarrow{=} \tilde{t}_1 \xrightarrow{=} t_2$ the following holds:*

$$Tr(e, s, h) = 1 \Rightarrow \forall\, \tilde{h} \geq h \in H : \exists (\tilde{t}_1, \tilde{t}_2) \in E : Tr(\tilde{t}_1, \tilde{t}_2, s, \tilde{h}) = 1 \text{ and } t_2 \xrightarrow{=} \tilde{t}_2. \tag{9}$$

The no-overtaking property bears some resemblance to the FIFO (first-in, first-out) property: A worse fare state, i.e., either worse fare attributes from H or a worse ticket, cannot give rise to a better fare state when relaxing the same arc in the routing graph. Note that the condition is necessary not only for the neighbors of t but for $R(t)$.

Subgraphs with the no-overtaking property allow for the strictest domination rules. We use them as comparability group C_F.

▶ **Definition 14** (Comparability Partition). *Let $G = (V, A)$ be a routing network and $\mathcal{N} = (\mathcal{F}, \mathcal{A}, \mathcal{S}, S, Tr, \pi)$ be its conditional fare network. We define*

$$C_F := \{t \in T : R(t) \text{ traceable and has the no-overtaking property } \} \tag{10}$$

$$C_P := \{t \in T \backslash C_F : \forall\, e \in R(t) : \forall\, s \in S : \forall\, h_1, h_2 \in H : Tr(e, s, h_1) = Tr(e, s, h_2)\} \tag{11}$$

$$C_N := \{t \in T \backslash (C_F \cup C_P)\}. \tag{12}$$

To be in the set C_F, the reach of a ticket has to be traceable, i.e., contain a Hamiltonian path. This condition is needed to avoid the divergence seen in Example 9. If a ticket has non-traceable reach it is placed in C_P. Transition functions here must be independent of H. This also ensures that comparable fare states do not diverge in an incomparable state. All remaining tickets are added to C_N. Fare states containing tickets from C_N cannot be compared at all.

The comparison operator defined by this comparability partition fulfills Equation 7.

▶ **Proposition 15** (Monotonicity of the Comparability Partition). *The partial order defined by Definitions 10 and 14 fulfills the monotonicity condition*

$$\forall\, f_1, f_2 \in F : f_1 \leq_C f_2 \Longrightarrow \forall\, a \in A : Up(f_1, a) \leq_C Up(f_2, a). \tag{13}$$

Propositions 15 and 11 enable us to apply dynamic programming shortest path algorithms to the PSEAP using the comparability partition from Definition 14. However, we obtain only the set of state-optimal paths. It remains to show that this set contains the cheapest price itinerary.

▶ **Proposition 16** (Correctness). *Let $\pi^* := \min_{P_{s,t}} \pi(p)$. Then, there is at least one s,t-path p^* with $\pi^* = \pi(p^*)$ and $p^* \in P^f_{s,t} := \{\tilde{p}\ s,t\text{-path} : \nexists\, p : f(p) <_C f(\tilde{p})\}$.*

▶ **Example 17** (Dominance Rules for MDV Fares). In the graph in Figure 1 all nodes have traceable reach. It is also easy to see that the no-overtaking property holds for all tickets as well and hence $C_F = \{T_1, T_2, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, M, H, L, K_L, K_H, K\}$, $C_P = \emptyset$ and $C_N = \emptyset$. Note that changing the transition condition from Z_1 to Z_2 from $|h^z| > 1$ to $|h^z| = 1$ breaks the no-overtaking property in cases where more than one fare zone is covered by a station. Since this is never the case, we can replace the inequalitiy by an equality while maintaining optimality.

4 Price-Sensitive RAPTOR

In this last section, we will discuss how to integrate conditional fare networks into the RAPTOR-algorithm. Finally, we introduce some speed-up techniques and evaluate their performance on real-world instances obtained from MDV. We will change notation slightly and use $P^*_{s,t}$ to refer to the Pareto-set optimized for transfers, arrival time and price and $P^f_{s,t}$ to refer to the Pareto-set optimized for transfers, arrival time and fare state.

4.1 Applying the Framework to RAPTOR

We use our framework to implement a price-sensitive version of the multi-criteria RAPTOR algorithm (McRAPTOR) [6]. That is, we solved the earliest arrival problem with price as an additional optimization criterion. RAPTOR implicitly optimizes also for the number of transfers. To facilitate discussion, we presented the framework in a graph-based context. However, RAPTOR does not use a graph model but works directly on the timetable. It operates in rounds $k = 1, \dots, n$. In each round, a set of marked routes is visited and labels are propagated along them. Labels are updated in this process by reading the new arrival time directly from the route's arrival time array. At each station, it is checked whether the new label improves upon the current optimal label. If so, that local label is updated and the station is marked as a starting point for the next round. The standard RAPTOR version labels all stations solely with arrival times.

Adapting our findings to work with RAPTOR is straightforward: Each pair of adjacent stations (v_i, v_{i+1}) on a trip can be interpreted as an arc. Hence, we store for every trip not only an array of arrival times but also an array of all fare attributes. A label $l_i = (t_i, f_i)$ (at station v_i) consists of an arrival time t_i and a fare state f_i. When traversing along a trip from station v_i to v_{i+1}, the arrival time is updated and the fare state is set to $f_{i+1} = Up(f_i, (v_i, v_{i+1}))$. Dominance of labels is checked according to the theory developed above while also taking arrival times into account. Update steps that were associated with arcs modeling transfers are performed whenever the algorithm hops on a new route. This requires storing an additional array with fare attributes at every station to represent transfer costs for every trip at the station. Note that, in most cases, it is enough to store one array for each route instead of trip, as fare attributes seldom vary among the trips of a route. Since walking is free, fare states do not need to be updated in the footpath stage. Depending on the data set, it might thus be possible to save money by walking a long distance in the middle of an itinerary. This kind of journey can be excluded during postprocessing.

4.2 Speed-up Techniques

Price-based Target Pruning

In RAPTOR as well as Dijkstra's algorithm, it is possible to prune labels that are worse than the labels that have already been found at the target station. Naturally, the same speed-up technique is also possible for our algorithm. Moreover, we need not use $\leq_C$ to compare fare states. Since the labels at the target station are never updated and the price function π is non-decreasing, a path already more expensive than the incumbent cheapest s,t-path cannot be price-optimal. Hence, we can prune all labels with a fare state f with $\pi(f^t) \geq \pi^*$, with π^* being the incumbent price at the target station.

Bounded McRAPTOR

By design of fare systems the cheapest path is often among the fastest as detours are penalized by increases in both price and travel time. Therefore, it seems beneficial for a multi-criteria search to compute a minimal travel time itinerary early on by running a standard RAPTOR query. The labels obtained in this first stage can then be used to prune the multi-criteria search. Let t_k be the optimal arrival time at the target station in round k of the first stage (computed with RAPTOR). During round k of McRAPTOR, we prune every label that has an arrival time t with $t > t_k + \epsilon$. Note that this possibly cuts off optimal paths from $P^f_{s,t}$ (and $P^*_{s,t}$) if ϵ is chosen to be small. This technique, alongside an even tighter pruning scheme, has been introduced in [5].

Problem Specific Speed-ups

Certain dimensions in H might only be relevant for some tickets in T. For example, many short-distance tickets depend on the number of stations visited while this number is irrelevant for all other tickets that can be reached from that ticket. We can therefore alter the comparison operator $\leq_C$ for those tickets to ignore the number of stations. Hence, more labels become comparable which results in a smaller Pareto-set $P^{pss}_{s,t}$ with $P^*_{s,t} \subseteq P^{pss}_{s,t} \subseteq P^f_{s,t}$.

4.3 Computational Results

We implemented the McRAPTOR algorithm in C++17 compiled with gcc 8.1.0 and -03 optimization. All tests were conducted on Dell Poweredge M620 machines with 64 GB of RAM. While the general structure of the MDV price system is captured in our model, our computations deviate from the prices charged by MDV in the following three cases: A list of relations that is, contrary to the general rules, not eligible for the short-distance discount is not taken into account. Neutral zones are as of yet not implemented. Instead, we add each neutral zone to one of its surrounding zones. Stations and fare zones that a route passes through without stopping are not represented in the available data and therefore cannot be taken into account.

From MDV's[2] GTFS[3] feed, we extracted the timetable of the 22 May 2019. It contains 4,371 stations, 18,215 trips, 5347 routes and 845 footpaths. The original footpath set was not transitively closed[4]. After computing the transitive closure, there were 1,029 footpaths. 40 unconnected stations as well as 960 duplicate trips were removed from the data. We then chose a test set of 5,000 queries uniformly at random from the set of connected stations.

In Table 1, we depict results for six different versions of RAPTOR. A standard RAPTOR query (*RAPTOR*) and a McRAPTOR query optimizing for fare zones (*zones*) were included for comparison. All other versions optimize for price and travel time using the framework presented above. Version *fare* uses price-based target pruning as presented in Section 4.2. Employing standard target pruning (using $\leq_C$ for comparison) instead yielded extremely poor results in exploratory computations and is therefore not included in this study (One query found 1000 paths of which only four were in $P^*_{s,t}$). Version *pss* additionally uses

[2] The MDV GTFS timetable is licensed under `CC BY 4.0` and is publicly available under `https://www.mdv.de/informationen/downloads/`.
[3] Fare data is not part of the feed and was obtained separately via InfraDialog GmbH.
[4] RAPTOR requires a transitively closed footpath set [6].

Table 1 Computation results for 5000 queries in the MDV network. All values depicted are averages. 30 queries for which no itinerary was found were excluded. The column *pareto* the size of the Pareto-set as computed by RAPTOR, while *sols* reports the size of $P^*_{s,t}$. We also report the number of scanned routes (*scan*). For *bound60* and *bound30*, the results of the actual multi-criteria query are reported (second phase). *Total* combines the results for the first and second phase.

	Criteria				Technique			Query				Total	
	transfers	*time*	*zones*	*fare*	*TP*	*EB*	*PSS*	*pareto*	*sols*	*time[ms]*	*scan*	*time[ms]*	*scan*
RAPTOR	•	•	◦	◦	◦	◦	◦	1.58	–	6	11032	–	–
zones	•	•	•	◦	◦	◦	◦	49.53	–	5066	19563	–	–
fare	•	•	◦	•	•	◦	◦	6.01	2.66	48084	16457	–	–
pss	•	•	◦	•	•	•	◦	3.5	2.66	406	14842	–	–
bound60	•	•	◦	•	•	•	•	2.51	2.21	274	14322	280	25354
bound30	•	•	◦	•	•	•	•	2.34	2.14	260	14063	266	25085

problem-specific speed-ups and *bound60* (*bound30*) computes bounds to cut off all paths that are more than 60 (30) minutes slower than those computed in the first phase. The field *pareto* reports the average number of solutions computed by the RAPTOR/McRaptor query. When optimizing for price and arrival time, this set is bigger than the size of $P^*_{s,t}$, which is reported as *sol*. Note that these sets are smaller for *bound60* and *bound30* since they compute restricted Pareto-sets. When taking fare zones into account (*zones*), the size of the Pareto-set increased from 1.58 (*RAPTOR*) to a staggering 49.53. This is reflected in a high run time of more than 5 s. The price-based target pruning used in *fare* results in a significantly smaller Pareto-set $P^*_{s,t}$ (6.01) on average. The average size of $P^*_{s,t}$ is 2.66. This indicates that most of the itineraries computed by *zones* are not interesting for a typical costumer and the fare framework can be leveraged to avoid their computation. However, the framework is computationally more involved and requires optimization of two additional criteria (distance and visited stations), which leads to run times of 48 s on average. In *pss*, fare zones are not compared anymore after reaching ticket M. Distance and the number of visited stations is only considered for the tickets K, K_L and K_H, T_1 and T_2. This results in an average run time of 406 ms which is even considerably faster than the *zones* queries. Computing bounds in a first phase reduces run times further to 280 and 266 ms, respectively, while not reducing the number of itineraries found too much.

5 Conclusion

We presented a novel framework for modeling fare systems of public transportation companies. It is independent of the shortest path algorithm used and can be used to solve price-sensitive earliest arrival queries in real-world networks. Our test case forces the implicit optimization for distance, number of stations visited and fare zones, which resulted in slow run times and large Pareto-sets. Both can be greatly improved by utilizing insights into the price structure to tighten dominance rules, which lead to the framework faring even better than a purely fare zone-based McRAPTOR. The speed-up is, however, dependent on the ticket transition rules and thus, performance might vary significantly depending on the fare system in question.

References

1 Chris Barrett, Riko Jacob, and Madhav Marathe. Formal-Language-Constrained Path Problems. *SIAM J. Comput.*, 30(3):809–837, May 2000. doi:10.1137/S0097539798337716.

2 Hannah Bast, Daniel Delling, Andrew V. Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route Planning in Transportation Networks. In *Algorithm Engineering - Selected Results and Surveys*, volume 9220 of *Lecture Notes in Computer Science*, pages 19–80. Springer, 2016.

3 Annabell Berger and Matthias Müller-Hannemann. Subpath-Optimality of Multi-Criteria Shortest Paths in Time- and Event-Dependent Networks. Technical report, Institute of Computer Science, Martin-Luther-Universität Halle-Wittenberg, 2009. URL: http://wcms.uzi.uni-halle.de/download.php?down=10850&elem=2163494.

4 Ralf Borndörfer, Ricardo Euler, Marika Karbstein, and Fabian Mett. Ein mathematisches Modell zur Beschreibung von Preissystemen im öV. Technical Report 18-47, ZIB, Takustr. 7, 14195 Berlin, 2018. URL: urn:nbn:de:0297-zib-70564.

5 Daniel Delling, Julian Dibbelt, and Thomas Pajor. Fast and Exact Public Transit Routing with Restricted Pareto Sets. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 54–65, 2019. doi:10.1137/1.9781611975499.5.

6 Daniel Delling, Thomas Pajor, and Renato F. Werneck. Round-Based Public Transit Routing. *Transportation Science*, 49(3):591–604, 2015. doi:10.1287/trsc.2014.0534.

7 Yann Disser, Matthias Müller-Hannemann, and Mathias Schnee. Multi-criteria Shortest Paths in Time-dependent Train Networks. In Catherine C. McGeoch, editor, *Proceedings of the 7th International Conference on Experimental Algorithms*, WEA'08, pages 347–361, Berlin, Heidelberg, 2008. Springer-Verlag. doi:10.1007/978-3-540-68552-4_26.

8 Mehryar Mohri. Semiring Frameworks and Algorithms for Shortest-distance Problems. *J. Autom. Lang. Comb.*, 7(3):321–350, January 2002. URL: http://dl.acm.org/citation.cfm?id=639508.639512.

9 Matthias Müller-Hannemann and Mathias Schnee. Paying less for train connections with MOTIS. In *Proceedings of the 5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, volume 2 of *OpenAccess Series in Informatics*, page 657, January 2005. doi:10.4230/OASIcs.ATMOS.2005.657.

10 U. Zimmermann. *Linear and combinatorial optimization in ordered algebraic structures*, volume 10 of *Annaly of discrete mathematics*. North-Holland, 1981.

A Fare Transition Functions for MDV

In the following, we provide the fare transition function associated with the fare transition arcs in Figure 1. We refer to the dimensions of the partially ordered monoid $(H, +_h, \leq_h)$ as $H := H^z \times H^s \times H^d \times H^t$, where $H^Z = 2^Z$, $H^s = \mathbb{N}$, $H^d = \mathbb{N}$ and $H^t = \{0, 1\}$. The set H^Z represents all possible combinations of fare zones, H^s and H^d represent distances measured in the number of stations and in meters, respectively, and H^t, whether a trip contains a transfer. The symbol set is $S = \{H, L, T_1, T_2, S_0\}$, where H and L are associated with all stations in Halle and Leipzig, respectively. The symbol T_1 is associated with stations in cities that allow the T_1 price, the symbol T_2 with stations in cities that allow the T_2 price. All remaining stations have the symbol S_0. Let $s \in S$ and $h = (h_z, h_s, h_d, h_t) \in H$. Then, the fare transition function is defined by

$$
\begin{aligned}
Tr(Z_i, Z_{i+1}, s, h) = 1 &\Leftrightarrow |h_z| > i && \forall\, i \in 1, \dots, 6 \text{ with } Z_7 := M \\
Tr(T_j, Z_1, s, h) = 1 &\Leftrightarrow \\
&\quad s \neq T_j \wedge (h_t = 1 \vee h_d > 4) && \forall\, j = 1, 2 \\
Tr(T_j, K, s, h) = 1 &\Leftrightarrow \\
&\quad s \neq T_j \wedge (h_t = 1 \wedge h_d \leq 4) && \forall\, j = 1, 2 \\
Tr(K, Z_i, s, h) = 1 &\Leftrightarrow |h_z| = i \wedge (h_t = 1 \vee h_d > 4) && \forall\, i = 1, 2, 3, 4 \\
Tr(L, Z_2, s, h) = 1 &\Leftrightarrow s \neq L \\
Tr(H, Z_2, s, h) = 1 &\Leftrightarrow s \neq H \\
Tr(K_L, Z_2, s, h) = 1 &\Leftrightarrow s \neq L \wedge (h_t = 1 \vee h_s > 4) \\
Tr(K_H, Z_2, s, h) = 1 &\Leftrightarrow s \neq H \wedge (h_t = 1 \vee h_s > 4) \\
Tr(K_L, L, s, h) = 1 &\Leftrightarrow s = L \wedge (h_t = 1 \vee h_s > 4) \\
Tr(K_H, H, s, h) = 1 &\Leftrightarrow s = H \wedge (h_t = 1 \vee h_s > 4).
\end{aligned}
$$

B Proof of Proposition 11

Proof. Let $p^* = (s = v_0, v_1, \dots, v_{n-1}, v_n = t) \in P^f_{s,t}$ be a state-optimal s,t-path and let $(f_0, \dots, f_n)$ be the fare sequence associated with it. Assume there is another s,t-path $\tilde{p} = (s = u_0, u_1, \dots, u_{l-1} = v_{n-1}, u_l = t)$ with fare states $(f_0, \tilde{f}_1, \dots, \tilde{f}_l)$. Let k be the largest integer such that $v_{n-k} = u_{l-k}$, i.e. the paths $(v_{n-k}, \dots, v_n)$ and $(u_{l-k}, \dots u_l)$ are equal. Now assume $\tilde{f}_{l-k-1} <_C f_{n-k-1}$. By definition, $f_{n-k} = Up(f_{n-k-1}, (v_{n-k-1}, v_{n-k}))$ and $\tilde{f}_{l-k} = Up(\tilde{f}_{l-k-1}, (u_{l-k-1}, u_{l-k}))$. We apply Equation 7 to obtain $\tilde{f}_{l-k} \leq_C f_{n-k}$. By repeating the process for $i \in \{k-1, \dots, 0\}$, we find $\tilde{f}_l \leq_C f_n$. Since p^* was state-optimal, it follows that $\tilde{f}_l =_C f_n$ and consequently p is also state-optimal. Since the number of paths in G is finite, we can repeat this procedure to find the path p'. ◀

C Proof of Proposition 15

Proof. Let $a \in A$ and $f_1, f_2 \in F$ such that $f_1 \leq_C f_2$. We write $\tilde{f}_1 := Up(f_1, a)$ and $\tilde{f}_2 := Up(f_2, a)$. Clearly, $f_1^h \leq f_2^h$ implies $\tilde{f}_1^h \leq \tilde{f}_2^h$. First, assume that $f_1^t \in C_P$, hence $f_1^t = f_2^t$. By applying the definition of C_P, we obtain

$$Tr(f_1^t, \mathcal{A}^s(a), f_1^h + \mathcal{A}^h(a)) = Tr(f_2^t, \mathcal{A}^s(a), f_2^h + \mathcal{A}^h(a)).$$

Thus, $\tilde{f}_1^t = \tilde{f}_2^t$. Note that the definitions of C_P and C_F imply that $\tilde{f}_1^t \subset C_P \cup C_F$ since $\tilde{f}_1^t \in R(f_1^t)$ and hence $\tilde{f}_1 \leq_C \tilde{f}_2$. Now, assume $f_1^t \in C_F$. Note that $R(f_1^t) \subset C_F$. Hence, if $\tilde{f}_1^t = \tilde{f}_2^t$, we obtain $\tilde{f}_1 \leq_C \tilde{f}_2$.

If instead $\tilde{f}_1^t \neq \tilde{f}_2^t$ we need to show that $\tilde{f}_1^t \to \tilde{f}_2^t$. Note that $f_2^t \in R(f_1^t)$ and $R(f_1^t)$ is traceable. Hence, $f_2^t, \tilde{f}_2^t$ and $\tilde{f}_1^t$ are on a directed path. Since $\mathcal{F}$ is acyclic and $f_2^t \Longrightarrow \tilde{f}_2^t$ holds there are four cases to consider:

1. $\tilde{f}_1^t \Longrightarrow f_2^t \Longrightarrow \tilde{f}_2^t$, which implies $\tilde{f}_1^t \Longrightarrow \tilde{f}_2^t$;
2. $f_2^t \Longrightarrow \tilde{f}_1^t \Longrightarrow \tilde{f}_2^t$, which implies $\tilde{f}_1^t \Longrightarrow \tilde{f}_2^t$;
3. $f_2^t \Longrightarrow \tilde{f}_2^t \to \tilde{f}_1^t$ and $f_1^t = \tilde{f}_1^t$, which creates the cycle $f_1^t \Longrightarrow f_2^t \Longrightarrow \tilde{f}_2^t \to \tilde{f}_1^t = f_1^t$ in $\mathcal{F}$ and is hence contradictory;

4. $f_2^t \Longrightarrow \tilde{f}_2^t \to \tilde{f}_1^t$ and $f_1^t \to \tilde{f}_1^t$. Note that $f_1^t \Longrightarrow f_2^t \to \tilde{f}_1^t$ and also that $Tr(f_1^t, \tilde{f}_1^t, \mathcal{A}^s(a), f_1^h + \mathcal{A}^h(a)) = 1$. Hence, we can apply the no-overtaking property for the edge $(f_1^t, \tilde{f}_1^t)$, the ticket f_2^t and the fare attribute $(\mathcal{A}^s(a), f_2^h + \mathcal{A}^h(a))$.
Since $f_2^h + \mathcal{A}^h(a) > f_1^h + \mathcal{A}^h(a)$, this yields the existence of an edge $(f_2^t, f^t) \in E$ with $Tr(f_2^t, f^t, \mathcal{A}^s(a), f_2^h + \mathcal{A}^h(a)) = 1$ and $\tilde{f}_1^t \Longrightarrow f^t$. Note that (f_2^t, f^t) is not necessarily the only outgoing edge at f_2^t but by definition of Tr there is only outgoing edge at f_2^t with transition function Tr equal to one for the fare attribute $(\mathcal{A}^s(a), f_2^h + \mathcal{A}^h(a))$. Since also $Tr(f_2^t, \tilde{f}_2^t, \mathcal{A}^s(a), f_2^h + \mathcal{A}^h(a)) = 1$, it follows that $\tilde{f}_2^t = f^t$. This creates the cycle $\tilde{f}_2^t \to \tilde{f}_1^t \to \tilde{f}_2^t$, which also yields a contradiction.

Hence, we conclude that in fact $\tilde{f}_1^t \to \tilde{f}_2^t$ and therefore, concluding the proof, $\tilde{f}_1 \leq_C \tilde{f}_2$. ◀

D Proof of Proposition 16

Proof. Consider $p \in P_{s,t}^{\pi} := \{\tilde{p} \text{ s,t-path} : \pi(f^t(\tilde{p})) = \pi^*\} \neq \emptyset$. If there is a path $p' \in P_{s,t}^f$ with $f^t(p') = f^t(p)$, we are done. If not, there is a path $p' \in P_{s,t}^f$ with $f^t(p') \to f^t(p)$. This implies $\pi(f^t(p')) \leq \pi(f^t(p))$ and hence that a path of the same price as p is present in $P_{s,t}^f$. ◀