Result Diversity for Multi-Modal Route Planning*

Hannah Bast[1], Mirko Brodesser[1], and Sabine Storandt[1]

1   Albert-Ludwigs-Universität Freiburg
    Freiburg, Germany
    {bast,brodessm,storandt}@informatik.uni-freiburg.de

—— Abstract ——

We study multi-modal route planning allowing arbitrary (meaningful) combinations of public transportation, walking, and taking a car / taxi. In the straightforward model, the number of Pareto-optimal solutions explodes. It turns out that many of them are similar to each other or unreasonable. We introduce a new filtering procedure, *Types aNd Thresholds (TNT)*, which leads to a small yet representative subset of the reasonable paths. We consider metropolitan areas like New York, where a fast computation of the paths is difficult. To reduce the high computation times, optimality-preserving and heuristic approaches are introduced. We experimentally evaluate our approach with respect to result quality and query time. The experiments confirm that our result sets are indeed small (around 5 results per query) and representative (among the reasonable Pareto-optimal paths), and with average query times of about one second or less.

1   Introduction

We want to efficiently compute small yet representative sets of reasonable paths in a multi-modal scenario (car, walking, transit). The majority of current route planning systems computes optimal paths for a certain type of transportation. If one wants to take a car, one uses a navigation system. If one wants to travel by public transportation, one can obtain the optimal paths from websites like Google Maps. Both require to decide in advance for a means of transportation. For the case when one does not want to decide this beforehand, we want to offer the user a small yet representative set of reasonable paths. Moreover, we allow optimal paths which include different means of transportation. Furthermore, also for metropolitan areas like New York, computation should work fast, such that interactive queries are possible. For road networks, state-of-the-art algorithms answer shortest path queries in the order of milliseconds [9]. Public transportation networks are more challenging and many algorithms of road networks are not applicable [1]. Computing optimal paths becomes more complex, since not only the fastest connection is demanded, but also the number of transfers is an important criterion for the quality of a path, potentially leading to multiple optimal paths. When combining road and transit networks, this increases complexity further, bringing along many similar paths. Typical variations are: *"take a bus for 30 minutes, then take a taxi for 9 minutes", "take a bus for 32 minutes, then take a taxi for 8 minutes", . . . , "take a bus for 42 minutes, then take a taxi for 3 minutes"*. To determine a small yet representative set of reasonable paths, it is necessary to filter, which is a challenge on its own. In the following we introduce an approach to deal with these challenges.

1.1   Contribution

We propose *Types aNd Thresholds*, an approach to efficiently compute small yet representative sets of reasonable paths in a multi-modal scenario (car, walking, transit). To obtain diverse sets of paths, we use Pareto sets [11] with multiple optimization criteria. Taking into account properties (velocity, availability, costs) of the various means of transportation, we argue that not all Pareto optimal paths are reasonable. We carefully define types of reasonable paths and propose a two-stage filtering procedure. In the first stage, all unreasonable paths are removed. In the second stage, a small yet representative subset of the remaining paths is determined. To achieve average query durations of roughly one second, we make use of properties of the types and propose a relaxation of the model and a (close to optimal) heuristic. We confirm query durations and quality with experimental results.

2   Related Work

When considering road and transit networks separately, many algorithms exist for both networks. Next, we give a brief overview. For road networks several variants of the famous Dijkstra algorithm exist. An outstanding one is Contraction Hierarchies [9], which after a brief precomputation enables to quickly answer queries, even for large areas (e.g., Europe on the order of milliseconds). Fast routing on transit networks requires other approaches and algorithms. Among them are a multi-criteria generalization of Dijkstra's algorithm [8], relaxed Pareto dominance [13] and Round-Based Public Transit Routing [6]. One state-of-the-art algorithm is to compute transfer patterns [2] between all pairs of stations. A transfer pattern is the sequence of stations where vehicle changes occur on an optimal path. For each pair of stations these are few and allow to answer queries efficiently. However, the computation of patterns is time-consuming.

Also for multi-modal networks several approaches exist. However, many of those are quite limited with respect to the extent of their multi-modality. For instance, [5] limits car usage to the beginning and end of journeys. Other approaches [12] compute a single optimal path by combining multiple criteria to one, but this results in missing reasonable paths (see [3] for an example). Further approaches [16, 7] expect the user to specify a hierarchy of modes (e.g. between train usage, car is forbidden). The problem is that one has to know the constraints in advance, which in practice is often not the case.

A less restricted approach focussing on computing multiple optimal paths in a multi-modal scenario similar to ours was presented in [4]. To not miss reasonable paths, multiple criteria with Pareto sets [11] are used. As this leads to numerous optimal paths, fuzzy filtering is used to rank them according to scores. For the found set of paths $P = \{p_1, \ldots, p_n\}$, the score of p_i is dependent on P and a measure for fractional dominance between a path $p \in P$ and the paths in $P \backslash \{p\}$. However, for the measures used in [4], the set of the top-k paths is not necessarily representative when k is small, and no experiments are provided on this quality aspect in [4]. In short, it is not clear if and how small yet representative sets of optimal paths can be determined with this approach.

3   Preliminaries

In this section, we describe how to model a multi-modal network and discuss the necessity for multiple optimality criteria. We briefly recapitulate Contraction Hierarchies as a speed-up technique and explain how to use existing algorithms to compute optimal paths in our setting.

3.1   Modelling

In the following we describe separate models for road and transit networks and how to combine them to a multi-modal model. To model the static road networks (car, walking), we use the common approach of one node per location (given as longitude, latitude) and time-independent arcs annotated with the duration to travel from one node to the other. For simplicity, we ignore turn restrictions and assume that all roads can be traveled in both directions by car and by foot. That is, the car and walking network have the same structure.

To model the transit network, we decided for a variation of the *train-route* model as explained in [15]. In the following we provide basic definitions and describe the model. A transit connection starting at a specific time at a specific station and ending at some station is called a *trip*. Trips sharing the same stop sequence and not overtaking each other are grouped as a *line*. For each stop, a *station arrival* and a *station departure* node are created. For each line, for each of its stops, a *line arrival* node and a *line departure* node are created. The nodes of a line are connected according to their stop sequence and the durations of the arcs are time-dependent. When boarding a line, the transfer buffer is added (we chose 5 minutes). Station arrival and station departure nodes are connected with their geographically closest car and walking node. We call these nodes *link* nodes. That is, each station arrival node has outgoing arcs to the closest car and walking node and they have outgoing arcs to the station departure node.

Unlike the model used in [4], our model prohibits to change from car to walking (and vice versa). The intention is that when going to a station by car, one does not stop on the way and walk the rest (or the other way around). Instead, to reach a station one either takes the car or walks. We consider this reasonable, since taking the car and walking is possible from and to all stations in our model. In practice, walking a short distance to or from a car is no problem, as we can consider this part of the transfer buffer. Note that car usage is *not* limited to the beginning and end of a journey, but is also allowed between taking two public transportation vehicles. We chose this model, because we consider it the most efficient in terms of query duration. Note that our filtering approach, which we describe in Section 4, is independent of the used model.

3.2   Optimality Criteria

In the following we explain the necessity for multiple optimality criteria to compute diverse sets of paths in our multi-modal scenario. When referring to *duration*, we mean the duration to reach the target, given a fixed departure time. Using duration as a single criterion would result in exactly one path. Therefore, we use multiple criteria with Pareto sets. Each criterion corresponds to one entry in a tuple. For tuples t_1, t_2, tuple t_1 is said to *dominate* t_2 if t_1 is at least as good as t_2 with respect to all criteria. Two tuples are called *incomparable* if none of them dominates the other. A Pareto set is a maximal set of non-dominating tuples. A *label* contains such a tuple and is associated with a predecessor label. In a multi-criteria Dijkstra, each node contains a Pareto set of labels.

For transit networks, duration and transfer penalty (= number of boarded vehicles) are two commonly used Pareto criteria. However, in our setting this almost always leads to exactly two optimal paths: walking the whole way and using the car for the whole way. The reasons are: Taking a car is very fast and in our model is available everywhere and boarding it once yields a transfer penalty of one, therefore it usually dominates all paths which include transit usage. Walking the whole way is incomparable to using the car, because it is slower and has zero transfers.

Therefore, we use car duration as another Pareto criterion, leading to a diverse set of optimal paths. However, solutions then become too numerous and many of them are similar. A typical variation is the one mentioned in the introduction, where paths differ only slightly in the car duration. In one of the scenarios from [4], they use arrival time (equivalent to our duration), number of transfers, walking duration, and taxi cost as Pareto criteria.

3.3   Contracting the Road Networks

As the majority of nodes are car and walking nodes (see Table 4 for details), optimizing the routing on the road network is important to reduce computation times. A well-known speed-up technique applicable to road networks is Contraction Hierarchies [9]. During shortest path queries it allows to skip many nodes, hence unnecessary propagation of labels is avoided. Recently, a variant [4] for a multi-modal scenario similar to ours was introduced, we refer to this as *contracting the road network*. Next, we summarize its most important properties. After an efficient precomputation, all nodes in the road network have a rank. There exists a *core* of nodes which comprises all link nodes and the rank of these nodes is infinity. For queries from all road network nodes the following holds: when ignoring arcs to nodes with lower rank, distances to all link nodes are equal to those in the original road network. As a special case distances between all pairs of link nodes are preserved. In the next section, we describe how to use these properties to efficiently compute shortest paths.

3.4   Computing Multi-Criteria Optimal Paths

In the following we explain how to perform location-to-location queries. We call the graph with inverted arc directions *backwards graph*. Given the road network is contracted as mentioned above (one contraction for usage by car, and one contraction for usage by walking), location-to-location queries are performed in two steps.

First, a query from the source to all nodes and a query (in the backwards graph) from the target to all nodes are performed. Recall that during the contraction process road network nodes were assigned a rank. For both queries, arcs to nodes with lower rank are ignored. We call the duration of walking (taking the car) from the source to the target, *pure* walking (car) duration.

Second, a multi-criteria Dijkstra initialized with the labels of the link nodes reached from the source is run. Temporary arcs with the previously computed durations from the link nodes to the target are added. Again, arcs to nodes with lower rank are ignored. The Pareto set of pure car and walking duration and the labels at the target forms the result. Dominance by early results and label forwarding [8] are used to accelerate query computation.

Note that this is essentially one of the query algorithms proposed in [4].

4   Types aNd Thresholds

In this section we describe the concept of *Types aNd Thresholds (TNT)* to obtain small but representative sets of reasonable paths. We present speed-up techniques to reduce query times towards practical usage. We start by introducing the idea of discretization which leads to TNT.

4.1   Discretization

Using duration, transfer penalty and car duration as Pareto criteria leads to numerous optimal paths, among which many are similar. To filter out a more concise subset, we
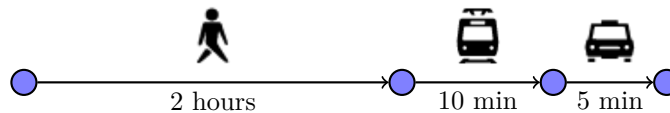
examine the post-processing step of *discretizing* car duration to certain blocks (for example, ten minutes). Table 1 shows an excerpt of the results of a query on New York.

Table 1 Excerpt of the tuples of the optimal paths of an example query using duration, transfer penalty and car duration as Pareto criteria. Green tuples are still Pareto optimal after the discretization, gray ones are not.

duration	transfer penalty	car duration	discretized car duration
0:28:57	1	0:28:57	0:30:00
		. . .	
1:43:43	3	0:16:35	0:20:00
1:44:01	3	0:16:26	0:20:00
1:44:09	3	0:16:04	0:20:00
1:44:34	5	0:11:07	0:20:00
1:44:36	3	0:15:56	0:20:00
1:45:12	4	0:15:51	0:20:00
		. . .	
7:06:00	0	0	0:00:00

Discretization was also introduced in [4], however, it was used as a heuristic during query time to reduce computation complexity. Our motivation is different: given that many Pareto optimal solutions are similar, we use it to choose a representative subset.

Although discretization allows to reduce the number of Pareto-optimal solutions remarkably, unreasonable paths can remain. Consider the example in Figure 1. It is not very

Figure 1 An example for a path which we consider unreasonable.

meaningful to walk a long distance and then take a taxi for a short distance. We argue that, in practice, one would either walk the whole way, walk and take a train or use the car for the whole way. In the following we propose a new approach to filter out such unreasonable paths, and then to obtain a small representative subset of the remaining paths.

4.2 Types

With the example in Figure 1, we illustrated that some Pareto optimal solutions can be unreasonable. To justify why certain types of paths are unreasonable, we analyze triples of transit, walking and car duration with respect to their *relative durations (RD)*. We classify each possible triple as either reasonable or unreasonable. As relative durations we use the abstract terms *zero ($\bigcirc$)*, *little ($\odot$)* and *much ($\bullet$)*, hence $RD := \{\bigcirc, \odot, \bullet\}$. In Section 4.3 we provide concrete definitions. We assume the natural order of $\bigcirc < \odot < \bullet$. For example, the triple ($\bullet, \bigcirc, \odot$) represents paths with much transit usage, zero walking and little car usage.

Our rationale behind this (rather coarse) classification is as follows. Small differences in duration are of little practical concern to users, little is little. However, there is a difference between little and zero, because using a particular mode of transport at all incurs a tangible overhead (organizing a car/taxi, dealing with the circumstances of public transportation).

Once a certain mode of transportation is used more than little, it is reasonable to assume that one is willing to use it as much as is necessary to obtain an optimal solution. For example, if one is willing to use the car for one hour, one might as well use it for the whole trip if that is the fastest way. This is not necessarily true for walking (one might be willing to walk 1 hour but not 10 hours), however, that is not a problem in practice, because optimal paths rarely comprise very much walking (with the exception of the trivial walk-everything solution, which is always computed in our model).

As ⊙ and ● can be distinguished with respect to the total duration only if both occur in a triple, the set of all triples containing ⊙ but not ● is equivalent to the set of triples containing ● but not ⊙. Moreover, the triple without ⊙ and ●, that is (○, ○, ○), does not exist for real paths. Therefore, we call a triple *valid* iff at least one component is ●.

Table 2 All combinations of relative durations (zero = ○, little = ⊙, much = ●) for transit, car and walking duration with the classifications and violated axioms. White background indicates a relative duration triple is not valid. From the remaining triples, the ones classified reasonable are green, the others are red.

transit duration	walking duration	car duration	violated axiom	classification
○	○	○		✗
○	○	⊙		✗
○	○	●		✓
○	⊙	○		✗
○	⊙	⊙		✗
○	⊙	●	A1	✗
○	●	○		✓
○	●	⊙	A2	✗
○	●	●	A1	✗
⊙	○	○		✗
⊙	○	⊙		✗
⊙	○	●	A1	✗
⊙	⊙	○		✗
⊙	⊙	⊙		✗
⊙	⊙	●	A1	✗
⊙	●	○		✓
⊙	●	⊙	A2	✗
⊙	●	●	A1	✗
●	○	○		✓
●	○	⊙		✓
●	○	●	A1	✗
●	⊙	○		✓
●	⊙	⊙		✓
●	⊙	●	A1	✗
●	●	○		✓
●	●	⊙	A2	✗
●	●	●	A1	✗

Consider the properties of our model and of the different modes of transportation:

- Public transit is limited to stations and schedules, medium-fast and medium-expensive.
- Walking is possible everywhere at all times, slow and cheap.
- Cars (taxis) are available everywhere at all times, fast and expensive.

Given these properties, we claim the following axioms should hold for all reasonable paths:

- A1: Much car usage implies zero walking and zero transit usage.
- A2: Much walking implies zero car usage.

From the axioms we deduce which triples of relative durations are reasonable. Table 2 contains all triples, annotated with the classification as *reasonable (✔)* or *unreasonable (✗)*.

The classification is consistent in the sense that for each triple classified as reasonable, each valid component-wise smaller triple is classified reasonable, too. This can be inferred from Table 2. For instance, triple $(\bullet, \circ, \odot)$ is valid and triple $(\bullet, \circ, \circ)$, too. Finally, we determine three **types** incorporating all triples classified as reasonable:

1. Only car.
2. Much transit, much walking, no car.
3. Much transit, little walking, little car.

Here, the attributes *much* and *little* should be thought of to include the smaller relative durations. The types are complete to the effect that all relative duration triples classified as reasonable are included. This can again be deduced from Table 2. For practical purposes, relative durations need to be defined concretely. In the following, we introduce such definitions.

4.3 Thresholds

To practically use the types defined above, we propose to use **threshold** values for the relative durations. The following definitions reflect that *zero* signifies a transportation mode is not used, *little* depends on the mode and *much* means unlimited usage of a mode (durations in minutes):

- $zero(*) := 0$ min
- $little(walking) := 10$ min
- $little(car) := \begin{cases} 0 \text{ min,} & \text{if pure car duration} < 20 \text{ min} \\ \max(10 \text{ min}, 0.25 \cdot \text{pure car duration}), & \text{otherwise} \end{cases}$
- $much(*) := \infty$ min

Note that we defined the thresholds for a metropolitan setting. The definition for *much* is natural, since it represents everything which is greater than *little*. One can observe that the definition of *little(car)* is only relevant for paths belonging to type 3. Moreover, a path of this type is only interesting if it significantly differs from the path of type 1 (using the car for the whole way) in terms of car usage, otherwise one could just choose the path of type 1. Therefore, we chose 25% of its duration as upper bound but at least 10 minutes in order to avoid enforcing absurdly low car durations. For *little(walking)* we decided for a fixed threshold in order to allow nearby stations to be reached but avoiding journeys where walking significantly exceeds car usage. We consider a fixed threshold reasonable, as paths of type 3 have a duration of at most a few hours (in a metropolitan setting). If we chose *little(walking)* dependent on this maximal duration, it would be bounded from above by an absolute value anyway. However, we want to stress that the types can be used with other definitions of thresholds as well.

4.4 Filtering

We introduce a post-processing procedure to obtain small yet representative sets of reasonable paths from Pareto sets. Given the above defined types and (arbitrary) thresholds, we explain how to use them to remove unreasonable paths and how to obtain small yet representative paths in a second step. We call the whole concept **Types aNd Thresholds (TNT)**.

We say a path *belongs to a type* if none of the type's thresholds is exceeded. For instance, assuming a pure car duration of 15 minutes, the path of Figure 1 belongs to none of the

three types. To remove all unreasonable paths, the ones belonging to no type are removed. Note that for optimal results, walking duration must be considered as a Pareto criterion, too. That is, Pareto criteria are duration, transfer penalty, car duration and walking duration. After removing the unreasonable paths, we drop walking duration as a Pareto criterion. This removes undesired (minor) variation in the result set with respect to walking duration. Potentially, this can lead to the loss of interesting (reasonable) optimal paths. However, that is unlikely because it is unlikely that a path with higher walking duration dominates a path with lower walking duration in all other criteria.

To determine a small and representative subset from the remaining paths, we propose to transform all car durations according to their relative durations:

$$rd(\text{car duration}) := \begin{cases} 0, & \text{if car duration} = \text{zero(car)} \\ 1, & \text{if zero(car)} < \text{car duration} \leq \text{little(car)} \\ 2, & \text{if little(car)} < \text{car duration} < \text{much(car)} \end{cases}$$

Note that this coarse discretization is in sync with our coarse classification of travel times into three categories (zero, little, much) argued for in Section 4.2. The Pareto set of the transformed labels constitutes the result, now indeed a small yet representative subset of the reasonable Pareto-optimal solutions. Table 3 shows an excerpt of the results of a real query for Dallas.

■ **Table 3** Excerpt of the tuples of the optimal paths for an example query for Dallas. Pareto criteria are duration, transfer penalty, car duration and walking duration. Green tuples remained after filtering with TNT, gray ones did not. Before filtering, there were 66 Pareto optimal paths, after filtering only 7 reasonable paths remain.

duration	transfer penalty	walking duration	car duration	type
0:29:17	1	0:00:00	0:29:17	1
		. . .		
1:52:11	4	0:07:33	0:13:35	none
1:56:10	4	0:04:18	0:09:52	3
1:56:10	5	0:06:54	0:09:35	3
		. . .		
2:08:49	3	0:02:17	0:09:43	3
2:42:13	3	0:48:42	0:00:00	2
2:57:49	2	0:54:38	0:00:00	2
3:37:10	1	2:23:07	0:00:00	2
6:02:31	0	6:02:31	0:00:00	2

Influence of Thresholds on the Results. The choice of fixed thresholds obviously restricts the space of possible paths, but one does not want to miss significantly better paths which do not severely exceed the thresholds. Next, we explain how this can be achieved.

An advantage of our system is that lowering the thresholds can never lead to better paths with respect to duration and transfer penalty. To avoid missing significantly better paths which are not severely above the thresholds we propose to offer the user an outlook of how the paths improve with a higher threshold. One way to achieve this is to compute the difference in duration of the fastest paths for two significantly different thresholds. This enables the user to decide if she wants to run another query with modified threshold values.

4.5 Speed-up Techniques for Faster Query Answering

Each additional Pareto criterion enlarges the set of optimal paths significantly, resulting in infeasible query times for large datasets. To speed up query computation we introduce an optimality preserving extension, a relaxation of the model and a heuristic.

4.5.1 Extended Dominance by Early Results

To prune labels during query computation dominance by early results was introduced in [8]. All labels dominated by the target labels can be discarded, since they and all their extensions can not become Pareto-optimal at the target. For TNT, this can be extended to labels not belonging to any type. Extensions of such labels can not belong to any type and can therefore be discarded. Therefore, for the multi-criteria Dijkstra, dominance by early results can be extended to ignore labels for which the following holds:

- *walking duration > little(walking)* and *car duration > zero(car)* or
- *car duration > little(car)*

Recall that *little(car)* depends on pure car duration, which is computed in the first step of the shortest-path computation described in Section 3.4. Hence, it is available for the (time-consuming) multi-criteria Dijkstra. Note that this optimization is applicable independent from the threshold definition.

4.5.2 Rounding on Transfers

As we exemplified in Table 1, many Pareto optimal paths are similar. The table indicates that many solutions differ only by seconds. In our implementation, arc durations are stored with a resolution of one second. This is enough to avoid the accumulation of rounding errors (which for a resolution of, say, one minute, would tangibly impact result optimality). For road networks, we calculate durations depending on distances and speed, leading to an accuracy of seconds. The GTFS data [10], which we use to model the transit network, provides durations in seconds.

However, public transportation in practice rarely provides accuracy by seconds and the speed of humans in terms of walking and car usage varies, too. Inspired by discretization, we propose to relax the model and *to round up durations to full minutes immediately before transfers*. Compared to rounding at each node, this restricts error accumulation sufficiently. Moreover, it can be interpreted as a coarse transfer buffer. With respect to reality, we consider this an optimality preserving technique.

Arc relaxations will happen for less labels, and less comparisons have to be performed when inserting a label to the set of labels attached to a node. Therefore, we expect rounding during query time to notably speed-up computation time.

4.5.3 Using Implicit Walking Duration

As mentioned in Section 4.4, when filtering labels to their types, walking duration has to be a Pareto criterion to obtain optimal results. Nevertheless, when walking duration is not a Pareto criterion, we expect the difference to the optimal results to be minor. Therefore, we propose the heuristic of using implicit walking duration by keeping it in a hidden variable, which is not used as Pareto criterion. The priority queue order is chosen such that in case of tie-breaking the label with less walking duration is released earlier from the queue. For labels with equal Pareto criteria the one with less walking duration is kept. It is worth noting that

using implicit walking duration does not affect the quality of type 1 (only car) and type 2 (much transit, much walking, no car) paths. For type 1, this is clear since the optimal path is computed separately using only duration as criterion. As labels of type 2 (that is, with car duration $= 0$) cannot be dominated from labels with car duration > 0, we can ignore car duration when proving the optimality for labels of type 2 in the following lemma.

▶ **Lemma 1.** *Let L_A be the label set at a node $u \in V$ after termination of the Pareto-Dijkstra run considering the criteria D(uration), T(ransfer) P(enalty) and W(alking) D(uration). Also let L_B be the respective label set for a Pareto-Dijkstra run regarding only the criteria D and TP. We claim that $\forall l \in L_A \exists l' \in L_B : l' \leq l \mid_{(D,TP)}$.*

Proof. Let $l^* \in L_A$ be the label with $TP(l^*) = TP(l)$ and minimal duration. Obviously when neglecting walking duration and following the same path that lead to the creation of l^* at u, the label $l^* \mid_{(D,TP)}$ is a possible candidate for being in L_B. Hence it must exist a label $l' \in L_B$ with $TP(l') \leq TP(l^*)$ and $D(l') \leq D(l^*)$. Therefore it holds $l' \leq l^* \mid_{(D,TP)} \leq l \mid_{(D,TP)}$.   ◀

To see that paths of type 3 are not necessarily optimal, consider the following tuples which are incomparable with Pareto criteria duration, transfer penalty, car and walking duration: (40 min, 2, 10 min, 5 min) and (30 min, 2, 10 min, 6 min). Using implicit walking duration, only the latter would be optimal. However, assuming an extension by 5 minutes of walking, the latter tuple would belong to no type and hence be filtered out, whereas the former would not.

5   Experimental Results

In this section we evaluate the concept of using Types aNd Thresholds (TNT) and its speed-up techniques with respect to result quality and query time.

5.1   Setup

Our implementation of the graph model and the optimal path algorithm, as described in Section 3, is written in `C++` and compiled with `GCC 4.6.3` with the `-O3` flag. Experiments were performed on a machine with 96GB of RAM and two Intel Xenon E5649 CPUs with 8 cores, each having a frequency of 2.53 GHz (exactly one core was used at a time). The used OS is Ubuntu `12.04`, operating in 64-bit mode.

To instantiate the multi-modal networks we used publicly available OSM [14] and GTFS data [10]. Details on our data sets can be found under `http://ad.informatik.uni-freiburg.de/publications`. We used the data of the first available Monday. OSM data was chosen to cover the terrain corresponding to the GTFS data. The road network graphs are symmetric and reduced to their largest connected component. For walking, we assumed an average speed of 5 km/h. For the car network, average velocity was chosen depending on the road type, ranging from 5 to 110 km/h. We evaluated our algorithm on the networks of *Austin, Dallas, Toronto* and *New York City* (in the following abbreviated as just *New York*). Table 4 contains an overview of the most important properties of these networks.

For each dataset, experiments were performed using 1000 queries between two random locations, each at most 1 km away from at least one transit station. We chose this restriction to avoid a significant amount of queries in areas where transit is not available, in which case the only interesting solutions would be car-only and walking-only. Departure times were chosen uniformly at random from the time range between 6:00 a.m. and 10:00 p.m.

■ **Table 4** Overview of important properties of the evaluated networks. Recall that the time-consuming step of the path computation operates on the cores instead of the whole road networks.

Graph		Austin	Dallas	Toronto	New York
Complete	Nodes	0.7M	2.8M	0.9M	4.0M
	Arcs	2.9M	12.0M	3.7M	17.3M
Transit	Stations	2.7K	11.6K	10.9K	16.9K
	Nodes	14.3K	49.0K	80.4K	118.6K
	Arcs	21.2K	73.0K	119.5K	175.9K
	Lines	235	563	1120	1989
	Trips	6062	10849	40740	62824
Car	Nodes (core)	3.3K	20.8K	10.9K	28.1K
Walking	Nodes (core)	4.0K	20.2K	12.2K	32.5K

5.2 Results

We evaluate the concept of TNT with respect to query time and quality of the found sets of paths. Experiments were performed for the normal graph model (Section 3.1) and the model with rounding on transfers (Section 4.5.2). We refer to the latter as the *relaxed* model. For both models we compare the basic algorithm (Section 3.4) and the heuristic of using implicit walking duration (IWD, Section 4.5.3).

Table 5 shows average query times for all of our four datasets. It indicates that both,

■ **Table 5** Average query times for all datasets.

Model	Algo	Austin	Dallas	Toronto	New York
Normal	Basic	0.6s	3.7s	16.6s	108.0s
	IWD	0.1s	0.8s	0.6s	1.7s
Relaxed	Basic	0.4s	2.2s	4.0s	18.0s
	IWD	0.1s	0.8s	0.6s	1.4s

rounding on transfers and the IWD heuristic reduce query times. While the heuristic has a stronger effect, the lowest query times (roughly one second) are achieved by applying both. It is noteworthy that the speed-up increases significantly with the size of the network (roughly from factor 5 to factor 75). Query times are comparable to those presented in [4].

For the largest dataset (New York) we evaluate the basic algorithm and the IWD heuristic in more detail, for both models and with respect to both result quality and query time; see Table 6. As quality measures of the heuristic we use precision[1] and recall[2]. It can be seen

■ **Table 6** Query times and result quality for New York. For all measured variables we list average, 50-percentile, 90-percentile and 99-percentile values.

Model	Algo	Time [s]				Precision				Recall			
		avg	50	90	99	avg	50	90	99	avg	50	90	99
Normal	Basic	108.0	32.7	289.0	865.0	1	1	1	1	1	1	1	1
	IWD	1.7	1.0	3.3	10.0	0.99	1	1	1	0.96	1	1	1
Relaxed	Basic	18.0	8.8	40.0	140.0	1	1	1	1	1	1	1	1
	IWD	1.4	1.0	2.5	6.6	0.99	1	1	1	0.96	1	1	1

■ **Table 7** Percentage of queries which lead to the different combinations of paths of type 2 and type 3. For each query one path of type 1 was optimal.
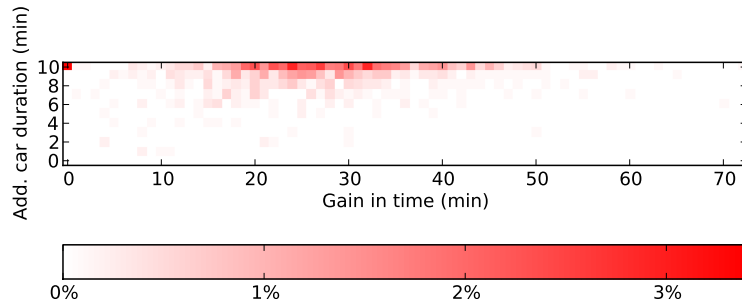
#type-3 paths	1	2	3	4	5	6	7
3	-	-	0.1%	1.0%	0.2%	-	-
2	-	0.9%	5.6%	7.0%	2.2%	0.4%	-
1	-	1.2%	**15.8%**	**14.4%**	2.7%	0.3%	0.1%
0	1.6%	**10.6%**	**20.9%**	**12.5%**	2.4%	0.1%	-

#type-2 paths

that (with the IWD heuristic in the relaxed model) for a few outliers the query time rises up to seven seconds, but the majority of queries can be answered in roughly one second. Precision and recall indicate that for both models the IWD heuristic leads to only a small fraction of non-optimal results.

To evaluate if the computed sets of paths are small and representative, Table 7 shows the distribution of paths with respect to our types for the basic algorithm on New York. Paths of type 3 that also belong to type 2 were only counted for type 3. For example, the table shows that around 15.8% of the queries lead to exactly one path of type 1, three paths of type 2 and one path of type 3. Note that for almost 50% of the queries there is no optimal path of type 3. One reason for this is that if pure car duration is relatively small (below 20 minutes, see section 4.3 and 4.4), no path solely belonging to type 3 can exist.

To see how paths of type 3 improve when increasing the *little(car)* threshold, we experimentally evaluated the maximal gain in time when extending the threshold by 10 minutes. For this, we considered queries which already for *little(car)* had labels of type 3 and compared the fastest such label (= with the smallest duration) with the fastest label when using the extended threshold. Figure 2 shows the results for New York. For 42% of the queries,

■ **Figure 2** Maximal possible gain in time for labels of type 3, when allowing additional car travel time of up to 10 minutes. The heat map shows the gain in time and respective additional car travel time (with respect to *little(car))* for the different queries.

increasing car travel time up to 10 minutes allows to reduce the total duration by 20-30 minutes. For practically every query increasing *little(car)* leads to faster paths. As explained in Section 4.4, this information could be communicated to the user (for the given query), with the option to relaunch the query with an accordingly modified threshold value.

---

[1]   Precision = |*relevant-paths* ∩ *found-paths*| / |*found-paths*|
[2]   Recall = |*relevant-paths* ∩ *found-paths*| / |*relevant-paths*|

6 Conclusions & Future Work

We studied multi-modal route planning involving (almost) unrestricted combinations of walking, car, and transit. The goal was to efficiently compute small yet representative sets of optimal paths. We illustrated that multiple criteria are necessary to obtain diverse sets of paths. To remove unreasonable paths and to extract a small representative subset of the remaining paths, we introduced a new approach of using Types aNd Thresholds (TNT). To reduce infeasible query times induced by multiple optimality criteria, we introduced an extension of dominance by early results, a relaxation of the model (rounding on transfers) and a heuristic. We experimentally evaluated TNT and the speed-up techniques. While the basic algorithm results in infeasible query times, relaxing the model and using the (almost optimal) heuristic reduces them to an average of roughly one second. Our experiments confirmed that our result sets are indeed small and representative, at least from the point of view of our model. Possible future work comprises examining other threshold definitions, extending the filtering step and considering fare zones. Lowering query times when using TNT is a further challenge. For the latter, one possibility could be to extend Transfer Pattern Routing [2] to our multi-modal scenario. Moreover, reliability and robustness (i.e., if connections are missed, how good are the alternatives) are important issues to consider.

References

1 Hannah Bast. Car or public transport - two worlds. In *Efficient Algorithms, LNCS 5760*, pages 355–367, 2009.

2 Hannah Bast, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev, and Fabien Viger. Fast routing in very large public transportation networks using transfer patterns. In *ESA, LNCS 6346*, pages 290–301, 2010.

3 Daniel Delling, Julian Dibbelt, Thomas Pajor, Dorothea Wagner, and Renato F. Werneck. Computing and evaluating multimodal journeys. Technical report, Karlsruhe Institute of Technology, 2012.

4 Daniel Delling, Julian Dibbelt, Thomas Pajor, Dorothea Wagner, and Renato F. Werneck. Computing multimodal journeys in practice. In *SEA, LNCS 7933*, pages 260–271, 2013.

5 Daniel Delling, Thomas Pajor, and Dorothea Wagner. Accelerating multi-modal route planning by access-nodes. In *ESA, LNCS 5757*, pages 587–598, 2009.

6 Daniel Delling, Thomas Pajor, and Renato Fonseca F. Werneck. Round-based public transit routing. In *ALENEX*, pages 130–140, 2012.

7 Julian Dibbelt, Thomas Pajor, and Dorothea Wagner. User-constrained multi-modal route planning. In *ALENEX*, pages 118–129, 2012.

8 Yann Disser, Matthias Müller-Hannemann, and Mathias Schnee. Multi-criteria shortest paths in time-dependent train networks. In *WEA, LNCS 5038*, pages 347–361, 2008.

9 Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *WEA, LNCS 5038*, pages 319–333, 2008.

10 General Transit Feed Specification (GTFS). `https://developers.google.com/transit/gtfs/`, October 2012.

11 P. Hansen. Bricriteria path problems. In *Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making - Theory and Application*, pages 109–127. Springer, 1979.

12   Paola Modesti and Anna Sciomachen. A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111(3):495 – 508, 1998.

13   Matthias Müller-Hannemann and Mathias Schnee. Finding all attractive train connections by multi-criteria pareto search. In *ATMOS, LNCS 4359*, pages 246–263, 2004.

14   Open Street Map (OSM). `http://www.openstreetmap.org`, October 2012.

15   Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos D. Zaroliagis. Efficient models for timetable information in public transportation systems. *ACM Journal of Experimental Algorithmics*, 12, 2007.

16   Haicong Yu and Feng Lu. Advanced multi-modal routing approach for pedestrians. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 2349 –2352, April 2012.