# Efficient Algorithms for Fully Multimodal Journey Planning

**Moritz Potthoff**
Karlsruhe Institute of Technology (KIT), Germany

**Jonas Sauer**
Karlsruhe Institute of Technology (KIT), Germany

## Abstract

We study the journey planning problem for fully multimodal networks consisting of public transit and an arbitrary number of non-schedule-based transfer modes (e.g., walking, e-scooter, bicycle). Obtaining reasonable results in this setting requires multicriteria optimization, making the problem highly complex. Previous approaches were either limited to a single transfer mode or suffered from prohibitively slow running times. We establish a fully multimodal journey planning model that excludes undesirable solutions and can be solved efficiently. We extend existing efficient bimodal algorithms to our model and propose a new algorithm, HydRA, which enables even faster queries. On metropolitan and mid-sized country networks with walking and e-scooter as transfer modes, HydRA achieves query times of around 30 ms, which is fast enough for interactive applications.

**2012 ACM Subject Classification** Theory of computation → Shortest paths; Mathematics of computing → Graph algorithms; Applied computing → Transportation

**Keywords and phrases** Algorithms, Journey Planning, Multimodal, Multicriteria, Public Transit

**Digital Object Identifier** 10.4230/OASIcs.ATMOS.2022.14

**Supplementary Material** Source code is available at `https://github.com/kit-algo/ULTRA`.

**Funding** This research was funded by Deutsche Forschungsgemeinschaft under grant number WA 654/23-2.

## 1 Introduction

In modern transportation systems, passengers can choose from a wide variety of different transport modes, such as public transit, bike-sharing or e-scooters. Finding journeys that reasonably combine these modes requires multimodal journey planning algorithms. While efficient algorithms exist for each mode individually, the combined multimodal problem is much more challenging [2]. Existing solutions are either prohibitively slow or can only handle restricted scenarios (e.g., limiting the number of available modes). In this work we study journey planning in a fully multimodal network consisting of public transit plus an arbitrary number of non-schedule-based *transfer modes* (e.g., walking, cycling, e-scooter).

**Related Work.** State-of-the-art journey planning algorithms for public transit networks include RAPTOR [6], Connection Scan Algorithm [7], and Trip-Based Routing (TB) [16]. These algorithms typically Pareto-optimize two criteria: arrival time and the number of used public transit trips. While they support limited walking between nearby stations, they cannot be considered fully multimodal. It has been shown that incorporating an unrestricted transfer mode can significantly reduce travel times [15], but this comes at the cost of increased discomfort for the passenger. In order to capture this tradeoff, it is necessary to add a third criterion that measures the discomfort associated with using the transfer mode [11]. Without this criterion, interesting alternatives that avoid excessive use of the transfer mode will not be found [5]. When considering multiple transfer modes, a combined discomfort criterion


for all modes is not sufficient, since some modes may not always be available (e.g., rented bicycles), mode preferences vary between users, and users may have difficulties specifying their preferences precisely [5]. Hence, multimodal journey planning requires one discomfort criterion per transfer mode.

The most flexible multimodal multicriteria algorithm is MCR [5], an extension of RAPTOR. It supports an arbitrary number of transfer modes and criteria, but becomes prohibitively slow in complex scenarios. This is for two reasons: Firstly, the number of Pareto-optimal solutions exhibits superlinear growth in the number of criteria. Secondly, the transfer modes are explored using Dijkstra's algorithm [8], which is comparatively slow. Heuristics for MCR offer acceptable query speed but miss relevant journeys.

The first problem can be solved by computing the restricted Pareto set [4], which excludes uninteresting journeys from the full Pareto set and can be computed quickly with a variant of RAPTOR called BM-RAPTOR. The Dijkstra searches can be omitted by employing ULTRA [3], a speedup technique which precomputes *transfer shortcuts*. In its original form, it only supports bimodal networks (public transit plus one transfer mode) and two optimization criteria (arrival time and number of trips). McULTRA [11] additionally optimizes the time spent in the transfer mode as a third criterion. Multimodal restricted Pareto sets can be computed with UBM-RAPTOR, which integrates BM-RAPTOR and McULTRA. Even faster is UBM-TB, which replaces RAPTOR with McTB [11], an efficient three-criteria algorithm. This allows the bimodal problem to be solved in milliseconds even on country-sized networks.

**Contribution and Outline.** We extend the results for bimodal networks in [11] to a more general setting with an arbitrary number of transfer modes. Section 2 establishes basic notation and introduces the algorithms which our work builds on. In Section 3, we establish and discuss a realistic model for fully multimodal journey planning with an arbitrary number of transfer modes, which we call the *multimodal discomfort scenario*. In addition to arrival time and number of trips, we Pareto-optimize the time spent in each transfer mode as an individual criterion. To ensure reasonable results, we exclude certain types of undesirable solutions, such as journeys that switch between transfer modes in the middle of a transfer.

The multimodal discomfort scenario requires algorithms for an arbitrary number of criteria. To enable efficient queries, we incorporate McULTRA transfer shortcuts. In Section 4, we show that this can be done by running a three-criteria McULTRA shortcut computation for each transfer mode individually, which only requires linear preprocessing effort in the number of modes. We adapt existing query algorithms to our scenario in Section 5. This enables the use of ULTRA-McRAPTOR to compute full Pareto sets and UBM-RAPTOR for restricted Pareto sets. We do not adapt UBM-TB since there is no apparent way to extend it to more than three criteria. Instead, Section 6 introduces UBM-HydRA, which combines the advantages of RAPTOR and TB in scenarios with an arbitrary number of criteria. We evaluate the performance of our algorithms on real-world multimodal networks with walking and e-scooter as transfer modes in Section 7. On large metropolitan and mid-sized country networks, UBM-HydRA achieves query times of around 30 ms, which is faster than the state of the art by more than two orders of magnitude and enables interactive applications.

## 2 Preliminaries

Following the notation in [3, 13, 11], a multimodal network is a 5-tuple $(\mathcal{S}, \mathcal{T}, \mathcal{R}, G, \mathcal{F})$ consisting of a set of *stops* $\mathcal{S}$, a set of *trips* $\mathcal{T}$, a set of *routes* $\mathcal{R}$, a directed, weighted *transfer graph* $G = (\mathcal{V}, \mathcal{E})$, and a set of *free transfers* $\mathcal{F}$. A stop $v \in \mathcal{S}$ is a location where passengers

can board or disembark a vehicle. A trip $T = \langle \epsilon_0, \dots, \epsilon_k \rangle \in \mathcal{T}$ represents the ride of a vehicle as a sequence of *stop events*. Each stop event $\epsilon_i$ represents a visit of the vehicle at a stop $v(\epsilon_i) \in \mathcal{S}$ with *arrival time* $\tau_{\text{arr}}(\epsilon_i)$ and *departure time* $\tau_{\text{dep}}(\epsilon_i)$. If passengers are required to observe a *departure buffer time* before entering $T$ via $\epsilon_i$, this can be represented by reducing $\tau_{\text{dep}}(\epsilon_i)$ accordingly [17]. The $i$-th stop event of a trip $T$ is denoted by $T[i]$. The trips are partitioned into a set of routes $\mathcal{R}$ such that all trips of a route follow the same stop sequence and no trip overtakes another. The unrestricted transfer graph $G = (\mathcal{V}, \mathcal{E})$ consists of a set of *vertices* $\mathcal{V}$ with $\mathcal{S} \subseteq \mathcal{V}$, and a set of *edges* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. It can be traversed with one of $m$ different transfer modes. Traveling along an edge $e = (v, w)$ in mode $i$ requires the *transfer time* $\tau_{\mathsf{t}}(e, i)$. To ease notation, we represent all modes in a single graph instead of using one graph per mode. If an edge $e$ cannot be traversed with mode $i$, we set $\tau_{\mathsf{t}}(e, i) = \infty$. Additionally, the set $\mathcal{F} \subseteq \mathcal{S} \times \mathcal{S}$ contains transfers which are "free" in the sense that the time spent using them is not penalized via an optimization criterion. These represent short transfers between nearby stops, e.g., to connect platforms belonging to the same station, which are considered an unavoidable part of the public transit network. We require that $\mathcal{F}$ is transitively closed and fulfills the triangle inequality. We also refer to the set of free transfers as mode 0, but do not count it as one of the $m$ transfer modes. The transfer time for a free transfer $e \in \mathcal{F}$ is denoted by $\tau_{\mathsf{t}}(e, 0)$.

Given source and target vertices $s, t \in \mathcal{V}$, an *s-t-journey* represents the movement of a passenger from $s$ to $t$. A journey is an alternating sequence of *trip legs* (i.e., subsequences of trips) and *transfers* (i.e., free transfers or paths in the transfer graph). It begins with an *initial transfer* between $s$ and the first trip leg, and ends with a *final transfer* connecting the final trip leg to $t$. In between, trip legs are connected by *intermediate transfers*. Each transfer is associated with the mode in which it is traversed. Switching between modes within a transfer is not permitted. To represent the time overhead required by some modes (e.g., renting and returning an e-scooter), a *mode overhead* $\omega(i)$ is added to the travel time of each transfer in mode $i \neq 0$.

**Problem Statement.** An $s$-$t$-journey $J$ is evaluated with respect to $m + 2$ criteria: the arrival time $\tau_{\text{arr}}(J)$ at $t$, the number of used trips $|J|$, and for each transfer mode, the transfer time spent using that mode. A journey $J$ *weakly dominates* another journey $J'$ if $J$ is not worse than $J'$ according to any criterion. If $J$ is also strictly better than $J'$ in at least one criterion, we say that $J$ *strongly dominates* $J'$. For source and target vertices $s, t \in \mathcal{V}$ and a departure time $\tau_{\text{dep}}$, the objective is to compute a *Pareto set* of $s$-$t$-journeys that depart no earlier than $\tau_{\text{dep}}$. A full Pareto set $\mathcal{J}$ is a set of minimal size such that every feasible journey is weakly dominated by a journey in $\mathcal{J}$. An *anchor Pareto set* $\mathcal{J}_A$ is a Pareto set for the two criteria arrival time and number of trips. For each journey $J \in \mathcal{J}$, its *anchor journey* $A(J)$ is the journey in $\mathcal{J}_A$ with the highest number of trips not greater than $|J|$. Given a *trip slack* $\sigma_{\text{tr}} \geq 1$ and an *arrival slack* $\sigma_{\text{arr}} \geq 1$, the *restricted Pareto set* [4] is defined as

$$\mathcal{J}_R := \{J \in \mathcal{J} \mid |J| \leq |A(J)| \cdot \sigma_{\text{tr}} \text{ and } \tau_{\text{arr}}(J) - \tau_{\text{dep}} \leq (\tau_{\text{arr}}(A(J)) - \tau_{\text{dep}}) \cdot \sigma_{\text{arr}}\}.$$

This set contains all journeys from the full Pareto set whose arrival time and number of trips do not exceed their respective slack compared to their anchor journey. Following [11], the slacks are relative to the overall length of the journey rather than absolute values as in [4].

**Algorithms.** We conclude this section with an overview of the algorithms which our work builds on. RAPTOR [6] Pareto-optimizes the two criteria arrival time and number of trips in a public transit network with a transitively closed transfer graph. It operates in rounds,

where round $i$ finds journeys with $i$ trips by extending Pareto-optimal journeys with $i-1$ trips. Each round consists of two phases: The route scanning phase collects and scans all routes that visit stops which were updated in the previous round. This is followed by the transfer relaxation phase, which relaxes the outgoing transfer edges of all stops that were updated in the route scanning phase.

McRAPTOR [6] is a variant of RAPTOR that can optimize an arbitrary number of criteria. For each stop $v$ and round $i$, it maintains a *bag* $B^i(v)$ of labels representing Pareto-optimal journeys ending at $v$. Additionally, a *best bag* $B^*(v)$ contains all Pareto-optimal labels across all rounds. When a new label is found at a stop $v$ in round $i$, it is compared to $B^*(v)$. If it is not dominated, it is merged into $B^*(v)$ and $B^i(v)$. During the scan for a route $R$, the algorithm maintains a *route bag* $B_{\text{route}}$ of labels which represent journeys that end with a trip of $R$. Associated with each label $\ell \in B_{\text{route}}$ is its *active trip* $T(\ell)$, which is the trip of $R$ used by the corresponding journey. When the route scan visits a stop $v$, journeys exiting the route at $v$ are found by merging $B_{\text{route}}$ into $B^i(v)$. Then, for each label in $B^{i-1}(v)$, the algorithm finds the earliest trip $T$ that can be entered at $v$, creates a label with active trip $T$ and merges it into $B_{\text{route}}$.

MCR [5] extends McRAPTOR for multimodal networks, replacing the transfer relaxation phases with multicriteria Dijkstra searches on the unrestricted transfer graphs. For each mode $j$ and each vertex $v$, the algorithm maintains a *Dijkstra bag* $B^j_{\text{Dij}}(v)$. When a label is added at a stop $v$ in the route scanning phase, it is also merged into $B^j_{\text{Dij}}(v)$ and inserted into the Dijkstra priority queue. When the Dijkstra search adds a label to the Dijkstra bag $B^j_{\text{Dij}}(v)$ of a stop $v$ in round $i$, the label is also inserted into $B^i(v)$.

Restricted Pareto sets can be computed with BM-RAPTOR [4], which operates in three steps: First, a *forward pruning search* is run using two-criteria RAPTOR. This computes the anchor set and, for each stop $v$ and round $i$, an *earliest arrival time* $\overrightarrow{\tau_{\text{arr}}}(v,i)$. The *backward pruning search* performs one backward RAPTOR search per anchor journey in order to compute a *latest departure time* $\overleftarrow{\tau_{\text{dep}}}(v,i)$ per stop $v$ and round $i$. These are used for pruning by the McRAPTOR *main search*. Let $K$ denote the maximum number of trips among all anchor journeys. Any journey that arrives at a stop $v$ with $i$ trips later than $\overleftarrow{\tau_{\text{dep}}}(v, K \cdot \sigma_{\text{tr}} - i)$ is discarded because it cannot be extended to a journey that meets the slack requirements.

ULTRA [3, 13] enables public transit algorithms that normally require a transitively closed transfer graph to operate on a bimodal network with a single unrestricted transfer graph. To this end, it employs a preprocessing phase which computes *transfer shortcuts* representing all required intermediate transfers. This is done by enumerating journeys with at most two trips. Journeys with exactly two trips and no initial or final transfer are called *candidates*, while all journeys with at most two trips are called *witnesses*. If a candidate is not dominated by any witness, a shortcut representing its intermediate transfer is generated. The algorithm used to enumerate candidates and witnesses resembles performing an MCR search restricted to two rounds for each source stop $s \in \mathcal{S}$ and each possible departure time at $s$. Additional pruning rules are integrated to make the search more efficient. ULTRA can compute two varieties of shortcuts: *stop-to-stop* shortcuts connecting pairs of stops are sufficient for most query algorithms, while TB [3] requires *event-to-event* shortcuts between pairs of stop events. While ULTRA only optimizes arrival time and number of trips, McULTRA [11] additionally optimizes transfer time as a third criterion. A (Mc)ULTRA query explores initial and final transfers with Bucket-CH [10, 9], a technique for one-to-many searches on road networks. Then, a public transit algorithm of choice is run, using the precomputed shortcuts as the transfer graph. Integrating BM-RAPTOR with McULTRA yields UBM-RAPTOR [11], which computes restricted Pareto sets in a network with an unlimited transfer graph.

## 3 Multimodal Discomfort Scenario

To ensure that our algorithms compute reasonable solutions, we define the *multimodal discomfort scenario*. We assume that public transit is generally the fastest and most comfortable available mode. Its main disadvantage is limited availability in rural areas and outside of peak hours. The transfer modes can bridge gaps in poorly serviced areas, but using them incurs discomfort, either because they are cumbersome (e.g., walking) or costly (e.g., e-scooter, bike-sharing). Accordingly, passengers prefer to use public transit unless using a transfer mode improves the arrival time or reduces the number of trips. As discussed in Appendix A, this assumption excludes car-based modes. We capture the discomfort associated with a transfer mode by penalizing the time spent using it. This requires one additional criterion per mode. It is well known [5, 1, 4] that full Pareto sets for more than two criteria are extremely large and contain many uninteresting journeys which are small variations of other solutions. In practice, it is not sensible to show more than a few journeys to the user. This motivates the approach of defining a subset of the full Pareto set (e.g., the restricted Pareto set) which excludes some, but not necessarily all undesirable journeys and can be computed quickly. Relevant solutions can then be selected in a post-processing step.

Supporting multiple transfer modes introduces new types of undesirable journeys. Most crucially, allowing mode changes within a transfer would vastly increase the number of Pareto-optimal journeys. Already in a single transfer with two available modes, it is possible to switch between the two modes at any vertex along the transfer, and none of these options dominates the others. Besides bloating the Pareto set with nearly identical journeys, this would lead to an infeasibly high number of ULTRA transfer shortcuts. In practice, such journeys are not attractive because changing modes in the middle of a transfer is cumbersome. We therefore prohibit mode changes in order to remove uninteresting solutions from the Pareto set and enable ULTRA as a speedup technique. One pattern which could be considered a desirable mode change is walking between public transit stops and pickup/dropoff locations for rented vehicles, such as e-scooters or bicycles. We do not treat this as a mode change but rather a part of the overall e-scooter/bicycle transfer. Access and egress for these more complex modes can be modeled directly in the transfer graph. For our experiments, we assume that rented vehicles can be picked up and dropped off at any location. Access and egress are modeled via the mode overhead, which is added to every transfer in the respective mode. This prevents solutions with unrealistically short scooter or bicycle transfers.

Finally, we discuss the inclusion of the free transfer mode, which is intended for short, "unavoidable" transfers. These transfers are still penalized indirectly via the number of trips, but the (negligible) time spent using them is not counted towards the walking transfer time. Modeling these transfers as part of the walking mode would lead to nonsensical Pareto-optimal journeys, in which very short transfers are circumvented via detours to other stops where no such transfers are necessary.

## 4 Adapting McULTRA

MCR can be easily adapted for the multimodal discomfort scenario, as we will show in Section 5. In order to obtain faster algorithms, we omit the costly Dijkstra searches by integrating ULTRA. So far, the most general ULTRA variant is McULTRA [11], which supports one transfer mode and optimizes transfer time as a third criterion. A naive approach for generalizing this to the multimodal discomfort scenario would be to extend ULTRA to support an arbitrary number of criteria. To show that this is not necessary, consider a candidate $J^c$ processed by McULTRA. By definition, $J^c$ includes at most one transfer leg and

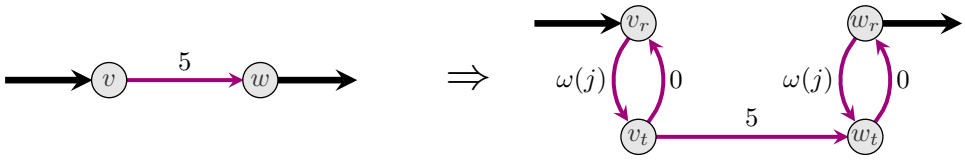


**Figure 1** Construction of the virtual transfer graph to represent mode overheads. Public transit trips are drawn in black, transfers of mode $j$ in purple.

therefore uses at most one transfer mode $i$. Witnesses with non-zero transfer time in any mode besides 0 or $i$ cannot dominate $J^{\mathsf{c}}$. This means we can decompose the preprocessing into one three-criteria McULTRA computation per transfer mode, making the overall preprocessing effort linear in $m$. The McULTRA computation for mode $i$ only considers transfers in modes 0 and $i$. It is guaranteed to find all relevant candidates, as well as all witnesses except those that use transfers of length 0 in a mode other than 0 or $i$. It is reasonable to assume that if a transfer leg of length 0 exists in any transfer mode, then a corresponding free transfer also exists in $\mathcal{F}$. If this is not the case, McULTRA will fail to find some witnesses and potentially generate superfluous shortcuts, but queries will remain correct.

To explore mode 0, McULTRA relaxes the outgoing free transfers of all updated stops before each Dijkstra search. Since $\mathcal{F}$ is already transitively closed, no stop-to-stop shortcuts need to be computed for it. Accordingly, journeys with free intermediate transfers are considered witnesses. Event-to-event shortcuts for $\mathcal{F}$ can be computed with two-criteria ULTRA using $(\mathcal{S}, \mathcal{F})$ as the transfer graph. Mode overheads are incorporated by constructing a virtual transfer graph, as shown in Figure 1. Each stop $v$ is split into a route vertex $v_r$ and a transfer vertex $v_t$. The two vertices are connected by directed edges $\overrightarrow{e_v} = (v_r, v_t)$ with $\tau_{\mathsf{t}}(\overrightarrow{e_v}, j) = \omega(j)$ and $\overleftarrow{e_v} = (v_t, v_r)$ with $\tau_{\mathsf{t}}(\overleftarrow{e_v}, j) = 0$. Each edge $(v, w) \in \mathcal{E}$ in the original transfer graph is replaced with an edge $(v_t, w_t)$ between the respective transfer vertices.

To ensure correctness, ULTRA requires that every query can be answered with a Pareto-optimal journey $J$ such that every candidate subjourney of $J$ is also Pareto-optimal. As shown by Figure 2, this is no longer the case if mode changes within a transfer are prohibited. In this example, the candidate $J^{\mathsf{c}}$ is dominated by a witness $J^{\mathsf{w}}$ that begins with an initial transfer in some mode $i$. However, adding a transfer in another mode $j \neq i$ as a prefix induces a mode change in $J^{\mathsf{w}}$, making it infeasible and leaving $J^{\mathsf{c}}$ as the only alternative. ULTRA will not generate a shortcut for the intermediate transfer of $J^{\mathsf{c}}$ and therefore fail to find journeys that include this transfer. However, note that this only affects journeys which would not be Pareto-optimal if mode changes were allowed. Since journeys with mode changes are undesirable, the same is true of the journeys which they dominate. Typically, such journeys include superfluous trip detours which only serve to circumvent the forbidden mode change. Hence, while ULTRA in the multimodal discomfort scenario cannot guarantee to find all Pareto-optimal journeys, the missed journeys are known to be undesirable.

## 5 RAPTOR-Based Query Algorithms

Existing McRAPTOR-based algorithms, namely MCR, ULTRA-McRAPTOR and UBM-RAPTOR, can be applied in the multimodal discomfort scenario with some minor changes. First, because mode changes within a transfer are prohibited, the pruning rules of McRAPTOR must be adjusted. Journeys that end with a trip may dominate journeys that end with a transfer, but not vice versa. This is because a transfer in mode $i$ cannot be followed by a transfer in any mode other than $i$, whereas a trip can always be followed by a

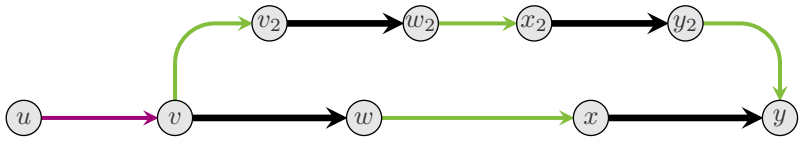


**Figure 2** An example where McULTRA misses a necessary shortcut in the multimodal discomfort scenario. Public transit trips are drawn in black, and transfers of two different modes in purple and green, respectively. Journey $J_1 = \langle u, v, w, x, y \rangle$ is optimal because the dominating journey $J_2 = \langle u, v, v_2, w_2, x_2, y_2, y \rangle$ includes a prohibited mode change at $v$. However, the candidate $J^{\mathsf{c}} = \langle v, w, x, y \rangle$ for the shortcut $(w, x)$ is dominated by the witness $J^{\mathsf{w}} = \langle v, v_2, w_2, x_2, y_2, y \rangle$.

transfer. McRAPTOR can take this into account by maintaining two bags per vertex $v$ and round $i$: a *trip bag* $B^i_{\text{trip}}(v)$ for labels ending with a trip, and a *transfer bag* $B^i_{\text{trans}}(v)$ for labels ending in a transfer. Accordingly, the algorithm also maintains two best bags $B^*_{\text{trip}}(v)$ and $B^*_{\text{trans}}(v)$ per vertex $v$. When a route scan in round $i$ generates a new label $\ell$ at a vertex $v$, it is compared to $B^*_{\text{trip}}(v)$, but not $B^*_{\text{trans}}(v)$. If $\ell$ is not dominated by $B^*_{\text{trip}}(v)$, it is merged into $B^i_{\text{trip}}(v)$. At the start of the transfer phase, $B^i_{\text{trip}}(v)$ is merged into $B^i_{\text{trans}}(v)$ for each updated stop $v$. This represents a direct transfer between trips at $v$ without using a transfer mode. Finally, when a label at a vertex $v$ is generated in the transfer phase, it is compared to both $B^*_{\text{trip}}(v)$ and $B^*_{\text{trans}}(v)$. If it is not dominated by either bag, it is merged into $B^i_{\text{trans}}(v)$.

Incorporating free transfers and mode overheads is straightforward. Since the set $\mathcal{F}$ of free transfers is transitively closed, no Dijkstra or Bucket-CH searches are required. It can be explored simply by relaxing the outgoing transfers of all updated stops in each round, as done by McRAPTOR. In ULTRA-based queries, the shortcuts already include the overheads. For the initial and final transfers, they are added when evaluating the results of the Bucket-CH searches. In MCR, the overhead is added when the Dijkstra searches are initialized: After inserting a label into the trip bag $B^i_{\text{trip}}(v)$ of a stop $v$ in round $i$, the mode overhead $\omega(j)$ is added before merging the label into the Dijkstra bag $B^j_{\text{Dij}}(v)$ for mode $j$.

Computing restricted Pareto sets requires adapting the pruning searches of UBM-RAPTOR to multiple transfer modes. Since they only optimize arrival time and number of trips, a transfer is always traversed with the fastest available mode. Due to overheads and limited availability of some modes, this is not necessarily the same mode for all transfers. The pruning searches therefore identify the fastest mode for each transfer individually. For initial and final transfers, this is done by merging the results of the Bucket-CH searches for each mode, choosing the minimum distance for each stop. For the intermediate transfers, the shortcut sets of all modes are merged, keeping the shortest shortcut in case of duplicates.

## 6 HydRA

In a bimodal scenario, event-to-event shortcuts enable the use of McTB, which is faster than RAPTOR. McTB avoids maintaining Pareto sets by representing arrival time and number of trips implicitly, leaving transfer time as the only remaining criterion. In the multimodal discomfort scenario with an arbitrary number of criteria, this approach is no longer applicable. We therefore propose HydRA (Hybrid Routing Algorithm), a new algorithm which is based on McRAPTOR but incorporates some aspects of McTB. In particular, it uses event-to-event shortcuts to reduce the search space and performs simpler, more cache-efficient route scans. Since HydRA is intended for scenarios with four or more criteria, where full Pareto sets are impractically large, we only design a variant for restricted Pareto sets, called UBM-HydRA.

Like UBM-TB [11], UBM-HydRA uses two-criteria TB to perform the pruning searches. For each trip $T$ and round $i$, these compute a *forward reached index* $\overrightarrow{r}(T,i)$ and a *backward reached index* $\overleftarrow{r}(T,i)$. The forward reached index $\overrightarrow{r}(T,i)$ indicates the index of the first stop along $T$ that is reachable from $s$ with $i$ trips. The backward reached index $\overleftarrow{r}(T,i)$ is the index of the last stop along $T$ which can be entered such that $t$ is reachable with $i$ additional trips and without exceeding the slack of the respective anchor journey. Additionally, for each stop $v$ and round $i$, the forward search computes the earliest arrival time $\overrightarrow{\tau_{\text{arr}}}(v,i)$ among all journeys that arrive at $v$ via a trip and use $i$ trips. Unlike the UBM-TB backward search, the UBM-HydRA backward search also computes an analogous latest departure time $\overleftarrow{\tau_{\text{dep}}}(v,i)$.

The main search is based on McRAPTOR but incorporates event-to-event shortcuts. Initial and final transfers are handled as in ULTRA-McRAPTOR. Route scans in round 0 are mostly unchanged but incorporate the backward reached indices for pruning. Consider a label $\ell$ with active trip $T(\ell)$ that is generated when entering a route $R$ at its $j$-th stop. If $\overleftarrow{r}(T(\ell), K \cdot \sigma_{\text{tr}}) < j$ holds, $\ell$ cannot be extended to an $s$-$t$-journey without exceeding the slack values, so it is discarded. When $\ell$ exits the route at a stop $v$ with index $k$, the label that is merged into the trip bag of $v$ stores its *exit event* $T(\ell)[k]$. Transfers in round $i$ are explored as follows. For each updated stop $v$ and each newly added label with exit event $\epsilon$ in $B^i_{\text{trip}}(v)$, all outgoing shortcuts of $\epsilon$ are relaxed. For each shortcut $(\epsilon, \epsilon')$, a new label $\ell$ is created which stores $\epsilon'$ as its *entry event*. As in UBM-RAPTOR, $\ell$ is discarded if its arrival time exceeds $\overleftarrow{\tau_{\text{dep}}}(v(\epsilon'), K \cdot \sigma_{\text{tr}} - i)$. Otherwise, it is merged into $B^i_{\text{trans}}(v(\epsilon'))$. The dominance rules of both trip and transfer bags are adjusted: if two labels are equivalent in all criteria but have different exit/entry events, both are kept.

The route scans for all rounds $i > 0$ make use of the computed entry events. For each updated stop $v$ and each newly added label with entry event $T[j]$ in $B^{i-1}_{\text{trans}}(v)$, $T$ is scanned. The last stop index where $T$ can be entered without exceeding the slack is $k := \overleftarrow{r}(T, K \cdot \sigma_{\text{tr}} - i)$. Accordingly, $T$ can be exited at all stop indices $x$ with $j + 1 \leq x \leq k + 1$. For each such stop index $x$, a new label with exit event $T[x]$ is created and merged into the trip bag of $v(T[x])$.

**Shortcut Augmentation.** To compute correct reached indices, the TB pruning searches replace the event-to-event shortcuts $\mathcal{E}^{\mathsf{t}}$ with a set $\mathcal{E}^{\mathsf{t}}_{\text{aug}}$ of *augmented shortcuts*. We briefly restate a simplified version of the augmentation step introduced in [11]: For two trips $T_1, T_2$ of the same route $R \in \mathcal{R}$, we write $T_1 \preceq T_2$ if $\tau_{\text{arr}}(T_1[i]) \leq \tau_{\text{arr}}(T_2[i])$ for every index $i$ along $R$. Trips from different routes are not comparable via $\preceq$. An augmented shortcut $(T_a[i], T_b[j])$ is added to $\mathcal{E}^{\mathsf{t}}_{\text{aug}}$ if there is a shortcut $(T_c[i], T_b[j]) \in \mathcal{E}^{\mathsf{t}}$ with $T_c \succeq T_a$. Especially for fast transfer modes, this augmented shortcut set can become impractically large. We therefore propose a *limited shortcut augmentation* step: Let $T_c$ be the trip directly succeeding $T_a$ in the respective route. Then an augmented shortcut $(T_a[i], T_b[j])$ is inserted if $(T_c[i], T_b[j]) \in \mathcal{E}^{\mathsf{t}}$. Even later trips $T_d$ of the route are not checked for potential shortcuts. Thus, if a Pareto-optimal journey $J$ uses the shortcut $(T_d[i], T_b[j])$ and the pruning search reaches $T_a[i]$, it may fail to enter $T_b$ at $j$. However, since $T_d[i]$ has a significantly higher arrival time than $T_a[i]$, $J$ is likely to exceed the arrival slack of its anchor journey. Therefore, we expect the error caused by limiting the shortcut augmentation to be very small.

Furthermore, we add a filtering step that removes superfluous shortcuts which are created by the augmentation procedure as introduced in [11]. A shortcut $(T_a[i], T_b[j])$ is superfluous if there is another shortcut $(T_a[k], T_d[\ell])$ such that $k \geq i$, $T_d \preceq T_b$ and $\ell \leq j$. Consider a TB pruning search with reached indices $r(\cdot)$. For each trip $T$, the search upholds the invariant that $r(T') \leq r(T)$ for all $T' \succeq T$. If the search reaches $T_a[i]$, it will also reach $T[k]$ and relax its outgoing shortcuts, including $(T_a[k], T_d[\ell])$. Thus, $T_d[\ell]$ is entered and $r(T_b) \leq r(T_d) \leq \ell \leq j$ will hold after the search.

**Table 1** Sizes of the multimodal networks, including public transit, free transfers, unrestricted transfer graphs, and transitive transfer graphs for evaluating the solution quality of McULTRA.

| | London | Switzerland | Stuttgart |
|---|---|---|---|
| Stops | 19 682 | 25 125 | 13 584 |
| Routes | 1 955 | 13 786 | 12 351 |
| Trips | 114 508 | 350 006 | 91 304 |
| Stop events | 4 508 644 | 4 686 865 | 1 561 972 |
| Free transfers | 42 928 | 12 806 | 37 383 |
| Vertices | 181 642 | 603 691 | 1 166 604 |
| Unrestricted edges | 575 364 | 1 853 260 | 3 682 232 |
| Transitive edges (Walking) | 3 212 206 | 2 639 402 | 1 369 928 |
| Transitive edges (Scooter) | 2 374 294 | 2 432 366 | 1 558 234 |

## 7 Experiments

All algorithms were implemented in C++17 compiled with GCC 10.3.0 and optimization flag -O3. Shortcut computations were run on a machine with two 64-core AMD Epyc Rome 7742 CPUs clocked at 2.25 GHz, with a turbo frequency of 3.4 GHz, 1024 GiB of DDR4-3200 RAM, and 256 MiB of L3 cache. All other experiments were conducted on a machine with two 8-core Intel Xeon Skylake SP Gold 6144 CPUs clocked at 3.5 GHz, with a turbo frequency of 4.2 GHz, 192 GiB of DDR4-2666 RAM, and 24.75 MiB of L3 cache.

**Networks.** We evaluated our algorithms on multimodal networks representing Switzerland, Greater London and the greater region of Stuttgart, which were previously used to evaluate ULTRA [3, 13] and McULTRA [11]. An overview of the networks is given in Table 1. The public transit networks and free transfers for London and Switzerland networks were sourced from Transport for London[1] and a publicly available GTFS feed[2], respectively. The Stuttgart network was introduced in [14] and is based on proprietary data. To generate its free transfers, we connected all stops within a geographical distance of up to 400 m and computed the transitive closure. Unrestricted transfer graphs were taken from OpenStreetMap[3], following the methodology in [3, 13, 11]. We used walking and e-scooter as the available transfer modes, assuming a constant speed of 4.5 km/h for walking and 15 km/h for scooter. We chose mode overheads of 0 s for walking and 300 s for scooter. To evaluate the solution quality of McULTRA, we compared it to using transitively closed intermediate transfer graphs, which we created using the methodology described in [15]: We connected all pairs of stops whose transfer time lies below a certain threshold with an edge and then computed the transitive closure. As thresholds, we chose 9 min of walking and 3 min of scooter time for Stuttgart and Switzerland, and 4 min of walking and 80 s of scooter time for London.

**Preprocessing.** Table 2 reports the results of the McULTRA shortcut computation, using the same settings as in [11]. Because many short transfers are now covered by the free transfer mode, the number of shortcuts per mode is slightly lower than in the bimodal setting.

[1] `https://data.london.gov.uk`
[2] `https://gtfs.geops.ch/`
[3] `https://download.geofabrik.de/`

**Table 2** Multimodal McULTRA shortcut computation results. Times are formatted as h:mm:ss.

| Network | Variant | Free | | Walking | | Scooter | |
|---|---|---|---|---|---|---|---|
| | | Time | # Shortcuts | Time | # Shortcuts | Time | # Shortcuts |
| London | Stop | – | – | 0:22:45 | 115 036 | 2:22:33 | 8 163 962 |
| | Event | 0:00:27 | 10 404 923 | 0:29:33 | 11 422 382 | 2:36:58 | 126 877 333 |
| Switzerland | Stop | – | – | 0:05:58 | 214 872 | 0:15:35 | 1 063 575 |
| | Event | 0:00:09 | 5 884 998 | 0:07:09 | 13 193 976 | 0:17:44 | 26 446 770 |
| Stuttgart | Stop | – | – | 0:03:46 | 110 199 | 0:09:57 | 739 022 |
| | Event | 0:00:07 | 4 151 859 | 0:04:28 | 4 203 492 | 0:11:20 | 9 552 810 |

**Table 3** Number $|\overrightarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}|$ of augmented forward shortcuts and $|\overleftarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}|$ of augmented backward shortcuts. All values given in millions of shortcuts. *Lim.* refers to limited shortcut augmentation.

| Network | Lim. | Free | | Walking | | Scooter | |
|---|---|---|---|---|---|---|---|
| | | $\lvert\overrightarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ | $\lvert\overleftarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ | $\lvert\overrightarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ | $\lvert\overleftarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ | $\lvert\overrightarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ | $\lvert\overleftarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ |
| London | ○ | 33 | 36 | 31 | 36 | 1 545 | 1 643 |
| | ● | – | – | – | – | 223 | 220 |
| Switzerland | ○ | 13 | 14 | 44 | 44 | 157 | 155 |
| Stuttgart | ○ | 8 | 8 | 11 | 11 | 46 | 46 |

The preprocessing times are slightly higher because exploring the free transfer mode requires additional time and incorporating the mode overheads increases the size of the network. The number of augmented event-to-event shortcuts is listed in Table 3. As reported in [11], the augmentation time is negligible compared to the shortcut computation time. For e-scooters on the London network, the preprocessing time and the number of shortcuts are extremely high. This is caused by the high number of Pareto-optimal labels per vertex bag. To reduce this number, we implemented a heuristic version of McULTRA that discretizes the transfer time criterion into *buckets* for the purpose of testing dominance. Let $\tau_{\mathsf{t}}$ be the transfer time of a label. Instead of using $\tau_{\mathsf{t}}$ as the criterion for testing dominance, we use $\lfloor \frac{\tau_{\mathsf{t}}}{x} \rfloor$, where $x$ is the bucket size. For our experiments, we set $x = 300\,\text{s}$. As shown in Table 4, discretization reduces the preprocessing time and the number of event-to-event shortcuts by a factor of 3.

**Table 4** Impact of transfer time discretization on the McULTRA shortcut computation for e-scooters on the London network. Times are formatted as h:mm:ss. $|\overrightarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}|$ and $|\overleftarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}|$ are the number of augmented forward and backward shortcuts, respectively, measured in millions of shortcuts.

| Variant | Disc. | Time | # Shortcuts | Full aug. | | Limited aug. | |
|---|---|---|---|---|---|---|---|
| | | | | $\lvert\overrightarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ | $\lvert\overleftarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ | $\lvert\overrightarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ | $\lvert\overleftarrow{\mathcal{E}^{\mathsf{t}}_{\text{aug}}}\rvert$ |
| Stop | ○ | 2:22:33 | 8 163 962 | – | – | – | – |
| Stop | ● | 0:43:33 | 3 971 836 | – | – | – | – |
| Event | ○ | 2:36:58 | 126 877 333 | 1 545 | 1 643 | 223 | 220 |
| Event | ● | 0:48:03 | 48 942 757 | 585 | 637 | 84 | 82 |

**Table 5** Coverage of the exact Pareto sets by our algorithms for 10 000 random queries. ULTRA-McRAPTOR is compared to the full Pareto set, all others to the restricted Pareto set. For each metric, we report the mean coverage across all queries and the coverage of the 5th percentile. *Disc.* refers to shortcut discretization.

| Network | Algorithm | Disc. | Exact coverage [%] | | Fuzzy coverage [%] | |
|---|---|---|---|---|---|---|
| | | | 5th perc. | Mean | 5th perc. | Mean |
| London | ULTRA-McRAPTOR | ○ | 97.29 | 99.47 | 99.99 | 99.97 |
| | BM-RAPTOR | ○ | 63.63 | 93.65 | 92.68 | 98.61 |
| | UBM-RAPTOR | ○ | 100.00 | 99.74 | 100.00 | 99.95 |
| | UBM-RAPTOR | ● | 93.02 | 98.93 | 99.98 | 99.94 |
| | UBM-HydRA | ○ | 94.11 | 99.17 | 99.99 | 99.98 |
| | UBM-HydRA | ● | 82.90 | 97.21 | 99.80 | 99.93 |
| Switzerland | ULTRA-McRAPTOR | ○ | 93.01 | 98.70 | 99.35 | 99.77 |
| | BM-RAPTOR | ○ | 53.22 | 91.35 | 81.37 | 97.14 |
| | UBM-RAPTOR | ○ | 95.13 | 99.28 | 99.98 | 99.94 |
| | UBM-HydRA | ○ | 93.32 | 99.04 | 99.95 | 99.92 |
| Stuttgart | ULTRA-McRAPTOR | ○ | 83.09 | 96.39 | 94.34 | 99.06 |
| | BM-RAPTOR | ○ | 52.85 | 91.74 | 81.43 | 97.29 |
| | UBM-RAPTOR | ○ | 100.00 | 99.52 | 100.00 | 99.96 |
| | UBM-HydRA | ○ | 69.98 | 96.16 | 99.95 | 99.78 |

**Result Coverage.** As shown in Section 4, McULTRA-based queries fail to find some Pareto-optimal journeys which are dominated by journeys with mode changes. We already argued that these journeys are undesirable. Nevertheless, we analyze the impact that their exclusion has on the computed results by evaluating how well the exact Pareto set is covered by our algorithms. We consider two coverage metrics. Exact coverage is the percentage of journeys in the (full or restricted) Pareto set that are found by the algorithm, whereas fuzzy coverage also accounts for similarity between journeys. If the algorithm does not find a journey $J$ but another journey $J'$ that is almost as good or better in all criteria, we consider $J$ well covered. As in [5], we measure similarity using fuzzy logic. Given two parameters $\chi \in (0,1)$ and $\varepsilon > 0$, the fuzzy coverage of a journey $J$ by another journey $J'$ for a criterion $c$ is defined as

$$\operatorname{cov}(J, J') := \begin{cases} \exp\left(\frac{\ln(\chi)}{\varepsilon^2}\left(c(J) - c(J')\right)^2\right) & \text{if } c(J) < c(J') \\ 1 & \text{else.} \end{cases}$$

The overall fuzzy coverage $\operatorname{cov}(J_1, J_2)$ is the minimum coverage across all criteria. The fuzzy coverage of a journey $J$ by a set of journeys $\mathcal{J}$ is $\operatorname{cov}(J, \mathcal{J}) := \max_{J' \in \mathcal{J}} \operatorname{cov}(J, J')$. Finally, the fuzzy coverage $\operatorname{cov}(\mathcal{J}, \mathcal{J}')$ of a set of journeys $\mathcal{J}$ by another set of journeys $\mathcal{J}'$ is the mean coverage by $\mathcal{J}'$ across all journeys in $\mathcal{J}$. Following [5], the fuzzy parameters $(\chi, \varepsilon)$ are set to $(0.8, 60\,\text{s})$ for arrival time, $(0.1, 1)$ for number of trips, and $(0.8, 300\,\text{s})$ for transfer time.

Coverage results are reported in Table 5. Limited shortcut augmentation did not affect the results in any of our experiments. For full Pareto sets, ULTRA-McRAPTOR achieves an exact coverage above 96% and a fuzzy coverage above 99%. For restricted Pareto sets, UBM-RAPTOR achieves an exact coverage above 99% on all networks and nearly perfect fuzzy coverage. UBM-HydRA exhibits slightly lower coverage because it uses event-to-event shortcuts, which are more fine-grained and therefore more prone to missing journeys. In

**Table 6** Query performance for full Pareto sets, averaged over 10 000 random queries. *Rnd.* is the number of performed rounds, while *Jrn.* refers to the number of computed journeys.

| Network | Algorithm | Rnd. | Jrn. | Time [ms] | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Routes | Transfers | | | Total |
| | | | | | Free | Walking | Scooter | |
| London | MCR | 10.9 | 130.1 | 307.6 | 89.3 | 1 235.2 | 1 943.8 | 3 600.6 |
| | ULTRA-McRAPTOR | 10.9 | 129.4 | 218.6 | 91.3 | 126.0 | 2 602.4 | 3 052.6 |
| Switzerland | MCR | 18.5 | 209.4 | 1 785.3 | 101.4 | 3 874.0 | 4 462.8 | 10 264.0 |
| | ULTRA-McRAPTOR | 18.5 | 206.2 | 1 641.6 | 119.2 | 449.4 | 1 528.4 | 3 775.0 |
| Stuttgart | MCR | 12.0 | 215.3 | 1 249.2 | 343.3 | 6 605.8 | 8 259.3 | 16 481.3 |
| | ULTRA-McRAPTOR | 12.0 | 206.9 | 1 005.3 | 414.9 | 579.1 | 2 686.6 | 4 701.4 |

stop-to-stop ULTRA, even if a candidate is missed, its shortcut is often represented by another candidate which is found. This is less likely in the event-to-event variant. Still, on London and Switzerland the exact coverage remains above 99% and the fuzzy coverage is barely affected. The values are slightly lower for Stuttgart, but the fuzzy coverage remains extremely high at 99.8%. Altogether, these results justify of our choice of prohibiting mode changes in order to reduce the number of irrelevant solutions. While doing so introduces some new, undesirable Pareto-optimal solutions, they are often discarded by McULTRA and our experiments show that they are rare and well covered by other, more relevant solutions.

Another possibility to reduce the Pareto set and speed up queries would be to limit the length of intermediate transfers. This would enable the use of a transitively closed transfer graph and remove the need for a preprocessing step such as ULTRA. To demonstrate that this negatively impacts the solution quality, we evaluated the coverage of BM-RAPTOR using transitive intermediate transfers but unlimited initial and final transfers. The exact coverage is still above 90% because most optimal journeys do not include long intermediate transfers. Still, we observe a significant number of optimal journeys with long intermediate transfers which are not well covered by other solutions with limited transfers.

On the London network, discretizing transfer time when testing dominance significantly reduced the preprocessing time and the number of shortcuts. As expected, this noticeably reduces the exact coverage, although it remains much higher than with transitive intermediate transfers. The fuzzy coverage, however, remains excellent because missing shortcuts caused by discretization are guaranteed to have similar alternatives.

**Query Performance.** We now evaluate the running times of our query algorithms, beginning with ULTRA-McRAPTOR for full Pareto sets in Table 6. On Switzerland and Stuttgart, ULTRA-McRAPTOR achieves a speedup of 3–4 over MCR. Since the performance gain of ULTRA comes from speeding up the transfer phases, this is higher than the speedup of 2 observed in the bimodal scenario, where the transfer phase takes up a smaller share of the running time. In the scooter mode, the speedup is limited due to the high number of stop-to-stop shortcuts. For London, relaxing scooter shortcuts is in fact slower than a Dijkstra search, causing the overall speedup to be marginal. Overall, the results demonstrate that computing full Pareto sets is not practical due to the extremely high number of Pareto-optimal journeys.

We therefore investigate the performance for restricted Pareto sets, which is shown in Table 7. The number of computed journeys is reduced to less than 30, which is manageable for the algorithm but still more than can be shown to users. On Switzerland and Stuttgart,

**Table 7** Query performance for restricted Pareto sets with slack values $\sigma_{\text{arr}} = \sigma_{\text{tr}} = 1.25$, averaged over 10 000 random queries. *Rnd.* is the number of performed rounds, while *Jrn.* refers to the number of computed journeys. Disc. refers to shortcut discretization, *Lim.* to limited shortcut augmentation.

| Network | Algorithm | Disc. | Lim. | Rnd. | Jrn. | Time [ms] | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Forward | Backward | Main | Total |
| London | UBM-RAPTOR | ○ | ○ | 2.2 | 25.4 | 60.3 | 12.3 | 41.3 | 113.8 |
| | UBM-RAPTOR | ● | ○ | 2.2 | 25.4 | 33.9 | 7.3 | 26.1 | 67.3 |
| | UBM-HydRA | ○ | ○ | 2.2 | 25.1 | 43.6 | 25.8 | 11.2 | 80.6 |
| | UBM-HydRA | ○ | ● | 2.2 | 25.1 | 12.5 | 5.3 | 9.2 | 27.1 |
| | UBM-HydRA | ● | ● | 2.2 | 24.9 | 9.3 | 3.6 | 7.4 | 20.3 |
| Switzerland | UBM-RAPTOR | ○ | ○ | 3.5 | 27.0 | 28.8 | 4.3 | 22.5 | 55.6 |
| | UBM-HydRA | ○ | ○ | 3.5 | 26.9 | 18.1 | 2.6 | 9.8 | 30.5 |
| Stuttgart | UBM-RAPTOR | ○ | ○ | 2.6 | 18.0 | 22.2 | 5.4 | 22.0 | 49.5 |
| | UBM-HydRA | ○ | ○ | 2.6 | 17.5 | 14.3 | 3.7 | 9.8 | 27.7 |

UBM-RAPTOR achieves a speedup of more than two orders of magnitude over MCR. On London, the speedup is only 32, again due to the high number of scooter shortcuts.

HydRA significantly speeds up the main search due to its more efficient route scans. Additionally, by using the more fine-grained event-to-event shortcuts, the number of explored shortcuts per label is significantly reduced. For Switzerland and Stuttgart, the pruning searches are also around 50% faster because they use TB instead of RAPTOR. On London, this is not the case with full augmentation because the number of shortcuts becomes extremely high. Limited augmentation solves this problem, improving the speedup over UBM-RAPTOR to 4 without affecting the computed results. Shortcut discretization further improves the query times by 41% for UBM-RAPTOR and 25% for UBM-HydRA, at the cost of a slight loss in solution quality. Overall, the speedup of UBM-HydRA over MCR ranges from around 130 for London, where MCR performs the best, to 600 for Stuttgart. With query times of around 30 ms, the performance is good enough for interactive applications.

## 8 Conclusion

We developed journey planning algorithms for fully multimodal networks with an arbitrary number of transfer modes. To ensure reasonable results, we established the multimodal discomfort scenario, which optimizes one discomfort criterion per transfer mode and prohibits mode changes within a transfer. We showed that McULTRA can be adapted to this scenario in a scalable fashion by preprocessing each mode independently. Besides adapting existing query algorithms, we proposed HydRA, which carries over the advantages of TB into a setting with an arbitrary number of criteria. Our experimental evaluation shows that our algorithms achieve query times which are fast enough for interactive applications. Future work could involve incorporating more complex transfer modes such as bike-sharing, which require additional modelling [12]. Furthermore, HydRA is a promising candidate for efficiently solving journey planning problems with other criteria, such as fare or vehicle occupancy.

## References


1 Hannah Bast, Mirko Brodesser, and Sabine Storandt. Result Diversity for Multi-Modal Route Planning. In *Proceedings of the 13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'13)*, volume 33 of *OpenAccess Series in Informatics (OASIcs)*, pages 123–136. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi:10.4230/OASIcs.ATMOS.2013.123.

2 Hannah Bast, Daniel Delling, Andrew 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 (LNCS)*, pages 19–80. Springer, 2016. doi:10.1007/978-3-319-49487-6_2.

3 Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. UnLimited TRAnsfers for Multi-Modal Route Planning: An Efficient Solution. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA'19)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.14.

4 Daniel Delling, Julian Dibbelt, and Thomas Pajor. Fast and Exact Public Transit Routing with Restricted Pareto Sets. In *Proceedings of the 21st Workshop on Algorithm Engineering and Experiments (ALENEX'19)*, pages 54–65. Society for Industrial and Applied Mathematics (SIAM), 2019. doi:10.1137/1.9781611975499.5.

5 Daniel Delling, Julian Dibbelt, Thomas Pajor, Dorothea Wagner, and Renato F. Werneck. Computing Multimodal Journeys in Practice. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, volume 7933 of *Lecture Notes in Computer Science (LNCS)*, pages 260–271. Springer, 2013. doi:10.1007/978-3-642-38527-8_24.

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

7 Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Connection Scan Algorithm. *Journal of Experimental Algorithmics (JEA)*, 23:1.7:1–1.7:56, 2018. doi:10.1145/3274661.

8 Edsger W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.

9 Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transportation Science*, 46:388–404, 2012. doi:10.1287/trsc.1110.0401.

10 Sebastian Knopp, Peter Sanders, Dominik Schultes, Frank Schulz, and Dorothea Wagner. Computing Many-to-Many Shortest Paths Using Highway Hierarchies. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX'07)*, pages 36–45. Society for Industrial and Applied Mathematics (SIAM), 2007. doi:10.5555/2791188.2791192.

11 Moritz Potthoff and Jonas Sauer. Fast Multimodal Journey Planning for Three Criteria. In *Proceedings of the 24th Workshop on Algorithm Engineering and Experiments (ALENEX'22)*, pages 145–157. Society for Industrial and Applied Mathematics (SIAM), 2022. doi:10.1137/1.9781611977042.12.

12 Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. Faster Multi-Modal Route Planning with Bike Sharing using ULTRA. In *Proceedings of the 18th International Symposium on Experimental Algorithms (SEA'20)*, volume 160 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SEA.2020.16.

13 Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. Integrating ULTRA and Trip-Based Routing. In *Proceedings of the 20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'20)*, volume 85 of *OpenAccess Series in Informatics (OASIcs)*, pages 4:1–4:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/OASIcs.ATMOS.2020.4.

14 Johannes Schlaich, Udo Heidl, and Regine Pohlner. Verkehrsmodellierung für die Region Stuttgart – Schlussbericht. Unpublished manuscript, 2011.

15 Dorothea Wagner and Tobias Zündorf. Public Transit Routing with Unrestricted Walking. In *Proceedings of the 17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'17)*, volume 59 of *OpenAccess Series in Informatics (OASIcs)*, pages 7:1–7:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/OASIcs.ATMOS.2017.7.

16 Sascha Witt. Trip-Based Public Transit Routing. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA'15)*, volume 9294 of *Lecture Notes in Computer Science (LNCS)*, pages 1025–1036. Springer, 2015. doi:10.1007/978-3-662-48350-3_85.

17 Tobias Zündorf. *Multimodal Journey Planning and Assignment in Public Transportation Networks*. PhD thesis, Karlsruhe Institute of Technology, 2020. doi:10.5445/IR/1000145076.


## A Car-Based Modes

Car-based modes such as park and ride (i.e., using a private car for the first or last leg of a journey), taxi or car-sharing do not fit the assumption of the multimodal discomfort scenario that public transit is the fastest and most comfortable mode. Since these modes are generally faster than using public transit, the solution that optimizes arrival time and number of trips is usually a direct car journey. Since usage of the fastest mode is now penalized, interrupting it in order to reduce car time always leads to a non-dominated solution. This causes a combinatorial explosion in the number of optimal journeys, as observed previously by Delling et al. [5] when including taxis in a multimodal network. The resulting journeys tend to be undesirable combinations of long car rides and short public transit detours. In practice, users are either willing to use a car for the entire journey (which yields a unimodal journey planning problem) or they are only willing to use it in a very limited capacity. In the latter case, there is no need for techniques designed for unrestricted transfer modes, such as ULTRA. In fact, because using a car incurs a large time overhead (e.g., for hailing a ride), short intermediate car transfers are typically not useful. Thus, cars should not be considered a transfer mode but rather an access/egress mode that connects the source and target vertex to the public transit network. Since such modes can be handled with existing techniques, we consider them out of scope for this work.