Multi-Modal Route Planning

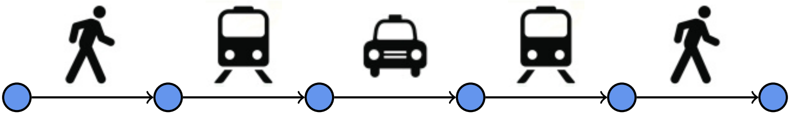
Mirko Brodesser

15. April 2013

Outline

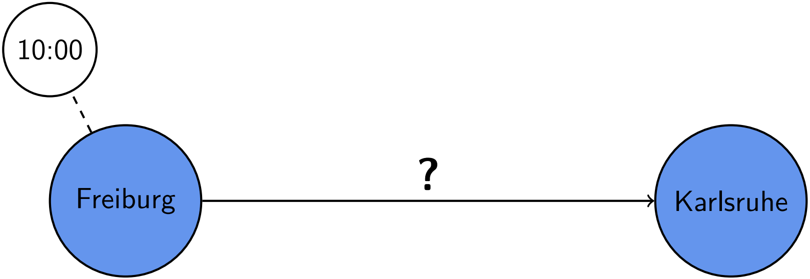
Introduction

- Current route planning is mostly **uni**-modal (or very restricted)
- We focus on **multi**-modal route planning, which allows (almost) all variations of , , 
- Especially, we allow such connections:



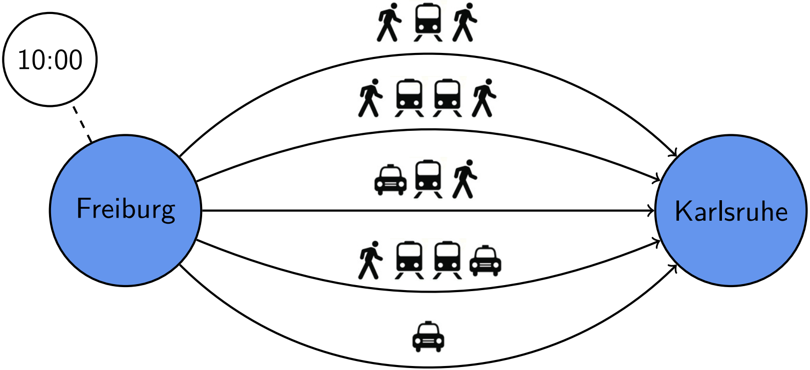
Introduction

We want to answer the question:
For a given departure time, how can one get from A to B, example:



Introduction

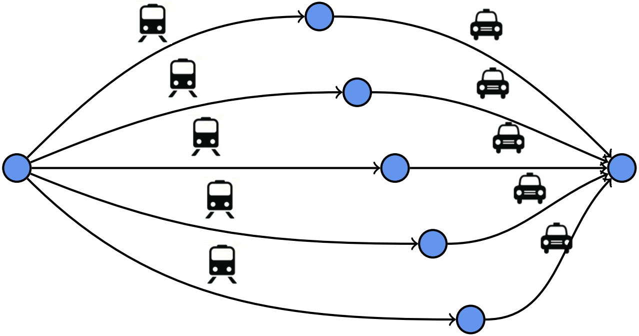
We want to answer the question:
For a given departure time, how can one get from A to B, example:



Goal: (Quick) computation of **concise** & **diverse** set of paths

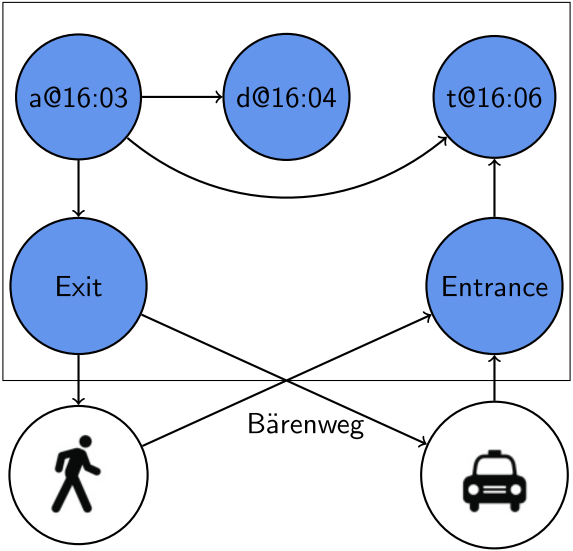
Introduction

- Example for set of paths which is **not concise**:



Model

Our **combined model** =
transit network + road networks + connections, example:



Algorithm

Algorithms to compute optimal paths:

- Multi-Criteria Dijkstra (source-to-all, for all networks, slow)
- Contraction Hierarchies (source-to-target, for road networks, very fast)
- Transfer Patterns (source-to-target, for transit networks, very fast)

Computing optimal paths on our combined model:

1. Taking the car/walking exclusively: Contraction Hierarchies
2. Remaining paths: Multi-Criteria Dijkstra
3. Result is union of 1. and 2.

Multi-Criteria Shortest Paths

Taking into account multiple optimality-criteria: Pareto Sets

Pareto Set: Set of tuples, each criterion corresponds to one component. If $t_1 \leq t_2$ with component-wise comparison, t_1 *dominates* t_2. Pareto Set consists of non-dominating tuples.

Multi-Criteria Shortest Paths

Taking into account multiple optimality-criteria: Pareto Sets

Pareto Set: Set of tuples, each criterion corresponds to one component. If $t_1 \leq t_2$ with component-wise comparison, t_1 *dominates* t_2. Pareto Set consists of non-dominating tuples.

Example with criteria (duration, transfer penalty):

$$\{(6, 1), (10, 1), (26, 0)\}$$

Breisgau S-Bahn

Bus 11

Walking

Multi-Criteria Shortest Paths / Filtering Methods

Recall goal: Concise & diverse sets of paths.

Which criteria to use?

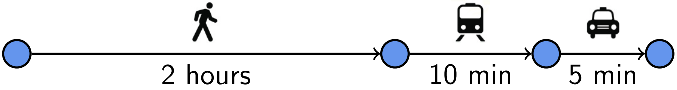
1. duration and transfer penalty? Two paths
2. duration, transfer penalty and car duration? Dozens of paths

In case of 2., set of paths is diverse: Post-process to concise subset

Our first approach: **Discretise** car duration

1. For example, in steps of 10 minutes
2. From many similar paths, only one is kept
3. But: Reveals that some Pareto optimal paths are undesirable

Example for 3.:



Filtering Methods: Types and Thresholds

Our second approach: Determine **types** of paths, using relative durations (much, little, zero):

1. Use the car exclusively.
2. Much transit, much walking, no car.
3. Much transit, little walking, little car.

Use **thresholds** for *much* & *little* (values in minutes):

- little(walking) $:= 10$
- little(car) $:= 0$ if pure car duration < 20, max(10, 25% pure car duration) otherwise
- much(*) $:= \infty$

Filter: Firstly by thresholds, secondly by using relative durations.

We call this **Types aNd Thresholds (TNT)**

Filtering Methods: Types and Thresholds

Example: Remaining paths after filtering with TNT:

duration	transfer penalty	car duration	path summary
0:23:00	1	0:23:00	
1:12:00	3	0:10:00	
1:47:00	2	0:10:00	
2:05:00	2	0	
2:35:00	1	0	
4:46:00	0	0	

Speed-up Techniques

Query times are infeasible, speed-up techniques required:

1. Flattening the transit graph: Keep one representative per line, assume it departs always (heuristic)
2. For discretisation: Perform it during query computation (heuristic)
3. For TNT: Discard paths not belonging to any type (optimality-preserving)
4. For TNT: Use implicit walking duration (heuristic)

Experimental Results

	Austin	Dallas	Toronto	New York
#Stations	3K	11K	11K	16K

Figure: Summary of evaluated datasets.

- Flattening the transit graph: Does not significantly reduce query times (factor 2-3)
- Discretisation: Query times in order of minutes, heuristic reduces it to tens of seconds. Number of filtered paths around 8-10. Recall of heuristic around 90%

Experimental Results: TNT

Data	Algorithm	Duration (seconds) avg/50/90/99	#Paths avg/50/90/99
Austin	Basic-p	2.7/0.8/7.6/14.9	4/4/6/8
	Heuristic-p	0.5/0.3/1.2/2.4	4/4/6/8
New York	Basic-p	308.0/260.0/628.0/1450.0	5/5/8/9
	Heuristic-p	54.1/25.8/81.0/298.0	5/5/7/9

Figure: Experimental results for TNT.

- Results for Dallas & Toronto between the ones of Austin & New York
- Missed optimal paths often not found approximately
- Heuristic close to optimal (For roughly 90% of the queries recall is 100%, in the worst case 70%)

Experimental Results: TNT

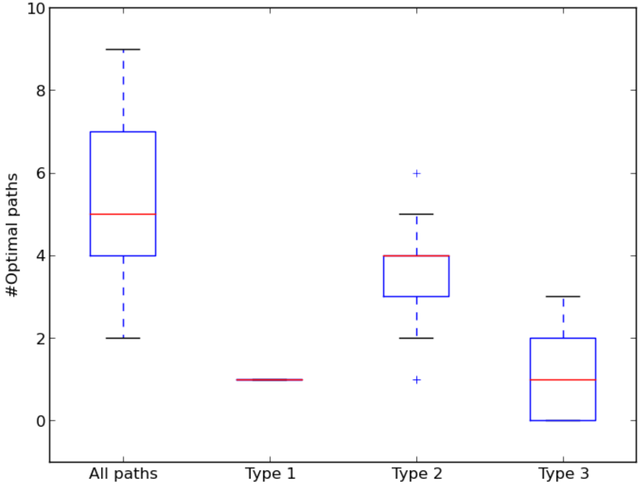


Figure: #Optimal paths and their distribution for New York.

Summary

- Goal was to (quickly) compute diverse & concise sets of paths
- Pareto Sets fulfil diversity, but optimal paths become to numerous
- We explored filtering methods: Discretisation and TNT
- Discretisation leads to more concise sets, but undesired paths remained
- TNT leads to concise sets
- Computation durations for discretisation and TNT without heuristics impracticable
- For TNT with heuristics in order of seconds, but for larger datasets still too high for practical use

Future Work

- Quality improvements still possible
- Running times need to be reduced to allow practical usage, for example with smarter pruning of labels
- Explore alternative graph models
- Take into account turn restrictions and traffic lights for more realistic modelling
- Support more criteria
- Investigate reliability and robustness of paths (how good are the alternatives if a transfer is missed)