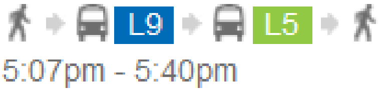
Route Planning

- Find optimal paths
  - Query: A@8:00 B

- Modes of transportation
  - Car only uni-modal
  - Walk + transit
   multi-modal



- Which criteria take into account?
  - Total travel time fastest
  - Price cheapest
  - Number of transfers less transfers

Brühl
Elsässer Straße
Seepark
Flückigersee
B
nhausen
Stühlinger
Berliner Allee
Freiburg
garten
3
Altstadt
31a
A
Haslach
Basler Straße
3

Source: https://maps.google.com/

Route planner multi-modal and multi-criteria

- Modes
  - Walk + transit + car
- Criteria
  - Total travel time, transfer penalty, car duration

Approach

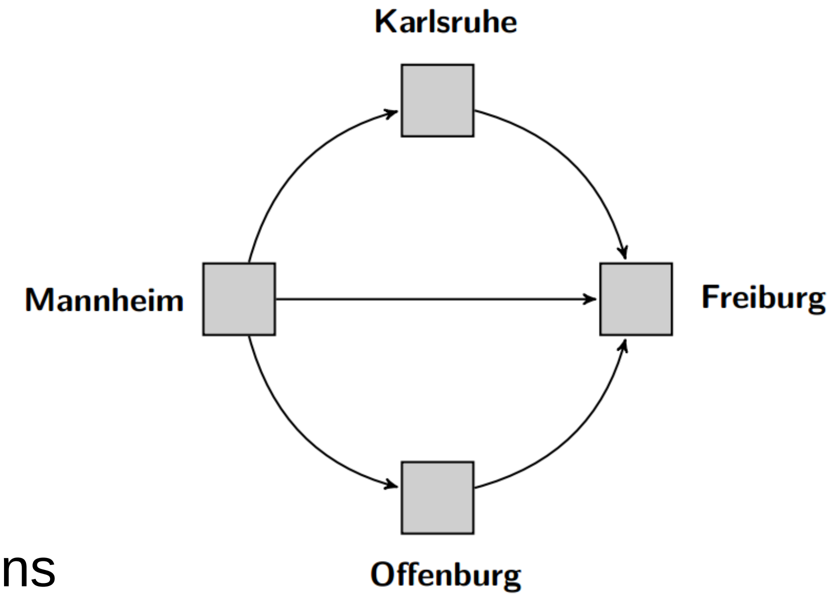
- Transfer Pattern Algorithm [1]
  - State-of-the-art routing algorithm for transit networks
  - Much **faster** than Dijkstra
- Types and Thresholds filter (TNT) [2]
  - Reduces inadequate results
  - Used in previous work with Dijkstra

Transfer pattern (TP)

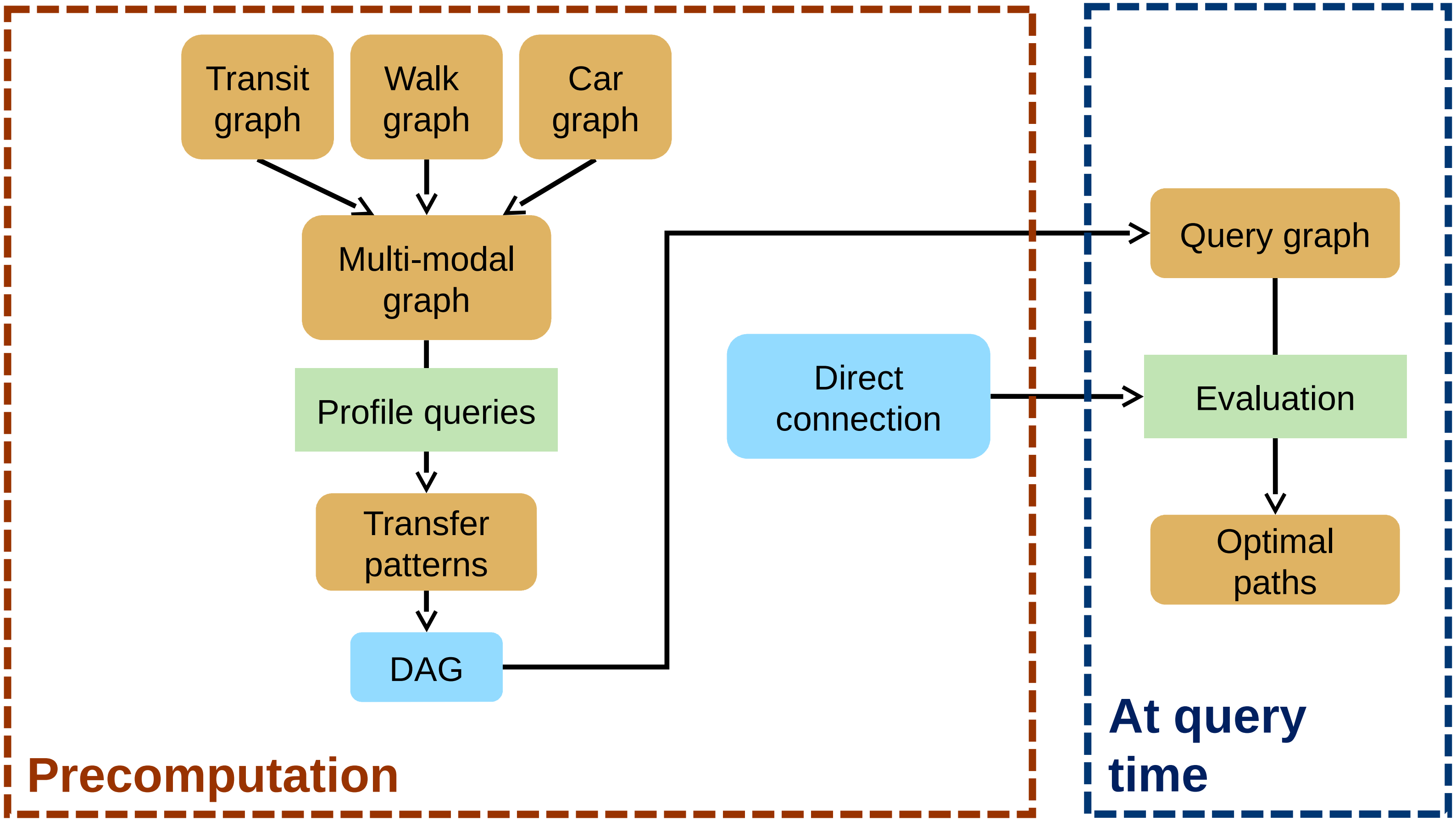
- Sequence of stations on a path where a transfer happens
- Example: Mannheim – Freiburg
  - 3 different transfer patterns
    MA-FR, MA-KA-FR , MA-OG-FR
- Few number of TPs for one journey

Basic Idea

- Precompute all TPs for pair of stations at all times and store them
- At query time (MA@8:00 □ FR)
  - Look into schedules of precomputed data
- Very fast responses

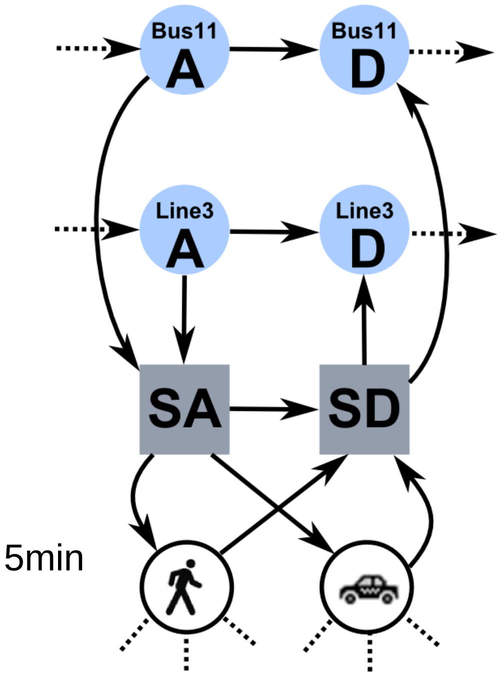


Routing with transfer patterns



Multi-modal graph

- Transit graph
  - Data (stations, lines, schedules)
  - For each station
  - Station arrival node (SA)
  - Station departure node (SD)
  - For each line serving a station
  - Line arrival node (LA)
  - Line departure node (LD)
- Road graphs (walk / car)
  - Node intersection of two roads
  - Arc road
  - Arc cost travel time



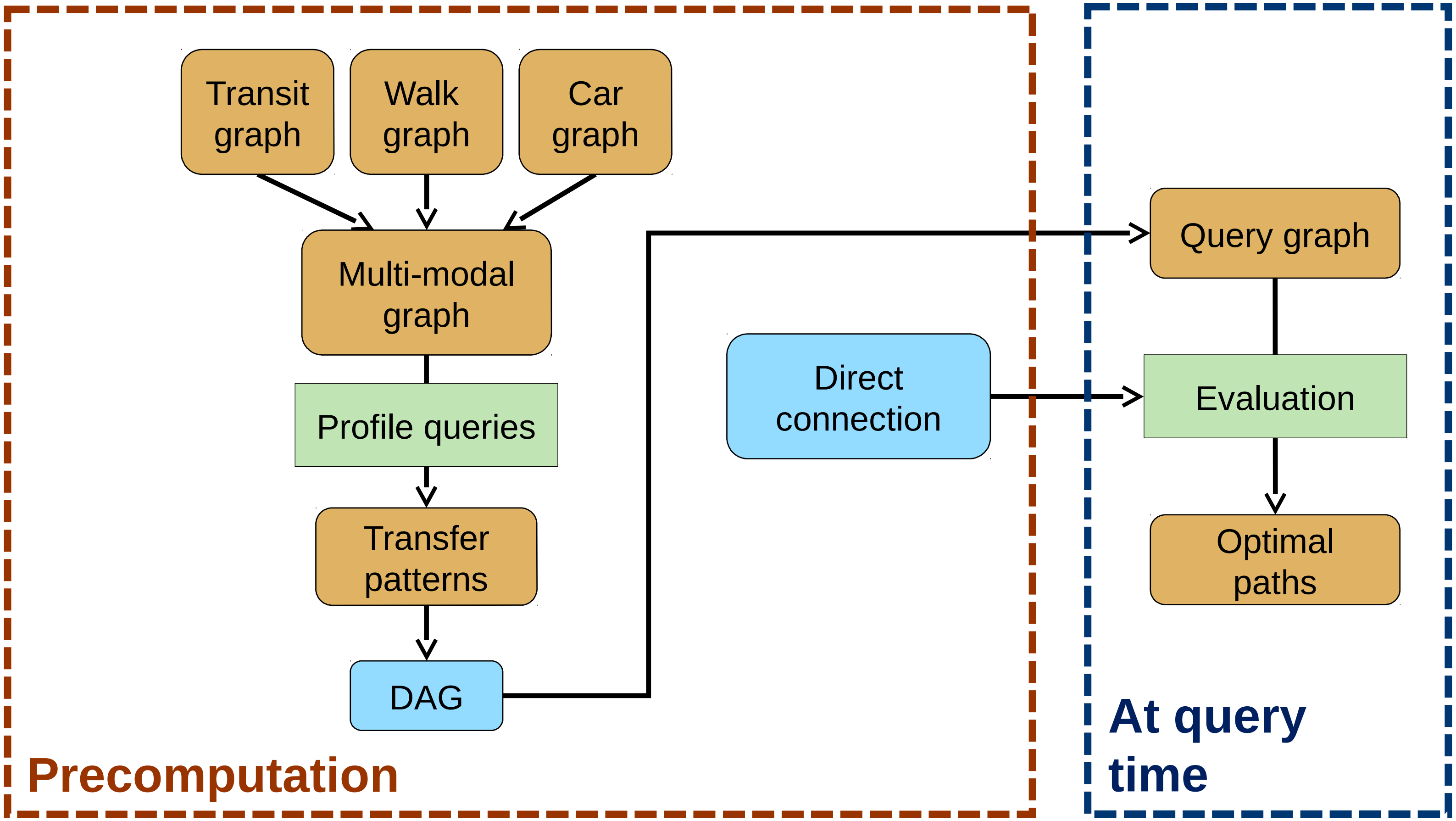
Profile queries

- Multi-label Dijkstra
  - For each station
  - All departure times
- Pareto set of labels
  - Total time, transfer penalty, car duration
  - Example: $\left.\begin{array}{l}(30\text{min}, 1)\\(40\text{min}, 0)\end{array}\right\}$ incomparable

    (30min, 1) better than
    (40min, 2)

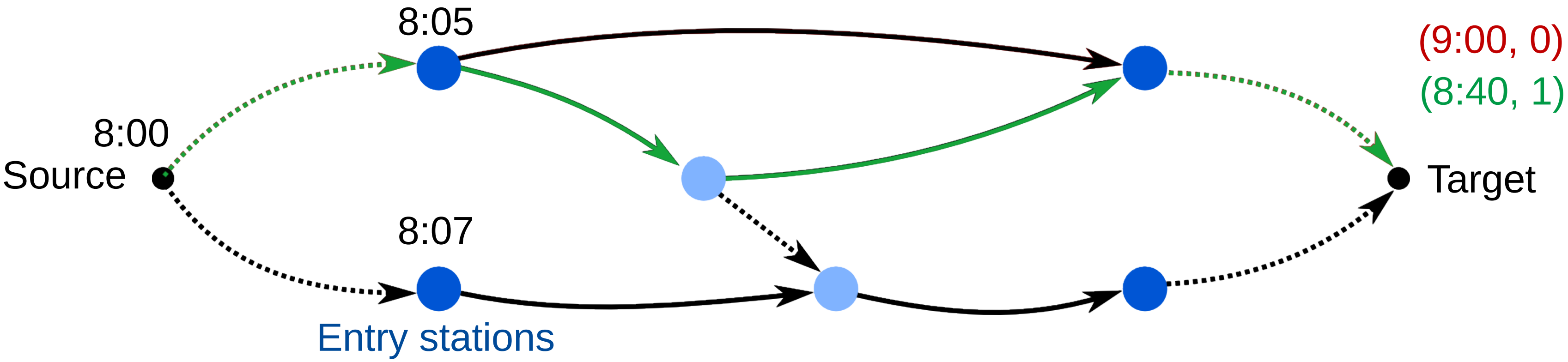
- Extract transfer patterns from optimal paths
- Store in Directed Acyclic Graph (DAG)

Routing with transfer patterns



Query graph

- Source@8:00 □ Target
- Construction
  - Entry stations 400 m around source and target
  - Precomputed transfer patterns
- Evaluation
  - Dijkstra on query graph
  - Arc cost □ direct connection queries
    □ road distances

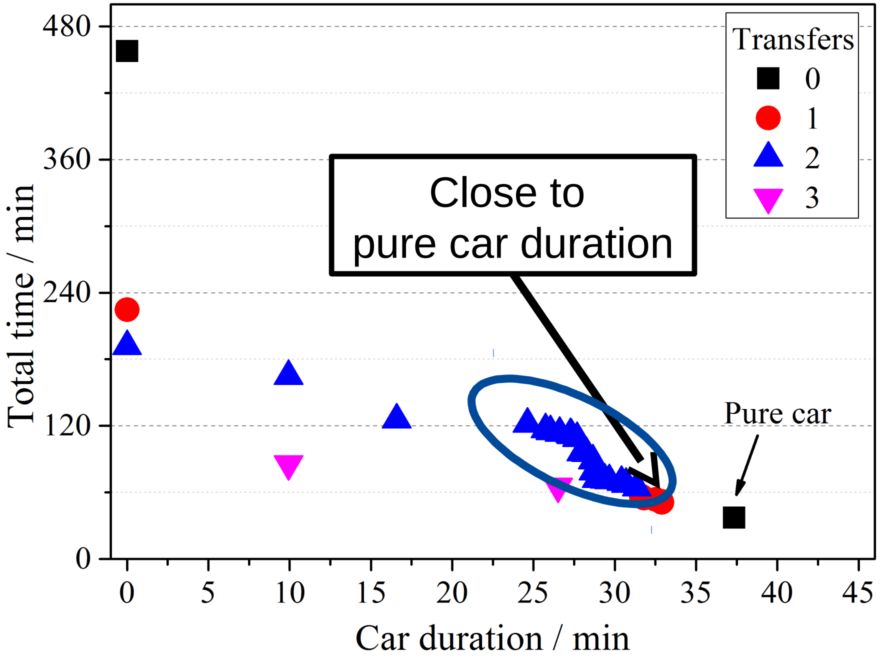


Query test

- Random query in Freiburg
- 30 optimal paths found

Issue

- Similar results
- Unreasonable paths



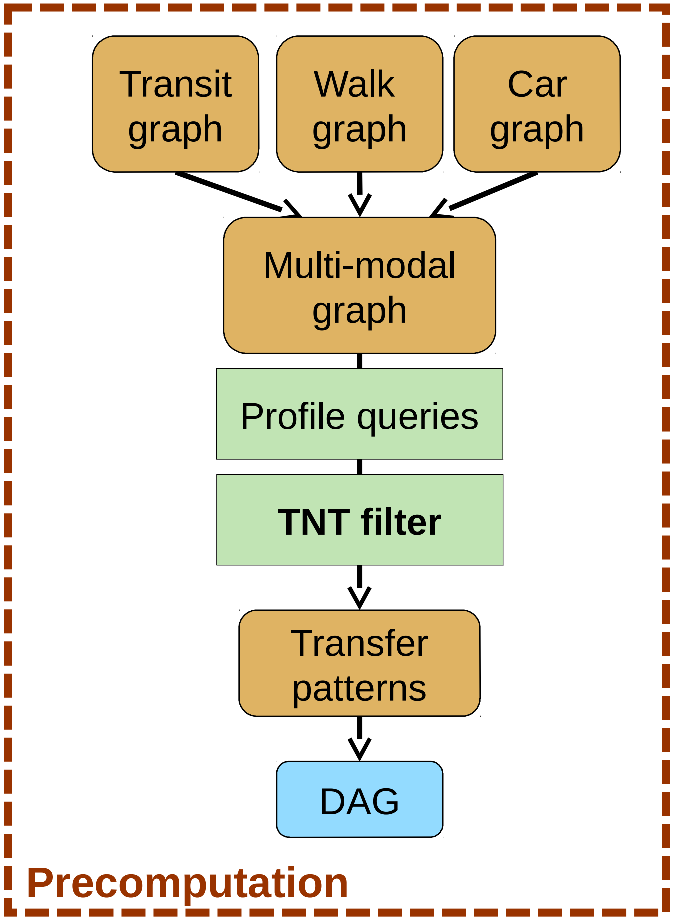
Close to pure car duration

Similar paths

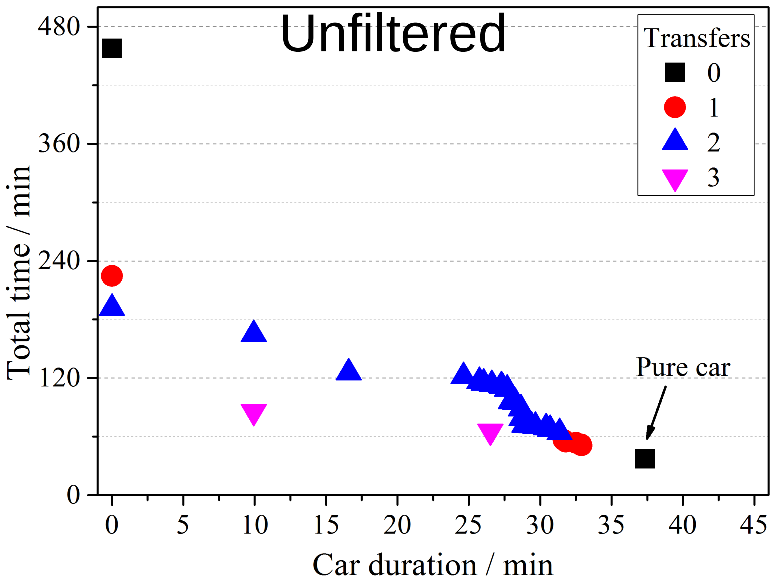
- Discretize car duration
  - Blocks 10 min

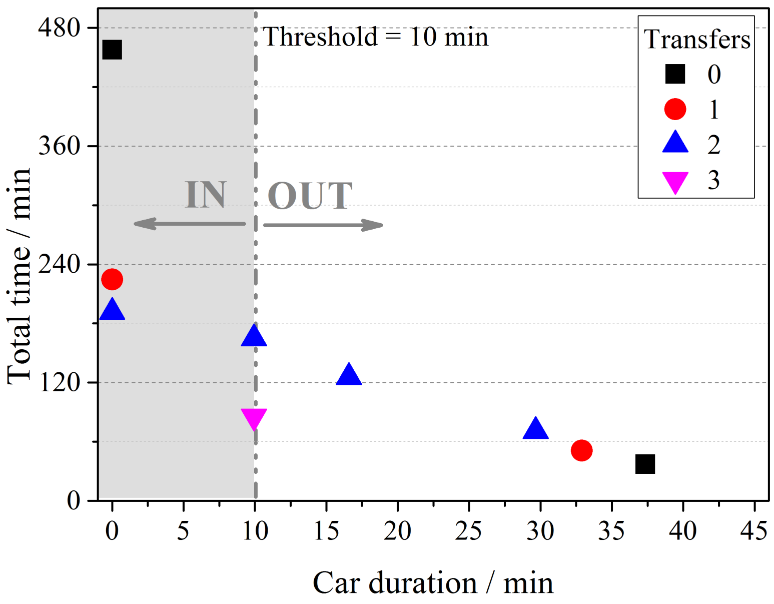
Unreasonable paths

- Types
- Car only
- Much transit, much walking, no car
- Much transit, little walking, little car
- Thresholds
- Little(walking) = 10 min
- Little(car) = 0 if pure car duration < 20min, otherwise max(10 min, 25% pure car duration)
- Much() = no limit



Query test with TNT filter





Datasets and multi-modal graph size

- Vitoria small bus network
- Freiburg medium network including surroundings
- Austin metropolitan area

	Vitoria	**Freiburg**	**Austin**
Stations	333	1,381	2,709
Lines	**40**	**569**	228
Trips	**2,733**	**2,328**	4,852
Nodes	2.8K	20.5K	27.9K
Arcs	11.4K	53.8K	96.9K

Precomputation

- Labels generated by random profile queries

	Vitoria	Freiburg	Austin
Transit	155K	101K	652K
Transit + walk	476K	352K	2,013K
Transit + walk + car	4,526K	7,695K	128,593K

- Transit transit + walk 3x
- With car greatly increases!
  - Car available everywhere and fast
  - A lot of combinations using car are optimal

Precomputation

- Average profile query times

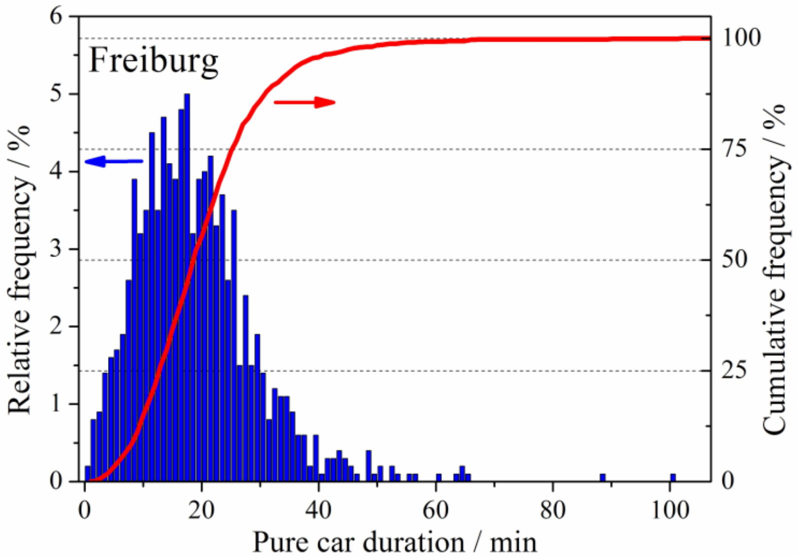
	Vitoria	Freiburg	Austin
Profile query time (min)	1.97	0.57	2,634.55

- Freiburg profile query time lower than Vitoria
  - Almost same number of trips, but Vitoria less lines
     high frequencies
- Austin very high profile query time
   not considered for further experiments

Quality evaluation for Freiburg network

- Precision
  - Fraction of retrieved paths that are relevant
  - ~ 99% quality preserved
- Recall
  - Fraction of relevant paths that are retrieved
  - Decreased due to TNT
  - Median pure car duration 17.6 min lower than limit (20 min)

	Precision	Recall
TP vs. Dijkstra	99.1 %	94.0 %
TP + TNT vs. Dijkstra	98.9 %	40.3 %



Number of labels and transfer patterns

- Comparison: Before and after TNT filter

Vitoria	Before	After
Labels	430	104
TP	15	3

Freiburg	Before	After
Labels	421	32
TP	145	13

- Vitoria: Bus lines with high frequency

   many labels compressed in one TP

- TNT filter reduces number of labels and TP
- Dijkstra
  - Multi-modal graph 20,531 nodes
  - Query graph w/o filter 90 nodes
  - Query graph w/ filter **30** nodes

Query times

- Average query graph (QG) construction and evaluation time

	Vitoria	Freiburg
Entry stations	5	3
Build QG (ms)	0.16	0.40
Evaluation QG (ms)	3	5
Build path (ms)	8	10
Total time (ms)	**11**	**15**

- Optimal paths computed in **milliseconds**!

- Multi-modal and multi-criteria route planning
  - Transfer Pattern Algorithm
  - TNT filter
  - Experiments with three different networks

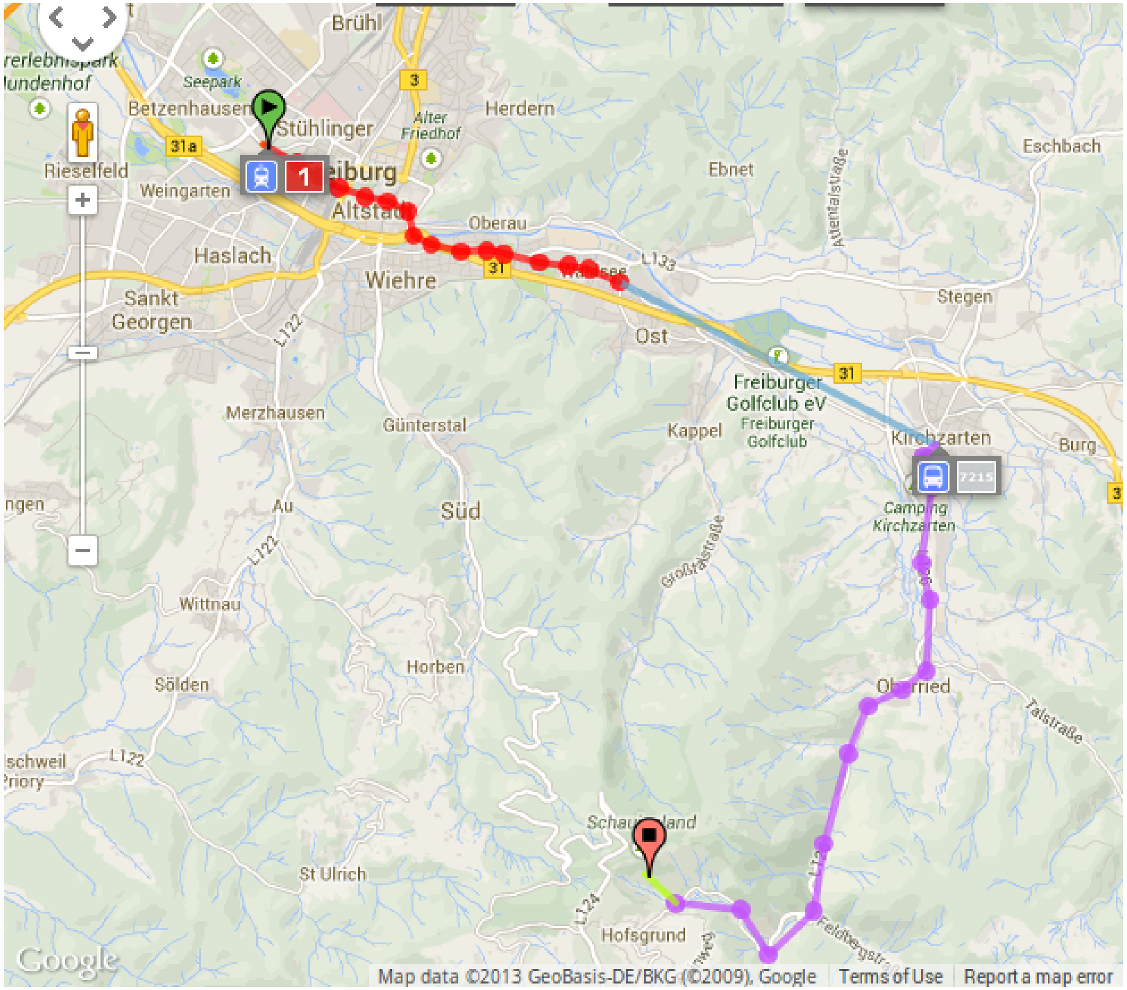
- Results
  - TNT filter eliminate similar and undesirable results
  - Car mode greatly increases number of labels
  - Network structure influences the profile query time
  - High frequency bus lines in Vitoria
  - TNT filter reduces labels and TPs
     smaller query graph
  - Fast query responses (milliseconds!)

Thank you for your attention!

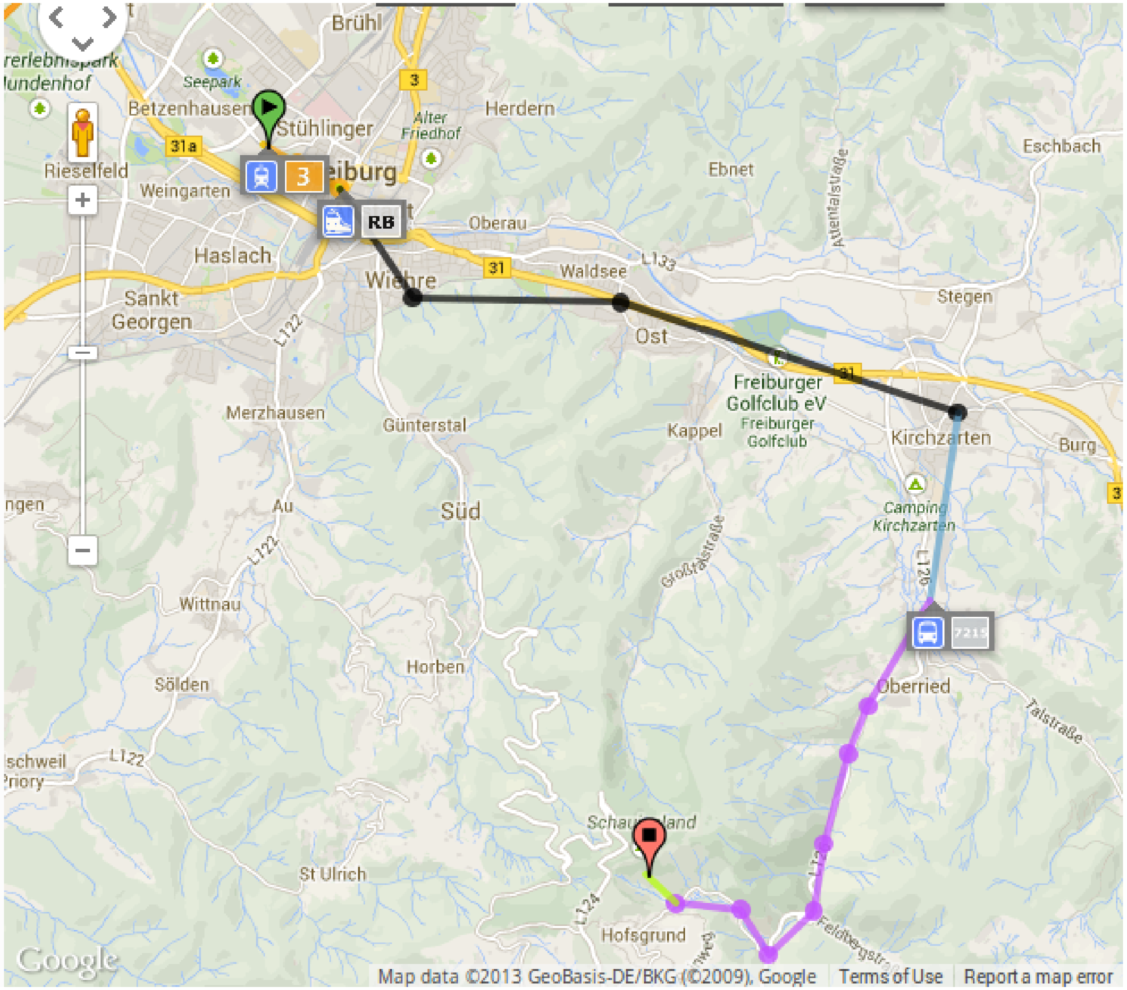
Questions?

- [1] Hannah Bast, et al. **Fast Routing in Very Large Public Transportation Networks using Transfer Patterns**. In Mark de Berg and Ulrich Meyer, editors, *ESA (1), volume* 6346 of *Lecture Notes in Computer Science, pages 290–301.* Springer, 2010.
- [2] Hannah Bast, et al. **Result Diversity for Multi-Modal Route Planning.** ATMOS-13 pages 123-135. 2013.

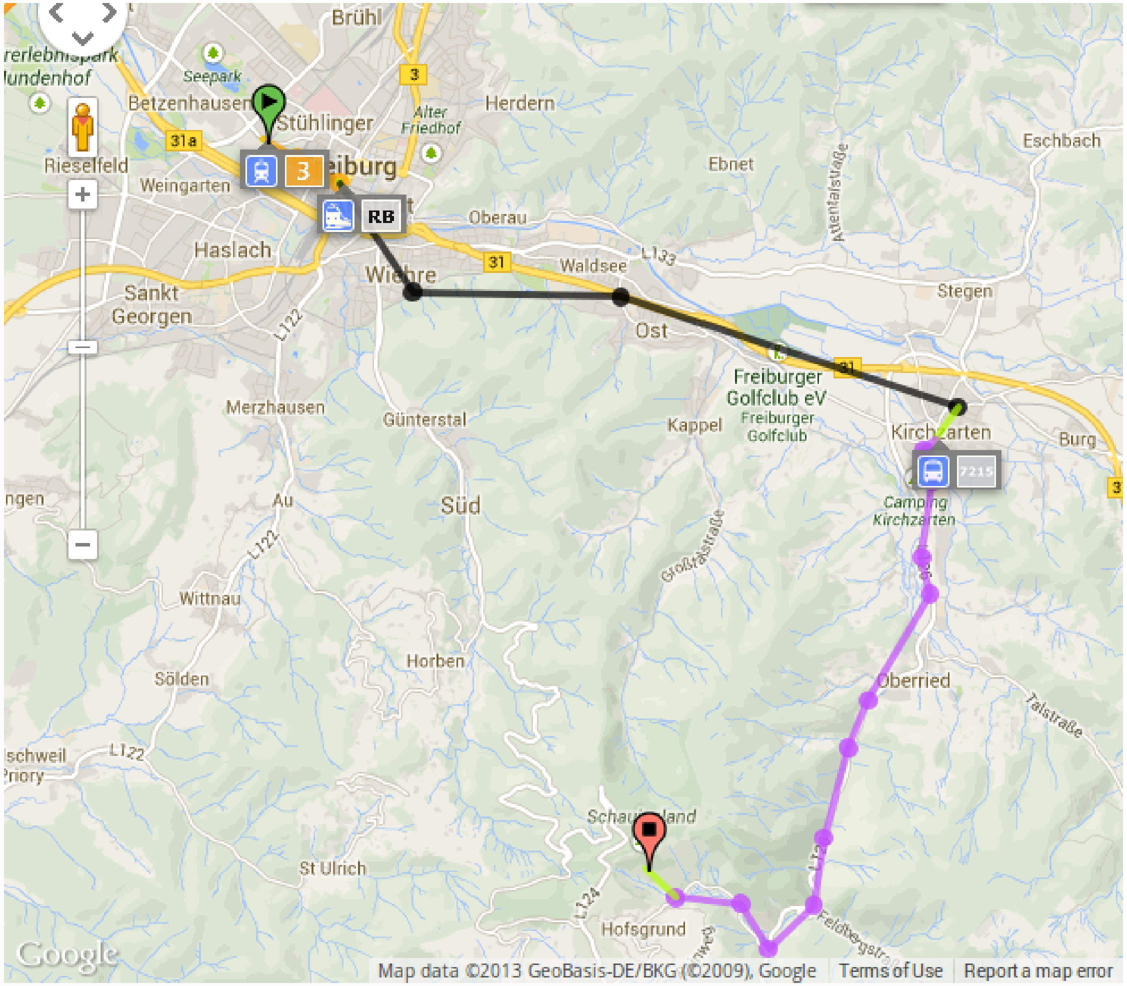
Route option 1



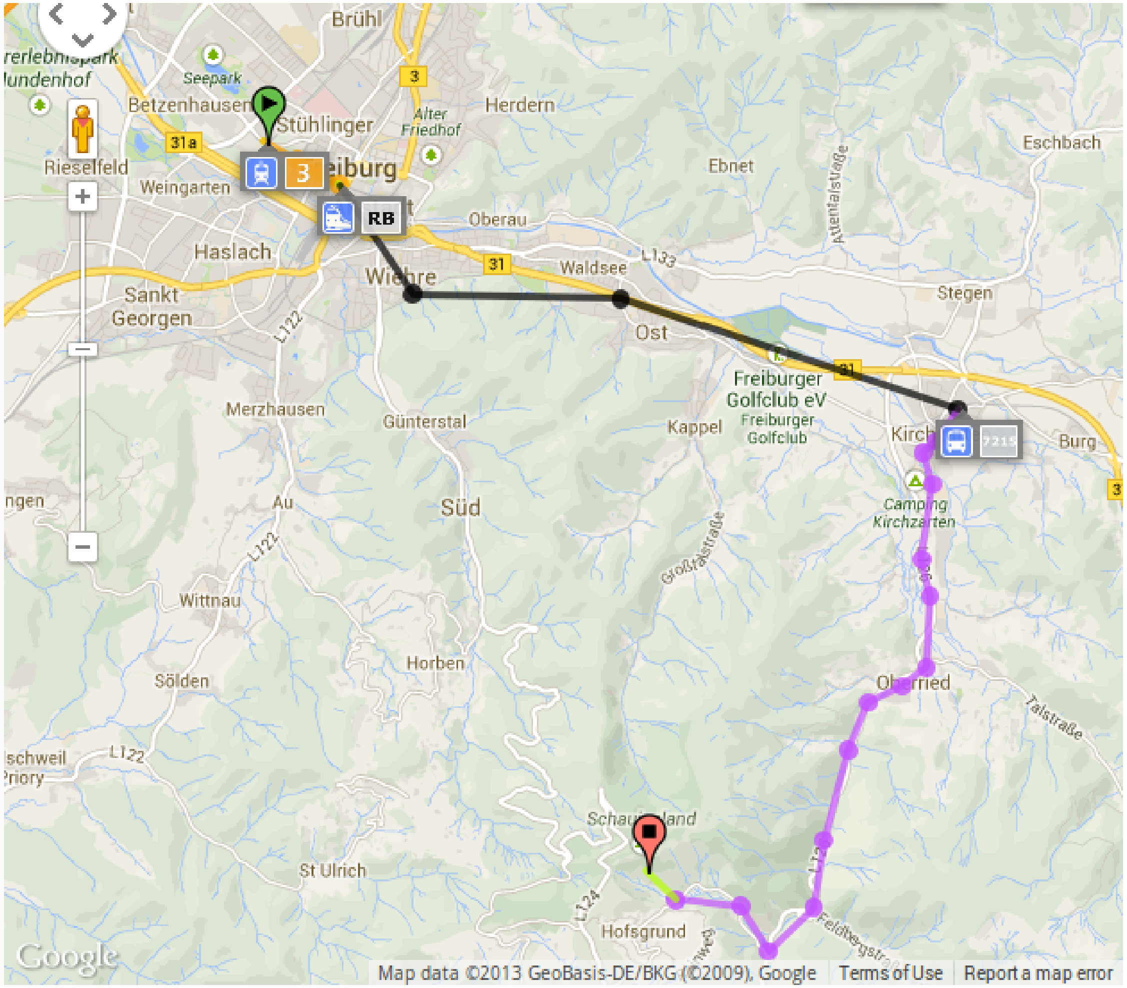
Route option 2



Route option 3



Route option 4



Ideas

- Take into account updates traffic
- Minimize profile query time
  - Limit the walking and car in the structure of the graph
  - Run profile queries for each departure time independently then use results for next run at the next departure time
  - Sort pareto sets to reduce comparisons
  - Important stations heuristic

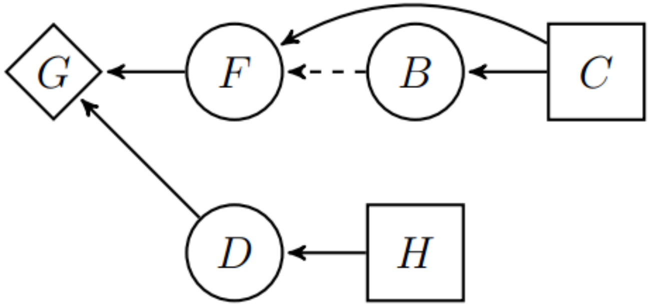
Direct connection

- Structure
  - Incident list for each station
  - Trip times for each line
- Query
  - Example:
  - HBF@8:00 Technische Fakultät
  - Intersect lists of two stations
  - Find earliest departure after 8:00

HBF	Techn. Fak.
(Bus11, 2)	(L13, 2)
(L3, 9)	(L5, 9)
(L6, 1)	**(Bus11, 4)**
...	**...**

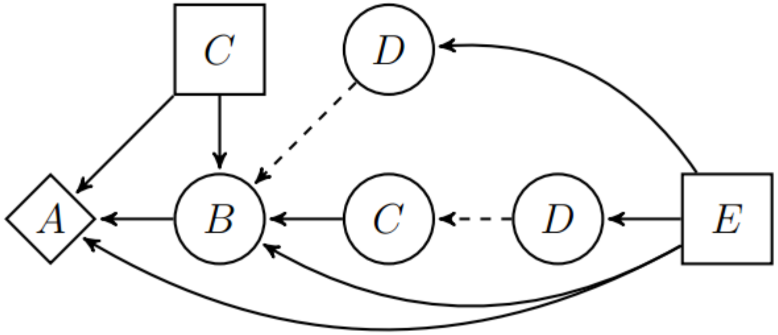
		Pos 2			**Pos 4**		
Bus11		HBF			Techn. Fak.		
Trip 1	…	8:05	**8:06**	…	**8:15**	8:16	…
Trip 2	…	8:45	8:46	…	8:55	8:56	…
...		...		...	**...**		...

Location-to-location query graph

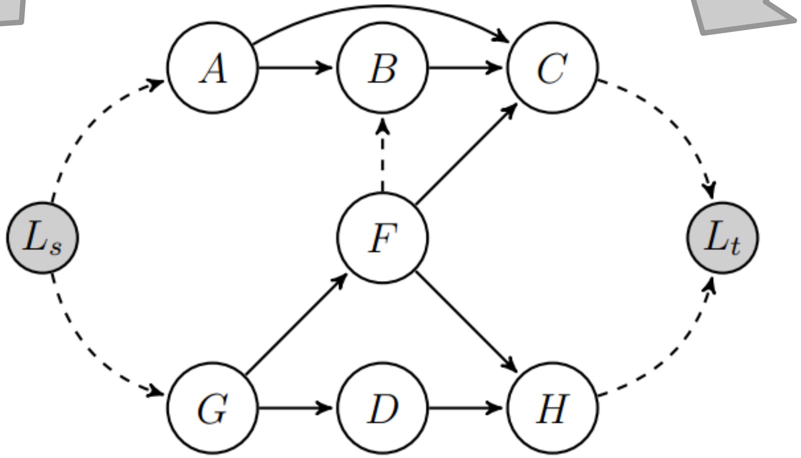


DAG
station G

GFC,
GFBC,
GDH

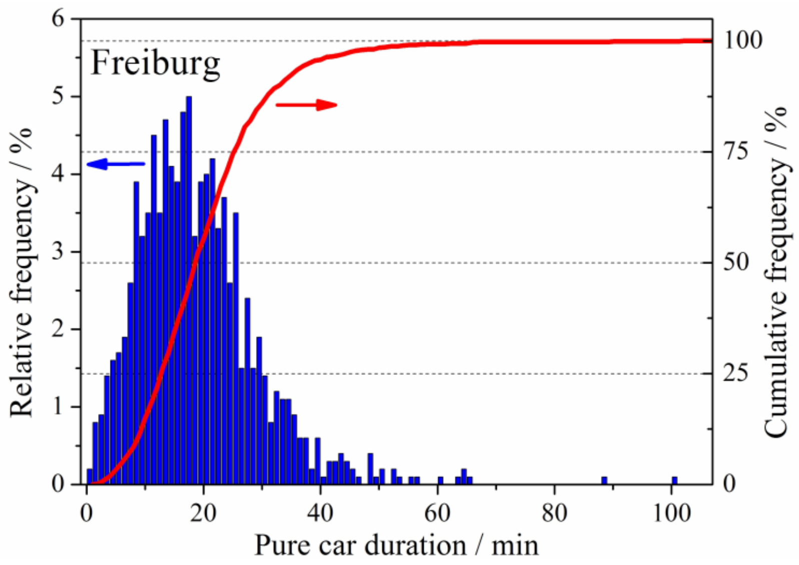


DAG
station A

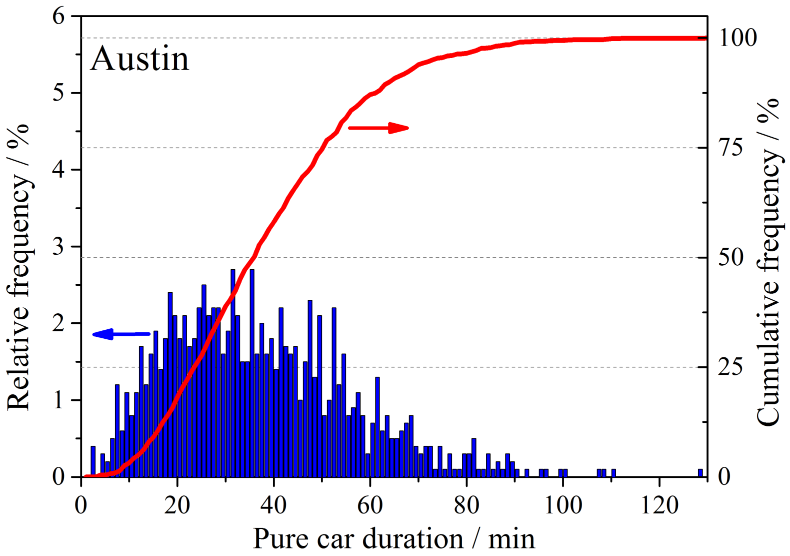


- Criteria
  - (duration, penalty, car duration)
- Pareto sets
  - Less or equal:
  - $(x, y, z) \leq (x', y', z')$ iff $(x \leq x') \land (y \leq y') \land (z \leq z')$
  - Less than:
  - $(x, y, z) < (x', y', z')$ iff $(x < x') \land (y \leq y') \land (z \leq z')$
  - $(x, y, z) < (x', y', z')$ iff $(x \leq x') \land (y < y') \land (z \leq z')$
  - $(x, y, z) < (x', y', z')$ iff $(x \leq x') \land (y \leq y') \land (z < z')$

Comparison: Pure car duration

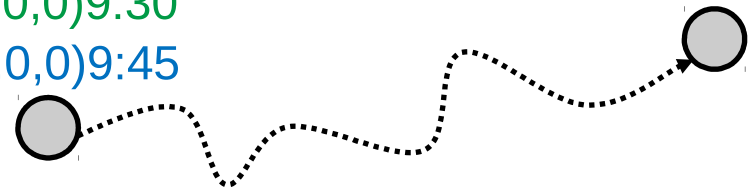


Median: 17.6 min



Median: 37.3 min

Pruning rule example



(45,2,0)10:00

(70,2,0)10:40

(45,2,0)10:30

□ Label not pruned

□ Label pruned

(0,0,0)9:15

(0,0,0)9:30

(0,0,0)9:45

Average number of labels and profile query time

	Vitoria	Freiburg	Austin
Profile query time (min)	1.97	0.57	2,634.55
Number of labels	1.94M	1.44M	128.59M