Karlsruher Institut für Technologie

Karlsruhe Reports in Informatics 2012,20
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

Computing and Evaluating Multimodal Journeys

Daniel Delling, Julian Dibbelt, Thomas Pajor, Dorothea Wagner,
and Renato F. Werneck

2012

KIT – University of the State of Baden-Wuerttemberg and National
Research Center of the Helmholtz Association

Fakultät für **Informatik**

Computing and Evaluating Multimodal Journeys*

Daniel Delling[1], Julian Dibbelt[2], Thomas Pajor[2], Dorothea Wagner[2], and Renato F. Werneck[1]

[1] Microsoft Research Silicon Valley, 1065 La Avenida, Mountain View CA 94043.
{dadellin,renatow}@microsoft.com
[2] Karlsruhe Institute of Technology (KIT), P. O. Box 6980, 76128 Karlsruhe, Germany.
{dibbelt,pajor,wagner}@kit.edu

Abstract. We study the problem of finding multimodal journeys in transportation networks, including unrestricted walking, driving, cycling, and schedule-based public transportation. A natural solution to this problem is to use multicriteria search, but it tends to be slow and to produce too many journeys, several of which are of little value. We propose algorithms to compute a full Pareto set and then score the solutions in a postprocessing step using techniques from fuzzy logic, quickly identifying the most significant journeys. We also propose several (still multicriteria) heuristics to find similar journeys, but much faster. Our experiments show that this approach enables the computation of high-quality multimodal journeys on large metropolitan areas, and is fast enough for practical applications.

1 Introduction

Online services for journey planning have become a commodity used daily by millions of commuters. The problem of efficiently computing good journeys in transportation networks presents several algorithmic challenges, and has been an active area of research in recent years. Much focus has been given to the computation of routes both in road networks [1, 16, 20, 32, 36, 49] and in scheduled-based public transit [7, 9, 12, 15, 19, 25, 42, 44, 48], but these are often considered separately. In practice, however, users want an integrated solution that can find the "best" way to get to their destination considering all available modes of transportation. Within a metropolitan area, this includes buses, trains, driving, cycling, taxis, and, of course, walking. We refer to this as the *multimodal route planning* problem.

In fact, any public transportation network necessarily has a multimodal component, since journeys require some amount of walking. Existing solutions [9, 12, 17, 19, 25] handle this by predefining transfer arcs between nearby stations, and running a search algorithm on the public transit network to find the "best" journey. Unlike in road networks, however, defining "best" is not straightforward. For example, while some people want to arrive as early as possible, others are willing to spend a little more time to avoid extra transfers. Most recent approaches therefore compute the *Pareto set* [35] of non-dominating journeys optimizing multiple criteria, which is practical even for large metropolitan areas [19, 44].

Extending public transportation solutions to a full multimodal scenario (with unrestricted walking, biking, and taxis) may seem trivial at first: one could just incorporate routing techniques for road networks [16, 32, 36] to solve the new subproblems. Unfortunately, meaningful multimodal optimization needs to take more criteria into account, such as walking duration and costs. Some people are happy to walk 10 minutes to avoid an extra transfer, while others are not. In fact, some will walk half an hour to avoid using public transportation at all. Taking a taxi all the way to the airport is a good solution for some; users on a budget may prefer a cheaper solution. Not only do these additional criteria significantly increase the Pareto set [21, 31], but some of the resulting journeys tend to look unreasonable, as Appendix A illustrates.

* Partially supported by DFG grant WA 654/16-1 and by the EU FP7/2007-2013 (DG CONNECT.H5-Smart Cities and Sustainability), under grant agreement no. 288094 (project eCOMPASS).

As a result, recent research efforts tend to avoid multicriteria search altogether [8], looking for reasonable routes by other means. A natural approach is to work with a weighted combination of all criteria, transforming the search into a single-criterion problem [2, 4, 41, 51]. When extended to find the k-shortest paths [13, 28], this method can even take user preferences into account. Unfortunately, linear combination may produce undesired results [14] (see Appendix A for an example). To avoid such issues, another line of multimodal single-criterion research considers the computation of label-constrained quickest journeys [6, 40]. The idea is to label edges according to the mode of transportation and require paths to obey a user-defined pattern (often given as regular expressions), typically enforcing a hierarchy of modes [13, 51] (such as "no car travel between trains"). The main advantage of this strategy is that preprocessing techniques developed for road networks carry over [5, 18, 23, 37, 38]. This approach, however, can hide interesting journeys (for example, taking a taxi between train stations in Paris may be an option). In fact, this exposes a fundamental conceptual problem with label-constrained optimization: it essentially relies on the user to know her options before planning the journey.

Given the limitations of current approaches, we revisit the problem of finding multicriteria multimodal journeys on a metropolitan scale. Instead of optimizing each mode of transportation independently [26], we argue in Section 2 that most users optimize three criteria: travel time, convenience, and costs. While this produces a large Pareto set, we propose using fuzzy logic [27, 53] to filter it in a principled way to a modest-sized set of representative journeys. This postprocessing step is not only quick, but can also be user-dependent, incorporating personal preferences. As Section 3 shows, recent algorithmic developments [19, 23, 32] allow us to answer exact queries optimizing time and convenience in less than two seconds within a large metropolitan area, for the simpler scenario of walking, cycling, and public transit. Unfortunately, this is not enough for interactive applications, and becomes much slower when additional criteria, such as costs, are incorporated. We therefore also propose (in Section 4) heuristics (still multicriteria) that are significantly faster, and closely match the top journeys in the Pareto set. Section 5 presents a thorough experimental evaluation of all algorithms in terms of both solution quality and performance, and shows that our approach can be fast enough for interactive applications. Moreover, since it does not rely on heavy preprocessing, it can be used in fully dynamic scenarios.

2 Problem Statement

We want to find journeys in a network built from several *partial networks*. First, we have a *public transportation network* representing all available timetable-based means of transportation, such as trains, buses, rail, or ferries. We can specify this network in terms of its timetable, which is defined as follows. A *stop* is a location in the network (such as a train platform or a bus stop) in which a user can board or leave a particular vehicle. A *route* is a fixed sequence of stops for which there is scheduled service during the day; a typical example is a bus or subway line. A route is served by one or more distinct *trips* during the day; each trip is associated with a unique vehicle, with fixed (scheduled) arrival and departure times for every stop in the route.

Besides the public transportation network, we also take as input several *unrestricted networks*, with no associated timetable. Walking, cycling, and driving are modeled as distinct unrestricted networks, each represented as a directed graph $G = (V, A)$. Each vertex $v \in V$ represents an intersection and has associated coordinates (latitude and longitude). Each arc $(v, w) \in A$ represents a (directed) road segment and has an associated *duration* $\mathrm{dur}(v, w)$, which corresponds to the (constant) time it takes to traverse it.

The *integrated transportation network* is the union of these partial networks (timetable-based and unrestricted) with appropriate *link vertices*. In other words, vertices (or stops) in different networks are identified with one another to allow for changes in modes of transportation.

A query takes as input a *source location* s, a *target location* t, and a *departure time* τ, and produces one or more *journeys* that leave s no earlier than τ and arrive at t. A *journey* is a valid path in the integrated transportation network that obeys all scheduling constraints.

Note that, unlike previous work [9, 17, 19, 25, 46], we do not need to specify *footpaths*, which are short, time-independent connections between nearby public transportation stops that allow for transfers. For pure public transport optimization, adding these footpaths is often done by the operator of the network or by heuristics [17].

Criteria. We still have to define *which* journeys the query should return. We argue that, in multimodal networks, users optimize three natural criteria: arrival time, costs, and convenience. Since convenience is hard to measure, we use number of trips and walking duration as proxies. This results in a setup with four criteria: arrival time, number of trips, walking duration, and costs. Note that for our first (simplified) scenario (with public transit, cycling, and walking, but no taxi) we do not consider costs and work with only three criteria.

Given this setup, a first natural problem we need to solve is the *full multicriteria problem*, which must return a full (maximal) Pareto set of journeys. We say that a journey J_1 *dominates* J_2 if J_1 is strictly better than J_2 according to at least one criterion, and no worse according to all other criteria. A *Pareto set* is a set of pairwise nondominating journeys [44, 35]. Note that, if two journeys have equal values in all criteria, we only keep one.

Fuzzy Dominance. As already mentioned, full Pareto sets can be very large, with many similar journeys. Moreover, several journeys in the Pareto set may seem irrelevant. For example, consider a journey J_2 that arrives much later than a similar journey J_1, only to save a few seconds of walking. Both journeys are Pareto-optimal, but J_2 is arguably less relevant. The same applies to all criteria, and in both directions: walking much more to save a few seconds is also undesirable. Intuitively, most criteria are diffuse to the user, and only large enough differences are significant.

To formalize the notion of significance, we propose to *score* the journeys in the Pareto set in a post-processing step. We compute scores using concepts from fuzzy logic [53] (and fuzzy set theory [52]), which we briefly review. Loosely speaking, fuzzy logic aims at generalizing Boolean logic to handle (continuous) degrees of truth. For example, the statement "60 and 61 seconds walking are equal" is false in classical logic, but might be considered "almost true" in fuzzy logic. A *fuzzy set* is a tuple $\mathcal{S} = (\mathcal{U}, \mu)$, where $\mathcal{U}$ is a set, and $\mu\colon \mathcal{U} \to [0,1]$ a *membership function* that defines "how much" each element in $\mathcal{U}$ is contained in $\mathcal{S}$. If $\mathcal{U}$ is clear from the context, we just use μ to refer to $\mathcal{S}$. When $\mathcal{U}$ consists of logical statements, we call $\mathcal{S}$ a *fuzzy predicate*, and μ assigns each statement a truth value from $[0,1]$. For our purposes we require fuzzy relational operators $\mu_<$, $\mu_=$, and $\mu_>$ over the real numbers $\mathbb{R}$. For any $x, y \in \mathbb{R}$, they are used by evaluating $\mu_<(x-y)$, $\mu_>(y-x)$, and $\mu_=(x-y)$. The operators $(\mu_<, \mu_=, \mu_>)$ fulfill *Ruspini's condition* [47] if $\mu_< + \mu_> + \mu_= = 1$ holds, which is required for consistency. In this work, we always use exponential membership functions for the operators, i. e., $\mu_=(x) := \exp(\frac{\ln(\chi)}{\varepsilon^2}x^2)$, where $0 < \chi < 1$ and $\varepsilon > 0$ control the degree of fuzziness. The other two operators are derived by $\mu_<(x) := 1 - \mu_=(x)$ if $x < 0$ (0 otherwise), and $\mu_> := 1 - \mu_=(x)$ if $x > 0$ (0 otherwise). A *triangular norm* (short: *t-norm*) $T\colon [0,1]^2 \to [0,1]$ is a commutative, associative, and monotone (i. e., $a \leq b, x \leq y \Rightarrow T(a,x) \leq T(b,y)$) binary operator to which 1 is the neutral element. If $x, y \in [0,1]$ are truth values, $T(x,y)$ is interpreted as a fuzzy conjunction (*and*) of x and y. Given a t-norm T, the *complementary conorm* (or *s-norm*) of T is defined as $S(x,y) := 1 - T(1-x, 1-y)$, which we interpret as a fuzzy disjunction (*or*). Note that the neutral element of S is 0. Two well-known pairs of t- and s-norms are $(\min(x, y), \max(x, y))$, called *minimum/maximum norms*, and $(xy, x + y - xy)$, called *product norm/probabilistic sum.*

Following [27], we now recap the concept of fuzzy dominance in multicriteria optimization. Given journeys J_1 and J_2 with M optimization criteria, we denote by $n_b(J_1, J_2)$ the (fuzzy) number of criteria in which J_1 is better than J_2. More formally $n_b(J_1, J_2) := \sum_{i=1}^{M} \mu^i_<(\kappa^i(J_1), \kappa^i(J_2))$, where $\kappa^i(J)$ evaluates the i-th criterion of J, and $\mu^i_<$ is the i-th fuzzy less-than operator. (Note that we might use different fuzzy operators for each criterion.) Analogously, we define $n_e(J_1, J_2)$ for equality and $n_w(J_1, J_2)$ for greater-than. Note that from Ruspini's condition it holds that $n_b + n_e + n_w = M$. Hence the Pareto dominance can be generalized to obtain a *degree of domination* $d(J_1, J_2) \in [0, 1]$, which is defined as $(2n_b + n_e - M)/n_b$ if $n_b > (M - n_e)/2$ (and 0 otherwise). Here, $d(J_1, J_2) = 0$ means that J_1 does not dominate J_2, while a value of 1 indicates that J_1 Pareto-dominates J_2. Otherwise, we say J_1 *fuzzy-dominates* J_2 by degree $d(J_1, J_2)$. Now, given a (Pareto) set $\mathcal{J}$ of n journeys $J_1, \dots, J_n$ and an s-norm S, we define a *score function* $\mathrm{sc}\colon \mathcal{J} \to [0, 1]$ that computes the degree of domination by the whole set for each J_i. More precisely, we define sc to be $\mathrm{sc}(J) := 1 - S(J_1, \dots, J_n)$. Note that if we set S to be the maximum norm, the score is based on the (one) journey that dominates J most. On the other hand, with the probabilistic sum the score may be based on several fuzzily dominating journeys. We finally use the score to order the journeys by significance. One may then decide to only show the k journeys with highest score to the user.

3 Exact Algorithms

This section considers exact algorithms for the multicriteria multimodal problem. Sections 3.1 and 3.2 propose two solutions, each building on a different algorithm for multicriteria optimization on public transportation networks (MLC [46] and RAPTOR [19]). Section 3.3 then describes an acceleration technique that applies to both. To simplify the discussion (and notation), we first describe the algorithms in terms of our simplest scenario, considering only the (timetable-based) public transit network and the (unrestricted) walking network. Section 3.4 explains how to handle cycling and taxis, which are unrestricted but have special properties.

3.1 Multi-label-correcting Algorithm

Traditional solutions to the multicriteria problem on public transportation networks work by first modeling the timetable as a graph [11, 17, 30, 43]. A particularly effective approach is to use the *time-dependent route model* [46]. For each stop p, we create a single *stop vertex* linked by time-independent *transfer edges* to multiple *route vertices*, one for each route serving p. In addition, we add *route edges* between route vertices associated to consecutive stops within the same route. To model the trips along this route during the day, the cost of a route edge is given by a piecewise linear function. The cost of traversing an edge includes not only the time in transit until the next stop, but also the time waiting until the next trip departs.

A journey in the public transportation network corresponds to a path in this graph. The *multi-label-correcting* (MLC) [46] algorithm uses this fact to find full Pareto sets for arbitrary criteria that can be modeled as edge costs. MLC is based on Dijkstra's algorithm [24], but operates on *labels*, which are tuples with one value per optimization criterion. Each vertex v maintains a *bag* $B(v)$ consisting of a Pareto set of nondominated labels. In each iteration, MLC extracts from a priority queue the minimum (in lexicographic order) unprocessed label $L(u)$. For each arc (u, v) out of the associated vertex u, the algorithm creates a new label $L(v)$ (by extending $L(u)$ in the natural way) and inserts it into $B(v)$; newly-dominated labels (possibly including $L(v)$ itself) are discarded, and the priority queue is updated if needed. This basic algorithm can be sped up significantly with techniques such as target pruning and avoiding unnecessary domination checks [25].

Extending MLC to solve the multimodal problem is straightforward: it suffices to augment its input graph to include the walking network. We can combine the original graphs by merging (public transportation) stops and (walking) intersections that share the same location (and keeping all edges). These *link vertices* can then be used to switch between modes of transportation. The MLC query itself remains essentially unchanged, and still processes labels in lexicographic order. Although labels can now be associated to vertices in different networks (public transportation or walking), they can all share the same priority queue.

3.2 Round-based Algorithm

An important drawback of the MLC algorithm (even restricted to public transportation networks) is that it can be quite slow: unlike Dijkstra's algorithm, MLC may scan the same vertex multiple times (the exact number depends on the criteria being optimized), and domination checks make each such scan quite costly. Delling et al. [19] have recently introduced RAPTOR (*Round bAsed Public Transit Optimized Router*) as a faster alternative for public transportation. The simplest version of the algorithm optimizes two criteria: arrival time and number of transfers. Unlike MLC, which searches a graph built from the timetable, RAPTOR operates directly on the timetable using a dynamic programming approach. The algorithm works in rounds, with round i processing all relevant journeys with up to $i - 1$ transfers. It maintains one label per round i and stop p representing the best known arrival time at p for up to i trips. During round i, the algorithm first processes each *route* once. It reads arrival times from round $i - 1$ to determine relevant trips (on the route), and updates the labels of round i at every stop along the way. Once all routes are processed, the algorithm considers potential transfers by looking at all (predefined) footpaths. Simpler data structures and better locality make RAPTOR an order of magnitude faster than MLC. Delling et al. [19] have also proposed McRAPTOR, an extension of RAPTOR that can handle more criteria (besides arrival times and number of transfers) by maintaining a *bag* (set) of labels with each stop and round.

Here we propose MCR (*multimodal multicriteria RAPTOR*), which extends McRAPTOR to handle multimodal queries. As in McRAPTOR, each round has two phases: the first processes trips in the public transportation network, while the second considers arbitrary paths in the unrestricted networks. We use a standard McRAPTOR round for the first phase (with no footpaths), and MLC for the second (on the walking network). For consistency with McRAPTOR, during the second phase MLC keeps a heap of bags (instead of individual labels). The algorithm keeps track of which labels have already been extended, and ensures that no label is processed more than once. Labels generated by one phase are naturally used as input to the other. The initialization routine (before the first round) runs Dijkstra's algorithm on the walking network from the source s to determine the fastest walking path to each stop in the public transportation network (and to t), thus creating the initial labels used by MCR. Note that, during round i, McRAPTOR reads labels from round $i - 1$ and writes to round i. In contrast, MLC may read and write labels of the same round if walking is not regarded as a trip.

3.3 Contracting the Unrestricted Networks

As our experiments will show, the bottleneck of the multimodal algorithms (based on either MLC or RAPTOR) is processing the (quite large) walking network. To improve performance, we use a quick preprocessing technique proposed by Dibbelt et al. [23], which we describe next.

The key observation is that, for any journey involving public transportation, walking between trips always begins and ends at the restricted set K of link vertices. We can exploit this fact by building (in a preprocessing step) an *overlay* [48] of the walking network $G = (V, A)$, which is a graph on $K \subset V$ where every pairwise distance is the same as in G. A standard way of computing such overlays is to use *contraction* [32], as follows. First, we define a total order (given by a rank function r) among the vertices in $V \setminus K$, then *shortcut* them in this order. (Vertices in K have infinite rank.) To *shortcut* a vertex v, we delete it (and its adjacent arcs) from the graph and add as few arcs as necessary to preserve distances among the remaining vertices. More precisely, given two arcs (u, v) and (v, w), we add a new *shortcut arc* (u, w) with $\mathrm{dur}(u, w) = \mathrm{dur}(u, v) + \mathrm{dur}(v, w)$ if and only if u–v–w is the only shortest path between u and w in the remaining graph. Once only vertices in K remain, the graph will be an overlay. Although the final outcome does not depend on the contraction order, the running time of this procedure does, since the density of intermediate graphs may vary. Among other criteria, it is usually advantageous to contract well-separated vertices with relatively small degrees first [32].

Even if good orders are observed, it is often the case that the overlay graph itself has too many arcs, negating the benefits of having fewer vertices. It is well known [10, 16, 23, 33] that this can be avoided by stopping the contraction as soon as the average degree of the contracted graph reaches a certain threshold (we use 12 in our experiments). Although not an overlay, this *core graph* does preserve the distances between all vertices in K, which is what we need.

We now have all the elements in place for a faster multimodal s–t query. We run essentially the same algorithm as before (based on either MLC or RAPTOR), but replace the full walking network with the (more compact) core graph. Since the source s and the target t may not be in the core, we handle them as special cases during initialization. It works on the graph $G^+ = (V, A \cup A^+)$ containing all original arcs A as well as all shortcuts A^+ added (even temporarily) during the contraction process. (Note that G^+ is a supergraph of both G and its core.) We run upward searches (i.e., only following arcs (u, v) such that $r(u) > r(w)$) in G^+ from s (scanning forward arcs) and t (scanning reverse arcs); they reach all potential entry and exit points in the core, and arcs within the core are not processed. The core vertices reached from either s or t (and their respective distances) are used as input to MCR's (or MLC's) standard initialization, which can operate on the core from this point on.

The main loop proceeds exactly as before, with one minor adjustment. For MLC, whenever we extract a label $L(v)$ associated with a scanned core vertex v, we check whether it has been reached by the backward search during initialization. If this is the case, we create a temporary label $L'(t)$ by extending $L(v)$ with the (already computed) walking path to t, potentially inserting it into $B(t)$. MCR is adjusted similarly, with bags instead of labels.

3.4 Beyond Walking

This section explains how we can handle other unrestricted networks (besides walking). In particular, our experiments include a bicycle rental scheme, which can be seen as a hybrid network: while it does not have a fixed schedule (and is thus unrestricted), bicycles can only be picked up and dropped off at certain designated locations (called *cycling stations*). We therefore consider them as stops (part of the public transportation network), but without a schedule—a bike can be used at any time. Moreover, we count one extra trip each time a bike is picked up from a cycling station. To handle cycling within MCR, we consider it during the first stage of each round (together with RAPTOR and before walking). Because bicycles have no schedule, we process them independently (from RAPTOR) by running MLC on the bicycle network. To do so, we initialize MLC with labels from round $i-1$ for all relevant bicycle stations, and during the algorithm we update labels of (the current) round i.

We handle taxis as a separate criterion reflecting the (monetary) *cost* of taxi rides. Moreover, we consider a taxi ride to be a trip, since we board a vehicle. In our round-based algorithms, we handle taxi the same way we handle walking, with the exception that in the taxi stage labels are read and written in consecutive rounds $i-1$ and i, respectively. Note that (unlike rental bicycles) we also allow taking a taxi as the first and/or last leg of any location-to-location query. As our experiments will show, incorporating taxi tends to increase the size of the full Pareto set (and running times) significantly. Note that, if taxis were not penalized in any way, an all-taxi journey would almost always dominate all other alternatives (even sensible ones), since it is fast and has no walking.

Dealing with personal cars or bicycles is somewhat simpler. Since we can assume that these vehicles are only available for the first or last legs of the journey, we must only consider them during initialization. The initialization phase can also be used to deal with other special cases, such as allowing a rented bicycle to be ridden to the final destination (to be returned later).

Note that the contraction scheme from Section 3.3 can also be used for the cycling and driving networks. For every unrestricted network (walking, cycling, driving), we keep the link vertices (stops and bicycle stations) in one common core, contracting (up to) all other nodes. As before, queries then start with upward searches in each relevant unrestricted network.

4 Heuristics

Even with the accelerations we consider, the exact algorithms proposed in Section 3 are not fast enough for interactive real-world applications. This section presents several heuristics aimed at quickly finding a set of journeys that is similar to the exact solution, which we take as ground truth. We consider three general approaches: weakening the dominance rules, reducing the number of criteria, and restricting the lengths of certain trips. We explain each in turn, then discuss how we can actually evaluate the quality of the solutions found by these heuristics.

Weak dominance. The first strategy we consider is to relax the domination rules during the algorithm, instead of computing the full Pareto set. Although this may lead to suboptimal solutions, it may be faster because it reduces the number of labels pushed through the network.

We consider four different implementations of this strategy. The first, MCR-hf, uses fuzzy dominance (instead of strict dominance) when comparing labels during the algorithm: for labels L_1 and L_2, we compute the fuzzy dominance value $d(L_1, L_2)$ (cf. Section 2), and dominate L_2 if d exceeds a given threshold (we use 0.9). The second, MCR-hb(κ), uses strict domination, but discretizes criterion κ: before comparing labels L_1 and L_2, we first round $\kappa(L_1)$ and $\kappa(L_2)$ to predefined discrete values (*buckets*). Of course, this can be extended to use buckets for several criteria. The third heuristic, MCR-hs(κ), uses strict domination but adds a slack of x units to κ. More precisely, L_1 already dominates L_2 if $\kappa(L_1) \leq \kappa(L_2) + x$ and L_1 is at least as good L_2 in all other criteria. The last heuristic, MCR-ht, relaxes the domination rule by trading off two or more criteria. More concretely, consider the case in which walking (walk) and arrival time (arr) are criteria. Then, L_1 already dominates L_2 if $\text{arr}(L_1) \leq \text{arr}(L_2) + a \cdot (\text{walk}(L_1) - \text{walk}(L_2))$, $\text{walk}(L_1) \leq \text{walk}(L_2) + a \cdot (\text{arr}(L_1) - \text{arr}(L_2))$, and L_1 is at least as good as L_2 in all other criteria, for a tradeoff parameter a.

Fewer criteria. The second heuristic acceleration we test is to reduce the number of criteria considered during the algorithm. For concreteness, we explain our approach for the simple multimodal scenario (without taxis), with three criteria: arrival time, walking duration, and number of trips. Our heuristic, MR-x, still works in rounds, but optimizes only the number of trips and arrival times explicitly (as criteria). To take walking duration into account, we count every x minutes of a walking segment (transfer) as a trip; the first x minutes are free. With this approach, it suffices to run plain Dijkstra to compute transfers, since link vertices no longer need to keep bags. The round index to which labels are written then depends on the walking duration (of the current segment) of the considered label. A particularly effective special case is $x = \infty$, where a transfer is never a trip. Another effective approach is a variant where a transfer always counts as a single trip, regardless of duration; we abuse notation and call this variant MR-0. One can obtain even better results by running MR-0 and MR-∞ independently and returning the union of the journeys they find; we call this approach MR-$(0, \infty)$.

When we must also consider costs are a criterion (to handle taxis), we propose the MCR-hc heuristic. Once again, we drop walking as an independent criterion, leaving only arrival time, number of trips, and costs to be optimized. To account for walking, we make it another (cheaper) component of the cost criterion.

Restricting criteria. Consider our simple scenario of walking and public transit. Intuitively, most journeys start with a walk to a nearby stop, followed by one or more trips (with short transfers) within the public transit system, and finally a short walk from the final stop to the actual destination. To take this observation into account, we propose MCR-tx-ry. It still runs three-criterion search (walking, arrival, and trips), but limits walking in the beginning and end to y minutes, and walking transfers between stops to x minutes. A related variant we tested (MR-tx-ry) has the same constraints, but runs only bicriteria search (optimizing arrival time and trips). Note that existing solutions often use such restrictions [9].

Quality evaluation. One challenge with heuristics is evaluating how good they are. We take the solution found by MCR to be the *ground truth*, then measure the quality of a heuristic by determining how close it is to the ground truth. To quantify this, we first compute the fuzzy score values for each journey with respect to their Pareto set. Then, for a given parameter k, we measure the similarity between the top k scored journeys returned by the heuristics and the top k scored journeys in the ground truth. Note that the score depends on the algorithm only, and does not assume knowledge of the ground truth, which is consistent with a real-world deployment of the algorithms. To compare two sets of k journeys, we run a greedy maximum matching algorithm. First, we compute a $k \times k$ matrix where entry (i, j) represents the similarity between the i-th journey in the first set and the j-th in the second. To measure the similarity, we make use of the same fuzzy relational operators we use for scoring. More precisely, given two journeys J_1 and J_2, the similarity with respect to the i-th criterion is given by $c^i := \mu^i_{=}(\kappa^i(J_1) - \kappa^i(J_2))$, where κ^i is the value of this criterion and $\mu^i_{=}$ is the corresponding fuzzy equality relation. Then, we define the similarity $\text{sim}(J_1, J_2)$ as $T(c^1, c^2, \ldots, c^M)$, where T is an arbitrary t-norm. We always select T to be consistent with the s-norm that we use to compute the score values. Having computed the pairwise similarities, we greedily select the unmatched pairs with highest similarity (by picking the highest entry that does not share a row or column with a previously picked entry) from the table. The similarity of the whole matching is then the average similarity of its pairs, but weighted by the fuzzy score of the reference journey. This means that matching the highest-score reference journey is more important than matching the k-th one.

5 Experiments

This section presents an extensive evaluation of the methods introduced in this paper. We implemented all algorithms from Sections 3 and 4 in C++ and compiled the code with `g++` 4.6.2 (64 bit) with optimization flag `-O3`. We conducted our experiments on one core of a dual 8-core Intel Xeon E5-2670 machine clocked at 2.6 GHz, with 64 GiB of DDR3-1600 RAM.

Table 1. Performance and solution quality on journeys considering walking, cycling, and public transportation. Bullets (•) indicate which criteria are taken into account by the algorithm.

					# Scans	# Comp.		Time	Quality-3		Quality-6	
Algorithm	Arr.	Trp.	Wlk.	# Rnd.	p. Ent.	p. Ent.	# Jn.	[ms]	Avg.	Sd.	Avg.	Sd.
MCR-full	•	•	•	13.8	13.8	168.2	29.1	4 634.0	100 %	0 %	100 %	0 %
MCR	•	•	•	13.8	3.4	158.7	29.1	1 438.7	100 %	0 %	100 %	0 %
MLC	•	•	•	—	10.6	1 246.7	29.1	4 543.0	100 %	0 %	100 %	0 %
MCR-hf	•	•	•	15.6	2.9	14.3	10.9	699.4	89 %	15 %	89 %	11 %
MCR-hb	•	•	•	10.2	2.1	12.7	9.0	456.7	91 %	12 %	91 %	10 %
MCR-hs	•	•	•	14.7	2.6	11.1	8.6	466.1	67 %	28 %	69 %	23 %
MCR-ht	•	•	•	10.5	2.0	6.4	8.6	373.6	84 %	22 %	82 %	20 %
MCR-t5	•	•	•	13.8	2.7	126.6	28.9	891.9	93 %	16 %	92 %	15 %
MCR-t10	•	•	•	13.8	2.7	132.7	29.0	1 467.6	97 %	10 %	95 %	10 %
MCR-t10-r15	•	•	•	10.7	1.7	73.3	13.2	885.0	38 %	40 %	30 %	31 %
MR-t10	•	•	○	7.6	1.1	4.8	4.5	22.2	63 %	28 %	62 %	24 %
MR-∞	•	•	○	7.6	1.4	4.8	4.5	44.4	63 %	28 %	63 %	24 %
MR-0	•	•	○	13.7	2.1	6.9	5.4	61.5	63 %	28 %	63 %	24 %
MR-10	•	•	○	20.0	1.1	4.8	4.3	39.4	51 %	33 %	45 %	29 %
MR-(0,∞)	•	•	○	13.7	3.5	11.6	6.1	108.8	66 %	27 %	66 %	23 %

Input and Methodology. We focus on the transportation network of London (England); results for other instances (available in Appendix B.2) are similar. For public transportation, we use the timetable information made available by Transport for London (TfL) [39, 50], from which we extracted a Tuesday in the periodic summer schedule of 2011. It includes all subway (tube), buses, tram, and light rail (DLR) data. To model the underlying road network, we use PTV data from 2006 [45], which explicitly indicates whether each road segment is open for driving, cycling and/or walking. In the walking network, we set the walking speeds to 5 km/h. To compute driving times (for experiments that need them), we assume driving at the maximum allowed speed limit. For simplicity, we do not consider turn costs (which are not well defined in the data). We obtained the bicycle station data from the TfL website [50] and assume an average cycling speed of 12 km/h. The resulting combined network has about 20 k stops, 5 M departure events, 564 cycle stations, and 259 k vertices in the walking network.

Recall that we specify the fuzziness of each criterion by a pair (χ, ϵ), roughly meaning that the corresponding Gaussian (centered at $x = 0$) has value χ for $x = \epsilon$. We set these pairs to $(0.8, 5)$ for walking, $(0.8, 1)$ for arrival time, $(0.1, 1)$ for trips, and $(0.8, 5)$ for costs (given in pounds; times are in minutes). Note that the number of trips is sharper than the other criteria. For simplicity, our experiments consider only the minimum/maximum norms. Later in this section we show that our approach is robust to small variations in these parameters, but they can be tuned to account for user-dependent preferences.

Our experiments consider *location-to-location* queries, with sources, targets, and departure times picked uniformly at random (from the walking network and during the day, respectively).

Algorithms Evaluation. For our first experiment, we ran 1 000 queries for each algorithm, considering walking, cycling, and the public transportation network, and considering three criteria: arrival time, number of trips, and walking duration. The results are summarized in Table 1 (Appendix B.1 has additional statistics). For each algorithm, the table first shows which of the three criteria are explicitly taken into account. The next five columns show the average (over all 1 000 runs) values observed for the number of rounds, scans per entity (stop/vertex), label comparisons per entity, journeys found, and running time (in milliseconds). The last four

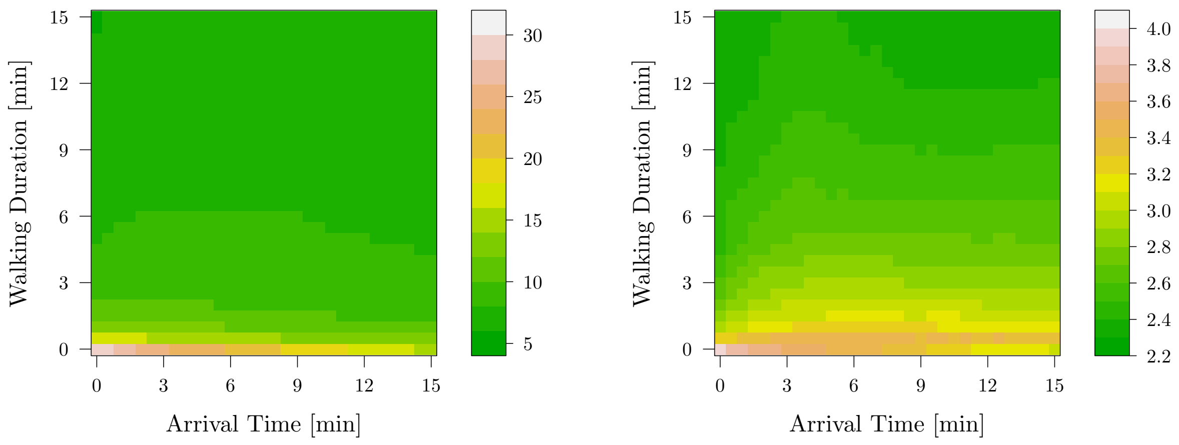

Fig. 1. Number of Pareto optimal journeys with score higher than 0.1 for varying fuzziness. We consider both the maximum norm (left) and probabilistic sum (right). The x axis varies the fuzziness in the arrival time, while the y axis considers the walking duration. The intensity (color) of the corresponding entry indicates the average number of journeys in the filtered output.

columns evaluate the quality of the top 3 and 6 journeys found by our heuristics, as explained in Section 4. Note that we show both averages and standard deviations.

The algorithms in Table 1 are grouped in blocks. The first contains methods that compute the full Pareto set according to all three criteria (arrival time, number of trips, and walking). Our reference algorithm is MCR, which is round-based and uses contraction to accelerate computations in the unrestricted networks. As anticipated, it is faster (by a factor of roughly three) than either MCR-full (which does not use the core) or MLC (which uses the core but is not round-based). Accordingly, all heuristics we consider are round-based and use the core.

The second block contains heuristics that accelerate MCR by weakening the domination rules, causing more labels to be pruned (and losing optimality guarantees). As explained in Section 4, MCR-hf uses fuzzy dominance during the algorithm, MCR-hb uses walking *buckets* (discretizing walking by steps of 5 minutes for domination), MCR-hs uses a slack of 5 minutes on the walking criterion when evaluating domination, and MCR-ht considers a tradeoff parameter of $a = 0.3$ between walking and arrival time. Although all heuristics are faster than pure MCR, the speedup is more limited for MCR-hf than for any of the other approaches. The best tradeoff between running time and solution quality is given by MCR-hb.

The third block has algorithms with restrictions on walking duration. Limiting transfers to 10 minutes (as MCR-t10 does) has almost no effect on solution quality (which is to be expected in a well-designed public transportation network). Unfortunately, this heuristic is not faster than than the full algorithm, since it adds too many arcs to represent precomputed footpaths. If we also limit the walking duration from s or t (MCR-t10-r15), the algorithm becomes slightly faster, but quality becomes unacceptably low: the algorithm misses good journeys (including all-walk) quite often. If instead we allow even more restricted transfers (with MCR-t5), we get similar speedups with much better quality. A much faster alternative is MR-t10, which drops walking duration as a criterion (it is considered only implicitly in the arrival time), making it essentially the same as RAPTOR, with a different initialization. Quality is much lower than for MCR-tx, however, indicating that considering the walking duration explicitly during the algorithm is important to obtain a full range of solutions.

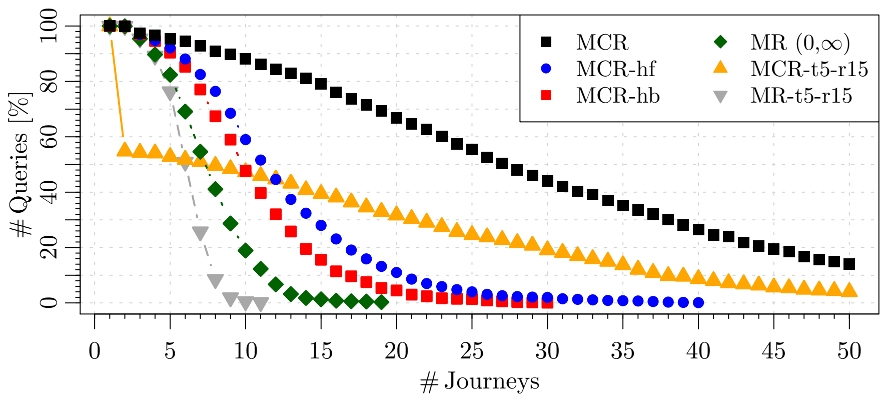

Fig. 2. Evaluating the number of journeys returned by some of our algorithms: For a given n (on the abscissa), we report the percentage of 1 000 random queries that compute n or more journeys.

The MR-x algorithms (fourth block) attempt to improve quality by transforming long walks into extra trips, but they are not particularly successful. While they do consider more journeys during the algorithm (resulting in higher running times), solutions are not much better. For such level of solution quality, MR-t10 seems to be a better choice. Summing up, MCR-hb should be the preferred choice for high-quality solutions, while MR-t10 can support interactive queries with reasonable quality.

Fuzzy Parameters Evaluation. We also evaluated the impact of the fuzzy parameters on the number of journeys we obtain. We again use London with walking, public transit, and cycling as input. Figure 1 shows the number of journeys given a score higher than 0.1 (by the fuzzy ranking routine) when we vary ε (the level of fuzziness) for two criteria, walking and arrival time. Note that we set $\chi = 0.8$, as in our main experiments. To simplify the exposition, we keep the fuzziness of the third criterion (number of trips) constant.

A comparison between the plots shows that, for the same set of parameters, probabilistic sum is significantly stricter than the maximum norm, and reduces the number of journeys much more drastically (for a fixed threshold). Qualitatively, however, they behave similarly. Under both norms, making the walking criterion fuzzier is more effective at identifying unwanted journeys. A couple of minutes of fuzziness in the walking criterion is enough to significantly reduce the number of journeys above the threshold. Adding fuzziness only to the arrival time has much more limited effect on the results.

Quality of the Heuristics. We here further investigate the quality of our heuristics. We use London with walking, public transit, and cycling as input. Figure 2 reports the size of the Pareto set (the input to scoring) for various algorithms, while Figure 3 shows how well the the top k heuristic journeys match the ground truth, for varying k. We observe that exact MCR does indeed produce many journeys, supporting the notion of ranking them afterwards (by score). A good heuristic, such as MCR-hb, computes much fewer journeys, but they match the top MCR journeys quite well. An interesting observation is that the quality of the heuristic hardly depends on the number of journeys we try to match.

Full Multimodal Problem. Our final experiment considers the full multimodal problem, also including taxis. As explained in Section 3.4, we add *cost* as fourth criterion (at 2.40 pounds per taxi-trip plus 60 pence per minute). For simplicity, we do not consider the cost of public transit, since it is significantly cheaper. Table 2 presents the average performance of some of

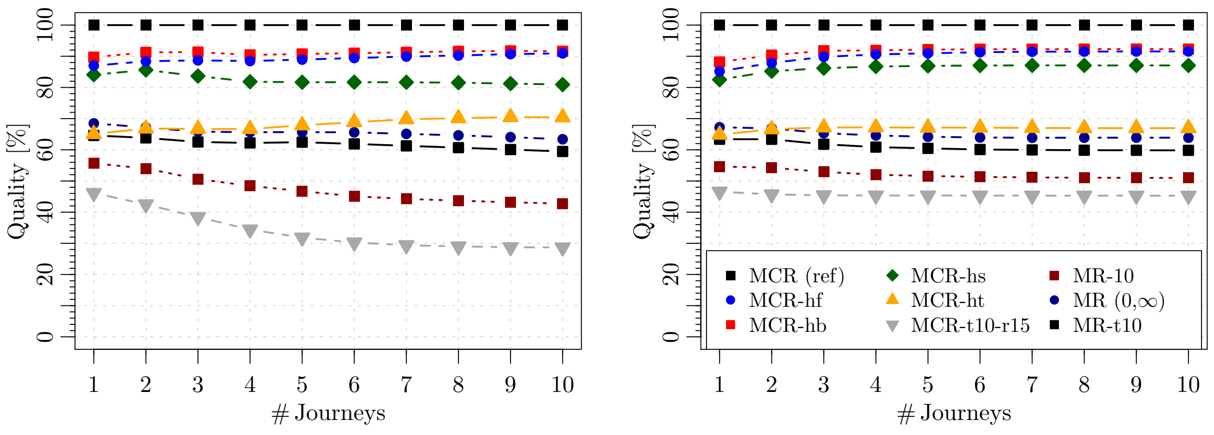

Fig. 3. Evaluating the solution quality by matching the top k journeys in the solution with the top k of the reference algorithm (MCR). The scores and similarity values are obtained by using the minimum/maximum norms (left) and the product norm/probabilistic sum (right). The legend of the right plot also applies to the left.

our algorithms over 1 000 random queries in London. The first block includes algorithms that optimize all four criteria (arrival time, walking duration, number of trips, and costs). Note that exact MCR becomes impractical. Fuzzy domination (MCR-hf) makes the problem tractable, with little loss in quality. Using 5-minute buckets for walking and 5-pound buckets for costs (MCR-hb) is even faster, though queries still take more than two seconds. The second block in the table shows that we can reduce times if we drop walking duration as criterion (we incorporate it into the cost function at 3 pence per minute, instead), with almost no loss in solution quality. This is still not fast enough, though. Using 5-pound buckets (MCR-hb) reduces the average query times to about 1 second, with reasonable quality.

6 Final Remarks

We have studied multicriteria journey planning in metropolitan multimodal networks. We argued that users of such networks optimize three criteria: arrival time, costs, and convenience. It turns out that the corresponding full Pareto set is large, with many unnatural journeys. Fuzzy set theory can extract the relevant journeys and rank them. Since exact algorithms are too slow, we have introduced several heuristics that closely match the best journeys in the Pareto set. Extensive experiments show that our approach enables efficient realistic multimodal journey planning. A natural avenue for future research is accelerating our approach further to enable interactive queries with an even richer set of criteria. Ultimately, the overall goal is to compute multicriteria multimodal journeys on a global scale in real time.

Table 2. Evaluating the performance on our London instance when taking taxi into account.

Algorithm	Arr.	Trp.	Wlk.	Cost	# Rnd.	# Scans p. Ent.	# Comp. p. Ent.	# Jn.	Time [ms]	Quality-3 Avg.	Quality-3 Sd.	Quality-6 Avg.	Quality-6 Sd.
MCR	•	•	•	•	16.3	3.1	369 606.0	1 666.0	1 960 234.0	100 %	0 %	100 %	0 %
MCR-hf	•	•	•	•	17.1	2.1	137.1	35.2	6 451.6	92 %	12 %	92 %	6 %
MCR-hb	•	•	•	•	9.9	1.3	86.8	27.6	2 807.7	96 %	8 %	92 %	6 %
MCR	•	•	○	•	14.6	2.4	7 901.4	250.9	25 945.8	98 %	6 %	97 %	5 %
MCR-hf	•	•	○	•	12.0	1.4	33.6	17.6	2 246.3	87 %	12 %	74 %	12 %
MCR-hb	•	•	○	•	9.0	1.0	20.0	11.6	996.4	86 %	12 %	74 %	12 %

References

1. I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. Hierarchical Hub Labelings for Shortest Paths. In L. Epstein and P. Ferragina, editors, *Proceedings of the 20th Annual European Symposium on Algorithms (ESA'12)*, volume 7501 of *Lecture Notes in Computer Science*, pages 24–35. Springer, 2012.
2. G. Aifadopoulou, A. Ziliaskopoulos, and E. Chrisohoou. Multiobjective Optimum Path Algorithm for Passenger Pretrip Planning in Multimodal Transportation Networks. *Journal of the Transportation Research Board*, 2032(1):26–34, December 2007. 10.3141/2032-04.
3. *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*. SIAM, 2012.
4. L. Antsfeld and T. Walsh. Finding Multi-criteria Optimal Paths in Multi-modal Public Transportation Networks using the Transit Algorithm. In *Proceedings of the 19th ITS World Congress*, 2012.
5. C. Barrett, K. Bisset, M. Holzer, G. Konjevod, M. V. Marathe, and D. Wagner. Engineering Label-Constrained Shortest-Path Algorithms. In Demetrescu et al. [22], pages 309–319.
6. C. Barrett, R. Jacob, and M. V. Marathe. Formal-Language-Constrained Path Problems. *SIAM Journal on Computing*, 30(3):809–837, 2000.
7. H. Bast. Car or Public Transport – Two Worlds. In S. Albers, H. Alt, and S. Näher, editors, *Efficient Algorithms*, volume 5760 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2009.
8. H. Bast. Next-Generation Route Planning: Multi-Modal, Real-Time, Personalized, 2012. Talk given at ISMP.
9. H. Bast, E. Carlsson, A. Eigenwillig, R. Geisberger, C. Harrelson, V. Raychev, and F. Viger. Fast Routing in Very Large Public Transportation Networks using Transfer Patterns. In *Proceedings of the 18th Annual European Symposium on Algorithms (ESA'10)*, volume 6346 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2010.
10. R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm. *ACM Journal of Experimental Algorithmics*, 15(2.3):1–31, January 2010. Special Section devoted to WEA'08.
11. A. Berger, D. Delling, A. Gebhardt, and M. Müller–Hannemann. Accelerating Time-Dependent Multi-Criteria Timetable Information is Harder Than Expected. In *Proceedings of the 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'09)*, OpenAccess Series in Informatics (OASIcs), 2009.
12. A. Berger, M. Grimmer, and M. Müller–Hannemann. Fully Dynamic Speed-Up Techniques for Multi-criteria Shortest Path Searches in Time-Dependent Networks. In Festa [29], pages 35–46.
13. M. Bielli, A. Boulmakoul, and H. Mouncif. Object modeling and path computation for multimodal travel systems. *European Journal of Operational Research*, 175(3):1705–1730, 2006.
14. D. W. Corne, K. Deb, P. J. Fleming, and J. D. Knowles. The Good of the Many Outweighs the Good of the One: Evolutionary Multi- Objective Optimization. *Connections*, 1(1):9–13, 2003.
15. D. Delling. Time-Dependent SHARC-Routing. *Algorithmica*, 60(1):60–94, May 2011.
16. D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable Route Planning. In P. M. Pardalos and S. Rebennack, editors, *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, volume 6630 of *Lecture Notes in Computer Science*, pages 376–387. Springer, 2011.
17. D. Delling, B. Katz, and T. Pajor. Parallel Computation of Best Connections in Public Transportation Networks. *ACM Journal of Experimental Algorithmics*, 2012. To appear.
18. D. Delling, T. Pajor, and D. Wagner. Accelerating Multi-Modal Route Planning by Access-Nodes. In A. Fiat and P. Sanders, editors, *Proceedings of the 17th Annual European Symposium on Algorithms (ESA'09)*, volume 5757 of *Lecture Notes in Computer Science*, pages 587–598. Springer, September 2009.
19. D. Delling, T. Pajor, and R. F. Werneck. Round-Based Public Transit Routing. In ALENEX'12 [3], pages 130–140.
20. D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering Route Planning Algorithms. In J. Lerner, D. Wagner, and K. A. Zweig, editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 117–139. Springer, 2009.
21. D. Delling and D. Wagner. Pareto Paths with SHARC. In J. Vahrenhold, editor, *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA'09)*, volume 5526 of *Lecture Notes in Computer Science*, pages 125–136. Springer, June 2009.
22. C. Demetrescu, A. V. Goldberg, and D. S. Johnson, editors. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, volume 74 of *DIMACS Book*. American Mathematical Society, 2009.
23. J. Dibbelt, T. Pajor, and D. Wagner. User-Constrained Multi-Modal Route Planning. In ALENEX'12 [3], pages 118–129.
24. E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.
25. Y. Disser, M. Müller–Hannemann, and M. Schnee. Multi-Criteria Shortest Paths in Time-Dependent Train Networks. In C. C. McGeoch, editor, *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, volume 5038 of *Lecture Notes in Computer Science*, pages 347–361. Springer, June 2008.

26. A. Ensor and F. Lillo. Partial order approach to compute shortest paths in multimodal networks. Technical report, http://arxiv.org/abs/1112.3366v1, 2011.
27. M. Farina and P. Amato. A Fuzzy Definition of “Optimality” for Many-Criteria Optimization Problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 34(3):315–326, 2004.
28. Q.-W. Feng, X. Zheng, and C. Yan. Study and Practice of an Improving Multi-path Search Algorithm in a City Public Transportation Network. In *Advanced Electrical and Electronics Engineering*, volume 87 of *Lecture Notes in Electrical Engineering*, pages 87–96. Springer, 2011.
29. P. Festa, editor. *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*, volume 6049 of *Lecture Notes in Computer Science*. Springer, May 2010.
30. R. Geisberger. Contraction of Timetable Networks with Realistic Transfers. In Festa [29], pages 71–82.
31. R. Geisberger, M. Kobitzsch, and P. Sanders. Route Planning with Flexible Objective Functions. In *Proceedings of the 12th Workshop on Algorithm Engineering and Experiments (ALENEX'10)*, pages 124–137. SIAM, 2010.
32. R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transportation Science*, 46(3):388–404, August 2012.
33. R. Geisberger and D. Schieferdecker. Heuristic Contraction Hierarchies with Approximation Guarantee. In *Proceedings of the 3rd International Symposium on Combinatorial Search (SoCS'10)*. AAAI Press, 2010.
34. F. Geraets, L. G. Kroon, A. Schöbel, D. Wagner, and C. Zaroliagis. *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*. Springer, 2007.
35. P. Hansen. Bricriteria Path Problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making – Theory and Application –*, pages 109–127. Springer, 1979.
36. M. Hilger, E. Köhler, R. H. Möhring, and H. Schilling. Fast Point-to-Point Shortest Path Computations with Arc-Flags. In Demetrescu et al. [22], pages 41–72.
37. D. Kirchler, L. Liberti, and R. W. Calvo. A Label Correcting Algorithm for the Shortest Path Problem on a Multi-Modal Route Network. In *Proceedings of the 11th International Symposium on Experimental Algorithms (SEA'12)*, volume 7276 of *Lecture Notes in Computer Science*. Springer, 2012.
38. D. Kirchler, L. Liberti, T. Pajor, and R. W. Calvo. UniALT for Regular Language Constraint Shortest Paths on a Multi-Modal Transportation Network. In *Proceedings of the 11th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'11)*, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 64–75, 2011.
39. London Data Store. http://data.london.gov.uk/, 2011.
40. A. O. Mendelzon and P. T. Wood. Finding Regular Simple Paths in Graph Databases. *SIAM Journal on Computing*, 24(6):1235–1258, 1995.
41. P. Modesti and A. Sciomachen. A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111(3):495–508, 1998.
42. M. Müller–Hannemann and M. Schnee. Finding All Attractive Train Connections by Multi-Criteria Pareto Search. In *Algorithmic Methods for Railway Optimization* [34], pages 246–263.
43. M. Müller–Hannemann, F. Schulz, D. Wagner, and C. Zaroliagis. Timetable Information: Models and Algorithms. In *Algorithmic Methods for Railway Optimization* [34], pages 67–90.
44. M. Müller–Hannemann and K. Weihe. Pareto Shortest Paths is Often Feasible in Practice. In *Proceedings of the 5th International Workshop on Algorithm Engineering (WAE'01)*, volume 2141 of *Lecture Notes in Computer Science*, pages 185–197. Springer, 2001.
45. PTV AG - Planung Transport Verkehr. http://www.ptv.de, 1979.
46. E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. Efficient Models for Timetable Information in Public Transportation Systems. *ACM Journal of Experimental Algorithmics*, 12(2.4):1–39, 2008.
47. E. H. Ruspini. A new approach to clustering. *Information and Control*, 15(1):22–32, 1969.
48. F. Schulz, D. Wagner, and K. Weihe. Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. *ACM Journal of Experimental Algorithmics*, 5(12):1–23, 2000.
49. C. Sommer. Shortest-Path Queries in Static Networks, 2012. Submitted. Preprint available at http://www.sommer.jp/spq-survey.htm.
50. Transport for London. http://www.tfl.gov.uk/, 2000.
51. H. Yu and F. Lu. Advanced multi-modal routing approach for pedestrians. In *2nd International Conference on Consumer Electronics, Communications and Networks*, pages 2349–2352, 2012.
52. L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8(3):338–353, 1965.
53. L. A. Zadeh. Fuzzy Logic. *IEEE Computer*, 21(4):83–93, 1988.

A Shortcomings of Existing Approaches

This appendix provides some examples where existing approaches fail. Figure 4 shows a known problem for pure multicriteria search in multimodal networks. Consider computing the best routes between two locations s and t in a road network, given an additional public transit line with stops p_i which operates very frequently. We want to find the best route allowing taxi and public transit and consider two criteria: cost and arrival time. When computing the non-dominating journeys from s to t, all journeys $\langle s, p_i \rangle \rightarrow \langle p_i, p_5 \rangle \rightarrow \langle p_5, t \rangle$, $i = 1 \dots 4$, and $\langle s, t \rangle$ are optimal because the bus is cheaper and slower than the taxi. Of course, this can be extended to arbitrarily long routes.

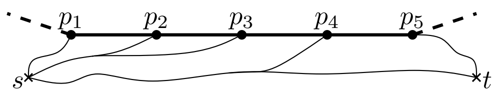

Fig. 4. Problem of multicriteria search in multimodal networks.

As discussed in Section 1, another approach to multimodal journey planning is computing a weighted combination of all criteria under consideration and then running a single-criterion search. However, especially for time-dependent problems (such as ours), the weighted combination of travel time with other criteria may yield bad journeys. For example, preferring cheaper subpaths might make us miss the last bus home, forcing us to take an expensive taxi.

Figure 5 shows a more concrete example. The Pareto-optimal set from stop p_1 to p_2 when departing at time $\tau = 0$ contains three journeys. Concatenating $J_1 + J_4$ yields an arrival time of 20 and a cost of 180, $J_1 + J_5$ yields 40 and 100, and $J_3 + J_6$ yields 180 and 20. Note that J_2 is never used. Now, for the weighted linear optimization of $\alpha \cdot (\text{arr} - \text{dep}) + (1 - \alpha) \cdot \text{cost}$ (with $\alpha \in [0, 1]$), one might expect we can obtain these journeys for different values of α. However, for $\alpha \in [0, 0.125]$ we get $J_3 + J_6$, for $\alpha \in (0.125, 0.875]$ we get $J_2 + J_6$, and for $\alpha \in (0.875, 1]$ we obtain $J_1 + J_4$. So, we do not find $J_1 + J_5$, which provides a reasonable trade-off between arrival time and costs. Even worse, for most values of α, we get a journey that is not part of the Pareto set.

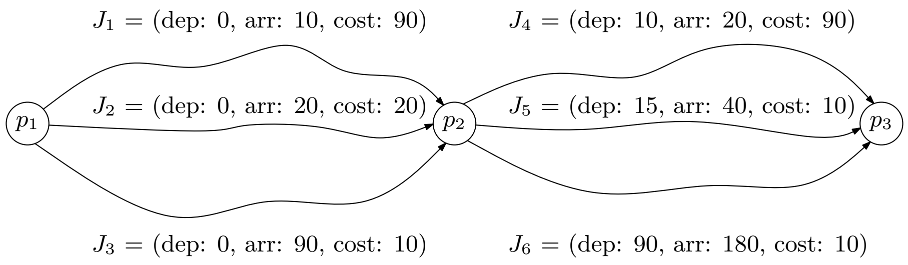
J_1 = (dep: 0, arr: 10, cost: 90)
J_4 = (dep: 10, arr: 20, cost: 90)
J_2 = (dep: 0, arr: 20, cost: 20)
J_5 = (dep: 15, arr: 40, cost: 10)
p_1
p_2
p_3
J_3 = (dep: 0, arr: 90, cost: 10)
J_6 = (dep: 90, arr: 180, cost: 10)

Fig. 5. Problem of linear combination search in time-dependent multimodal networks.

B Further Experiments

This appendix provides a more detailed analysis of our algorithms, and presents experiments on additional inputs.

B.1 Detailed Performance

Table 3 presents a more detailed analysis of the main experiment in Section 5 (without taxis). For each algorithm, it shows the effort (number of scans per vertex and/or stop, as well as running times in milliseconds) spent in each of the networks (public transit, walking, and cycling) and in total. The table shows that all round-based algorithms except MR-t10 spend more time processing the unrestricted networks (walking and cycling) than dealing with public transportation. This was to be expected: not only are the unrestricted networks bigger (they have more vertices), but also they must be processed with a (slower) Dijkstra-based algorithm (as in MLC, rather than RAPTOR). This is the reason for the good performance of the MR-t10 heuristic.

Table 3. Detailed performance analysis of our algorithms. The total running time includes additional overhead, such as for initialization.

				Public Transit		**Walking**		**Cycling**		**Total**	
Algorithm	Arr.	Trp.	Wlk.	# Scans p. Stop	Time [ms]	# Scans p. Vert.	Time [ms]	# Scans p. Vert.	Time [ms]	# Scans p. Ent.	Time [ms]
MCR-full	●	●	●	32.1	350.6	9.6	3 030.9	43.6	1 203.1	13.8	4 634.0
MCR	●	●	●	32.1	341.4	1.2	889.3	1.7	159.2	3.4	1 438.7
MLC	●	●	●	119.3	—	2.6	—	2.1	—	10.6	4 543.0
MCR-hf	●	●	●	28.1	157.7	1.0	483.9	0.7	25.6	2.9	699.4
MCR-hb	●	●	●	21.1	115.2	0.7	297.4	0.5	19.7	2.1	456.7
MCR-hs	●	●	●	25.1	97.3	0.9	322.2	0.6	16.8	2.6	466.1
MCR-ht	●	●	●	20.2	86.8	0.7	246.4	0.5	17.4	2.0	373.6
MCR-t5	●	●	●	31.5	318.4	0.5	348.6	1.7	157.2	2.7	891.9
MCR-t10	●	●	●	31.6	326.2	0.5	913.7	1.7	158.5	2.7	1 467.6
MCR-t10-r15	●	●	●	20.0	207.5	0.3	554.0	1.2	103.6	1.7	885.0
MR-t10	●	●	○	14.4	9.4	0.2	9.5	0.3	1.6	1.2	22.2
MR-∞	●	●	○	14.2	10.0	0.5	31.0	0.3	1.8	1.4	44.4
MR-0	●	●	○	21.4	13.9	0.7	42.5	0.4	2.4	2.1	61.5
MR-10	●	●	○	9.7	6.3	0.5	30.5	0.2	1.3	1.1	39.4
MR-(0,∞)	●	●	○	35.6	23.3	1.1	76.7	0.6	4.3	3.5	108.8

B.2 Additional Inputs

In addition to London, we tested inputs representing other large metropolitan areas (New York, Los Angeles, and Chicago). We built the public transit network from publicly available General Transit Feeds (GTFS), restricting ourselves to the timetable for August 10, 2011 (a Wednesday). The road network data is still given by PTV, and these instances do not include bicycles. Detailed statistics for all instances are presented in Table 4.

Table 5 compares the performance of our algorithms on these inputs. For reference, we also consider a simplified version of the London network, without bicycles. For each input, we show the average values (over 1 000 queries) for number of journeys found, running time, and quality (considering the top 6 journeys). The results are consistent with those obtained for the full London network, showing that our preferred choice of heuristics also holds here. MCR-hb is always the best choice in terms of solution quality, while MR-t10 is preferred if query times should be as low as possible.

Table 4. Size figures for our input instances. We link every stop and cycle station with the walking/road network.

Figure	London	New York	Los Angeles	Chicago
Public Transit				
Stops	20 843	17 894	15 003	12 137
Routes	2 184	1 393	1 099	710
Trips	133 011	45 299	16 376	20 303
Daily Departure Events	4 991 125	1 825 129	931 846	1 194 571
Vertices (Route Model)	99 230	66 124	81 657	47 561
Edges (Route Model)	260 583	193 159	214 369	118 452
Walking				
Vertices	258 840	255 808	224 053	70 440
Vertices in Core	27 840	25 808	21 053	16 440
Edges	1 433 814	1 586 782	1 395 185	586 979
Footpaths $\leq$ 5 min	150 948	219 040	83 844	122 450
Footpaths $\leq$ 10 min	518 174	670 702	271 444	426 818
Cycling				
Cycle Stations	564	—	—	—
Vertices	23 311	—	—	—
Vertices in Core	1 311	—	—	—
Edges	130 971	—	—	—
Taxi				
Vertices	259 122	263 407	233 612	72 062
Vertices in Core	27 122	24 407	18 612	16 062
Edges	1 339 487	1 502 924	1 343 471	583 876

Table 5. Evaluating the performance of MCR and MR with different heuristics on other instances. The quality is determined identically to Table 1 (cf. Section 5).

				London No Bike			**New York**			**Los Angeles**			**Chicago**		
Algorithm	Arr.	Trp.	Wlk.	# Jn.	Time [ms]	Qual. Avg.	# Jn.	Time [ms]	Qual. Avg.	# Jn.	Time [ms]	Qual. Avg.	# Jn.	Time [ms]	Qual. Avg.
MCR	●	●	●	27.5	1 215.9	100 %	25.5	1 703.0	100 %	16.7	644.6	100 %	22.1	532.8	100 %
MCR-hf	●	●	●	10.5	677.3	89 %	8.6	611.0	91 %	8.9	445.0	88 %	8.3	241.3	72 %
MCR-hb	●	●	●	8.7	430.3	91 %	7.2	413.8	94 %	7.6	295.8	93 %	7.1	160.8	92 %
MCR-hs	●	●	●	8.5	450.6	68 %	6.7	414.0	84 %	7.4	310.7	62 %	6.6	158.8	58 %
MCR-ht	●	●	●	8.3	342.6	81 %	6.6	300.9	80 %	6.7	228.4	69 %	6.2	113.9	79 %
MCR-t5	●	●	●	27.3	671.7	94 %	25.6	695.5	69 %	16.6	262.7	93 %	21.9	277.7	95 %
MCR-t10	●	●	●	27.4	1 123.0	96 %	25.3	1 401.4	85 %	16.8	424.5	96 %	22.0	578.8	98 %
MCR-t10-r15	●	●	●	11.9	688.1	28 %	5.4	677.9	10 %	3.9	202.0	13 %	9.6	372.7	28 %
MR-t10	●	●	○	4.4	19.7	61 %	3.6	10.6	60 %	3.6	11.0	51 %	3.3	7.1	63 %
MR-∞	●	●	○	4.4	40.0	61 %	3.4	26.3	65 %	3.6	21.5	51 %	3.3	12.3	63 %
MR-0	●	●	○	5.2	55.7	61 %	3.8	37.6	65 %	4.3	28.5	52 %	3.7	15.6	63 %
MR-10	●	●	○	6.1	36.8	43 %	6.0	26.1	41 %	6.1	26.6	42 %	5.1	13.9	50 %
MR-(0,∞)	●	●	○	9.6	99.3	64 %	7.3	69.6	65 %	8.0	54.0	53 %	7.0	29.6	64 %