Project Number 288094

eCOMPASS
eCO-friendly urban Multi-modal route PlAnning Services for mobile uSers

STREP
Funded by EC, INFSO-G4(ICT for Transport) under FP7

eCOMPASS – TR – 022

Computing Multimodal Journeys in Practice

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

June 2013

Computing Multimodal Journeys in Practice*

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

[1] Microsoft Research Silicon Valley, Mountain View, CA 94043, USA.
{dadellin,renatow}@microsoft.com
[2] Karlsruhe Institute of Technology (KIT), 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 much faster, making the approach practical even for large metropolitan areas.

1 Introduction

Efficiently computing good journeys in transportation networks has been an active area of research in recent years, with focus on the computation of routes in both road networks [11] and schedule-based public transit [2, 5], but these are often considered separately. In practice, users want an integrated solution to find the "best" journey considering all available modes of transportation. Within a metropolitan area, this includes buses, trains, driving, cycling, taxis, and walking. We refer to this as the *multimodal route planning* problem.

In fact, any public transportation network has a multimodal component, since journeys require some amount of walking. To handle this, existing solutions [4, 10, 14] predefine transfer arcs between nearby stations, then run 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* of non-dominating journeys optimizing multiple criteria, which is practical even for large metropolitan areas [10].

Extending public transportation solutions to a full multimodal scenario (with unrestricted walking, biking, and taxis) may seem trivial: one could just incorporate routing techniques for road networks [9, 17] to solve the new subproblems.

* Partial support by DFG grant WA654/16-1 and EU grant 288094 (eCOMPASS).

Unfortunately, meaningful multimodal optimization must 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 to the airport is a good solution for some; users on a budget may prefer cheaper alternatives. Considering more criteria leads to much larger Pareto sets, however, with many of the additional journeys looking unreasonable (see full paper [8]).

Previous research thus tends to avoid multicriteria search altogether [3], 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 [19]. When extended to find the k-shortest paths [6], this method can even take user preferences into account. Unfortunately, linear combination may miss Pareto-optimal journeys [7] (also see full paper [8]). To avoid such issues, another line of multimodal single-criterion research considers label-constrained quickest journeys [1]. Here, journeys are required to obey a user-defined pattern, typically enforcing a hierarchy of modes [6] (such as "no car travel between trains"). Although this approach can be quite fast when using preprocessing techniques for road networks [12], it has a fundamental conceptual problem: it 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 [15], we argue in Section 2 that most users optimize three criteria: travel time, convenience, and costs. As this produces a large Pareto set, we propose using fuzzy logic [20] to identify, in a principled way, a modest-sized subset of representative journeys. This postprocessing step is very quick and can incorporate personal preferences. As Section 3 shows, we can use recent algorithmic developments [10, 12, 17] 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 more criteria, such as costs, are incorporated. Section 4 proposes heuristics (still multicriteria) that are significantly faster and closely match the representative journeys in the actual 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 dynamic scenarios.

2 Problem Statement

We want to find journeys in a network built from several *partial networks*. The first is a *public transportation network* representing all available schedule-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. Each stop may also keep a *minimum change time*, which must be obeyed when changing trips. 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 to traverse it. The *integrated transportation network* is the union of these partial networks with appropriate *link vertices*, i. e., vertices (or stops) in different networks are identified with one another to allow for changes in modes of transportation. Note that, unlike previous work [18], we do not necessarily require explicit *footpaths* in the public transportation networks (to walk between nearby stops). A query takes as input a *source location* s, a *target location* t, and a *departure time* τ, and it produces *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 timetable constraints.

We still have to define *which* journeys the query should return. We argue that users optimize three natural criteria in multimodal networks: arrival time, costs, and "convenience". For our first (simplified) scenario (with public transit, cycling, and walking, but no taxi), we work with three criteria. Besides arrival time, we use number of trips and walking duration as proxies for convenience. We add cost for the scenario that includes taxi. 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. If two journeys have equal values in all criteria, we only keep one.

Solving the full multicriteria problem, however, can lead to solution sets that are too large for most users. Moreover, many solutions provide undesirable tradeoffs, such as journeys that arrive much later to save a few seconds of walking (or walk much longer to save a few seconds in arrival time). Intuitively, most criteria are diffuse to the user, and only large enough differences are significant. Pareto optimality fails to capture this. To formalize the notion of significance, we propose to *score* the journeys in the Pareto set in a post-processing step using concepts from fuzzy logic [20]. Loosely speaking, fuzzy logic generalizes Boolean logic to handle (continuous) degrees of truth. For example, the statement "60 and 61 seconds of walking are equal" is false in classical logic, but "almost true" in fuzzy logic. Formally, 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 $\mathcal{S}$. Mostly, we use μ to refer to $\mathcal{S}$. Our application requires fuzzy relational operators $\mu_<$, $\mu_=$, and $\mu_>$. For any $x, y \in \mathbb{R}$, they are evaluated by

$\mu_<(x-y)$, $\mu_>(y-x)$, and $\mu_=(x-y)$. We use the well-known [20] exponential membership functions for the operators: $\mu_=(x) := \exp(\frac{\ln(\chi)}{\varepsilon^2}x^2)$, 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). Moreover, we require binary operators (norms) $T, S : [0,1]^2 \to [0,1]$ to represent fuzzy (logical) disjunction (T) and conjunction (S). We use the *maximum/minimum norms*, i. e., $T = \max$ and $S = \min$. Note that $S(x,y) = 1 - T(1-x, 1-x)$ holds, which is important for consistency. Other norms are evaluated in the full paper [8].

We now recap the concept of fuzzy dominance in multicriteria optimization, which is introduced by Farina and Amato [16]. 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 each criterion may use different fuzzy operators.) Analogously, we define $n_e(J_1, J_2)$ for equality and $n_w(J_1, J_2)$ for greater-than. By definition, $n_b + n_e + n_w = M$. Hence the Pareto dominance can be generalized to obtain a *degree of domination* $d(J_1, J_2) \in [0,1]$, 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 $J_1, \ldots, J_n$, 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, $\mathrm{sc}(J) := 1 - \max(J_1, \ldots, J_n)$, i. e., the value $\mathrm{sc}(J)$ is determined by the (one) journey that dominates J most. See the full paper [8] for more details, including an illustration of the fuzzy dominance function d. 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

We now study exact algorithms for the multicriteria multimodal problem. We first propose two solutions (building on different methods for multicriteria optimization on public transportation networks), then describe an acceleration technique that applies to both. For simplicity, we describe the algorithms considering only the (schedule-based) public transit network and the (unrestricted) walking network. We later deal with cycling and taxis, which are unrestricted but have special properties.

Multi-label-correcting Algorithm. Traditional solutions to the multicriteria problem on public transportation networks typically model the timetable as a graph. A particularly effective approach is to use the *time-dependent route model* [18]. 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. We also add *route edges* between route vertices associated to consecutive stops within the same route. To model the trips along a route, the cost of a route edge is given by a function reflecting the traversal time (including waiting for the next departure).

A journey in the public transportation network corresponds to a path in this graph. The *multi-label-correcting* (MLC) [18] algorithm uses this to find full Pareto sets for arbitrary criteria that can be modeled as edge costs. MLC extends Dijkstra's algorithm [13] by operating on labels that have multiple values, one per criterion. Each vertex v maintains a *bag* $B(v)$ 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, MLC 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. MLC can be sped up with target pruning and by avoiding unnecessary domination checks [14].

To solve the multimodal problem, we extend MLC by augmenting its input graph to include the walking network, creating an integrated network. The MLC query remains essentially unchanged. Although labels can now be associated to vertices in different networks, they can all share the same priority queue.

Round-based Algorithm. A drawback of MLC (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. [10] have recently introduced RAPTOR (*Round bAsed Public Transit Optimized Router*) as a faster alternative. The simplest version of the algorithm optimizes two criteria: arrival time and number of transfers. Unlike MLC, which searches a graph, RAPTOR uses dynamic programming to operate directly on the timetable. It works in rounds, with round i processing all relevant journeys with exactly $i-1$ 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 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 to nearby (predefined) stops in a second phase. Simpler data structures and better locality make RAPTOR an order of magnitude faster than MLC. Delling et al. [10] have also proposed McRAPTOR, which extends RAPTOR to handle more criteria (besides arrival times and number of transfers). It maintains a *bag* (set) of labels with each stop and round.

Even with multiple modes of transport available, one trip always consists of a single mode. This motivates adapting the round-based paradigm to our scenario. 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 (on the timetable network) and MLC for the second (on the walking network). Labels generated by one phase are naturally used as input to the other. During the second phase, MLC extends bags instead of individual labels. To ensure that each label is processed at most once, we keep track of which labels (in a bag) have already been extended. 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. During round i, the McRAPTOR subroutine reads labels from round $i-1$ and writes to round i. In contrast, the MLC subroutine may read and write labels of the same round if walking is not regarded as a trip.

Contracting Unrestricted Networks. As our experiments will show, the bottleneck of the multimodal algorithms is processing the walking network $G = (V, A)$. We improve performance using a quick preprocessing technique [12]. For any journey involving public transportation, walking between trips always begins and ends at the restricted set $K \subset V$ of link vertices. During queries, we must only be able to compute the pairwise distances between these vertices. We therefore use preprocessing to compute a smaller *core graph* that preserves these distances. More precisely, we start from the original graph and iteratively *contract* [17] each vertex in $V \setminus K$ in the order given by a rank function r. Each contraction step (temporarily) removes a vertex and adds shortcuts between its uncontracted neighbors to maintain shortest path distances (if necessary). It is usually advantageous to first contract vertices with relatively small degrees that are evenly distributed across the network [17]. We stop contraction when the average degree in the core graph reaches some threshold (we use 12 in our experiments) [12].

To run a faster multimodal s–t query, we use essentially the same algorithm as before (based on either MLC or RAPTOR), but replacing the full walking network with the (smaller) core graph. Since the source s and the target t may not be in the core, we handle them during initialization. It works on the graph $G^+ = (V, A \cup A^+)$ containing all original arcs A as well as all shortcuts A^+ added during the contraction process. We run upward searches (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 of the core, but arcs within the core are not processed [12]. These core vertices (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 works as before, with one minor adjustment. Whenever MLC extracts a label $L(v)$ for a scanned core vertex v, we check if it has been reached by the reverse search during initialization. If so, we create a temporary label $L'(t)$ by extending $L(v)$ with the (already computed) walking path to t and add it to $B(t)$ if needed. MCR is adjusted similarly, with bags instead of labels.

Beyond Walking. We now consider other unrestricted networks (besides walking). In particular, our experiments include a bicycle rental scheme, which can be seen as a hybrid network: it does not have a fixed schedule (and is thus unrestricted), but bicycles can only be picked up and dropped off at designated *cycling stations.* Picking a bike from its station counts as a trip. 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 consider a taxi ride to be a trip as well, since we board a vehicle. Moreover, we also optimize a separate criterion reflecting the (monetary) *cost* of taxi rides. 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. Our round-based algorithms handle taxis as they do walking, except that in the taxi stage labels are read from round $i-1$ and written into round i. Note that we link the taxi network to public transit stops and bicycle stations.

Dealing with personal cars or bicycles is simpler. Assuming that they are only available for the first or last legs of the journey, we must only consider them during initialization. Initialization can also handle other special cases, such as allowing rented bicycles to be ridden to the destination (to be returned later).

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

4 Heuristics

Even with all accelerations, the exact algorithms proposed in Section 3 are not fast enough for interactive applications. This section proposes quick heuristics aimed at finding a set of journeys that is similar to the exact solution, which we take as ground truth. We consider three approaches: weakening the dominance rules, restricting the amount of walking, and reducing the number of criteria. We also discuss how to measure the quality of the heuristic solutions we find.

Weak Dominance. The first strategy we consider is to weaken the domination rules during the algorithm, reducing the number of labels pushed through the network. We test four 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 dominance, 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*); this can be extended to use buckets for several criteria. The third heuristic, MCR-hs(κ), uses strict dominance 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, weakens 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 (we use $a = 0.3$).

Restricting Walking. 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. This motivates a second class of heuristics, MCR-tx. It still runs three-criterion search (walking, arrival, and trips), but limits walking transfers between stops to x minutes; in this case we precompute these transfers. MCR-tx-ry also limits walking in the beginning and end to y minutes. Note that existing solutions often use such restrictions [4].

Fewer Criteria. The last strategy we study is reducing the number of criteria considered during the algorithm. As already mentioned, this is a common approach in practice. We propose MR-x, which still works in rounds, but optimizes only the number of trips and arrival times explicitly (as criteria). To account for walking duration, we count every x minutes of a walking segment (transfer) as a trip; the first x minutes are free. With this approach, we can 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 special case is $x = \infty$, where a transfer is never a trip. Another variant is to always count a transfer as a single trip, regardless of duration; we abuse notation and call this variant MR-0. We also consider MR-∞-tx: walking duration is not an explicit criterion and transfers do not count as trips, but are limited to x minutes.

For scenarios that include cost as a criterion (for taxis), we consider variants of the MCR-hb and MCR-hf heuristics. In both cases, we drop walking as an independent criterion, leaving only arrival time, number of trips, and costs to optimize. We account for walking by making it a (cheap) component of the costs.

Quality Evaluation. To measure the quality of a heuristic, we compare the set of journeys it produces to the *ground truth*, which we define as the solution found by MCR. To do so, we first compute the score of each journey with respect to the Pareto set that contains it (cf. Section 2). 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 only on the algorithm itself and does not assume knowledge of the ground truth, which is consistent with a real-world deployment. 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. 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 total similarity between J_1 and J_2 as $\min(c^1, c^2, \ldots, c^M)$. After computing the pairwise similarities, we greedily select the unmatched pairs with highest similarity (by picking the highest entry in the matrix that does not share a row or column with a previously picked entry). The similarity of the whole matching is the average similarity of its pairs, weighted by the fuzzy score of the reference journey. This means that matching the highest-scored reference journey is more important than matching the k-th one.

5 Experiments

All algorithms from Sections 3 and 4 were implemented in C++ and compiled with `g++` 4.6.2 (64 bits, flag `-O3`). We ran our experiments on one core of a dual 8-core Intel Xeon E5-2670 clocked at 2.6 GHz, with 64 GiB of DDR3-1600 RAM.

We focus on the transportation network of London (England); results for other instances (available in the full paper [8]) are similar. We use the timetable information made available by Transport for London (TfL), from which we extracted a Tuesday in the periodic summer schedule of 2011. The data includes subway (tube), buses, tram, and light rail (DLR), as well as bicycle station locations. To model the underlying road network, we use data provided by PTV AG from 2006, which explicitly indicates whether each road segment is open for driving, cycling and/or walking. We set the walking speed to 5 km/h and the cycling speed to 12 km/h, and we assume driving at free-flow speeds. We do not consider turn costs, which are not defined in the data. The resulting combined network has 564 cycle stations and about 20 k stops, 5 M departure events, 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 $x = \varepsilon$. 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. Our approach is robust to small variations in these parameters, but they can be tuned to account for user-dependent preferences. We run *location-to-location* queries, with sources, targets, and departure times picked uniformly at random (from the walking network and during the day, respectively).

For our first experiment, we use walking, cycling, and the public transportation network and consider three criteria: arrival time, number of trips, and walking duration. We ran 1 000 queries for each algorithm. Table 1 summarizes the results (the full paper [8] has additional statistics). For each algorithm, the table first shows which criteria are explicitly taken into account. The next five columns show the average 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 columns evaluate the quality of the top 3 and 6 journeys found by our heuristics, as explained in Section 4. We show both averages and standard deviations.

The methods in Table 1 are grouped in blocks. Those in the first block compute the full Pareto set considering all three criteria (arrival time, number of trips, and walking). MCR, our reference algorithm, is round-based and uses contraction in the unrestricted networks. As anticipated, it is faster (by a factor of about three) than MCR-nc (which does not use the core) and MLC (which

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

					Scans	Comp.		Time	Quality-3		Quality-6	
Algorithm	Arr.	Trp.	Wlk.	Rnd.	/ Ent.	/ Ent.	Jn.	[ms]	Avg.	Sd.	Avg.	Sd.
MCR-nc	●	●	●	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-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 %
MCR-t5	●	●	●	13.8	2.7	126.6	28.9	891.9	93 %	16 %	92 %	15 %
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-∞-t10	●	●	○	7.6	1.1	4.8	4.5	22.2	63 %	28 %	62 %	24 %

uses the core but is not round-based). Accordingly, all heuristics we test 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. All heuristics are faster than pure MCR, and MCR-hb gives the best quality at a reasonable running time.

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 expected in a well-designed public transportation network). Moreover, adding precomputed footpaths of 10 minutes is not faster than using the core for unlimited walking (as MCR does). Additionally limiting the walking range from s or t (MCR-t10-r15) improves speed, but the 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 a similar speedup with much better quality (comparable to MCR-hb).

The MR-x algorithms (fourth block) reduce the number of criteria considered by combining trips and walking. The fastest variant is MR-∞-t10, which drops walking duration as a criterion but limits the amount of walking at transfers to 10 minutes, making it essentially the same as RAPTOR, with a different initialization. As expected, however, quality is much lower than for MCR-tx, confirming

Table 2. Performance on our London instance when taking taxi into account.

Algorithm	Arr.	Trp.	Wlk.	Cost	Rnd.	Scans / Ent.	Comp. / 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 %

that considering the walking duration explicitly during the algorithm is important to obtain a full range of solutions. MR-10 attempts to improve quality by transforming long walks into extra trips, but is not particularly successful.

Summing up, MCR-hb should be the preferred choice for high-quality solutions, while MR-∞-t10 can support interactive queries with reasonable quality.

Our second experiment considers the full multimodal problem, including taxis. We add *cost* as fourth criterion (at 2.40 pounds per taxi-trip plus 60 pence per minute). We do not consider the cost of public transit, since it is significantly cheaper. Table 2 presents the average performance of some of 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). While exact MCR is 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 shows that we can reduce running times by dropping walking duration as a 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 time to about 1 second, with reasonable quality.

6 Final Remarks

We have studied multicriteria journey planning in multimodal networks. We argued that users optimize three criteria: arrival time, costs, and convenience. Although the corresponding full Pareto set is large and has 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. Our experiments show that our approach enables efficient realistic multimodal journey planning in large metropolitan areas. A natural avenue for future research is accelerating our approach further to enable interactive queries with an even richer set of criteria in dynamic scenarios, handling delay and traffic information. The ultimate goal is to compute multicriteria multimodal journeys on a global scale in real time.

References

1. C. Barrett, K. Bisset, M. Holzer, G. Konjevod, M. V. Marathe, and D. Wagner. Engineering Label-Constrained Shortest-Path Algorithms. In *The Shortest Path Problem: 9th DIMACS Impl. Challenge*, DIMACS 74, pp. 309–319. AMS, 2009.
2. H. Bast. Car or Public Transport – Two Worlds. In S. Albers, H. Alt, and S. Näher, editors, *Efficient Algorithms*, LNCS 5760, pp. 355–367. Springer, 2009.
3. H. Bast. Next-Generation Route Planning: Multi-Modal, Real-Time, Personalized, 2012. Talk given at ISMP.
4. 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 *ESA*, LNCS 6346, pp. 290–301. Springer, 2010.
5. R. Bauer, D. Delling, and D. Wagner. Experimental Study on Speed-Up Techniques for Timetable Information Systems. *Networks*, 57(1):38–52, 2011.
6. M. Bielli, A. Boulmakoul, and H. Mouncif. Object modeling and path computation for multimodal travel systems. *EJOR*, 175(3):1705–1730, 2006.
7. D. Corne, K. Deb, P. Fleming, and J. Knowles. The Good of the Many Outweighs the Good of the One: Evolutionary Multi-Objective Optimization. *Connections*, 1(1):9–13, 2003.
8. D. Delling, J. Dibbelt, T. Pajor, D. Wagner, and R. F. Werneck. Computing and Evaluating Multimodal Journeys. Technical Report 2012-20, Faculty of Informatics, Karlsruhe Institute of Technology, 2012.
9. D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable Route Planning. In *SEA*, LNCS 6630, pp. 376–387. Springer, 2011.
10. D. Delling, T. Pajor, and R. F. Werneck. Round-Based Public Transit Routing. In *ALENEX*, pp. 130–140. SIAM, 2012.
11. D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering Route Planning Algorithms. In *Algorithmics of Large and Complex Networks*, LNCS 5515, pp. 117–139. Springer, 2009.
12. J. Dibbelt, T. Pajor, and D. Wagner. User-Constrained Multi-Modal Route Planning. In *ALENEX*, pp. 118–129. SIAM, 2012.
13. E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.
14. Y. Disser, M. Müller–Hannemann, and M. Schnee. Multi-Criteria Shortest Paths in Time-Dependent Train Networks. In *WEA*, LNCS 5038, 347–361. Springer, 2008.
15. A. Ensor and F. Lillo. Partial order approach to compute shortest paths in multimodal networks. Technical report, http://arxiv.org/abs/1112.3366v1, 2011.
16. M. Farina and P. Amato. A Fuzzy Definition of "Optimality" for Many-Criteria Optimization Problems. *IEEE Tr. Syst., Man, and Cyb. A*, 34(3):315–326, 2004.
17. R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transp. Sci.*, 46(3):388–404, 2012.
18. M. Müller–Hannemann, F. Schulz, D. Wagner, and C. Zaroliagis. Timetable Information: Models and Algorithms. In *Algorithmic Methods for Railway Optimization*, LNCS 4359, pp. 67–90. Springer, 2007.
19. P. Modesti and A. Sciomachen. A Utility Measure for Finding Multiobjective Shortest Paths in Urban Multimodal Transportation Networks. *EJOR*, 111(3):495–508, 1998.
20. L. A. Zadeh. Fuzzy Logic. *IEEE Computer*, 21(4):83–93, 1988.