UnLimited TRAnsfers for Multi-Modal Route Planning: An Efficient Solution

ESA · September 11, 2019

Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP

Multi-Modal Route Planning

Goals:

- Journey planning for public transit
- Find **optimal** journeys
- Considered modes of transportation:
  - All timetable-based modes
    (trains, trams, buses, ...)

Multi-Modal Route Planning

Karlsruhe Institute of Technology

Goals:

- Journey planning for public transit
- Find **optimal** journeys
- Considered modes of transportation:
  - All timetable-based modes
    (trains, trams, buses, ...)

But also:

- Allow secondary transfer mode
- Non-schedule-based
  (walking, bike, e-scooter, ...)

Problem Statement

Karlsruhe Institute of Technology

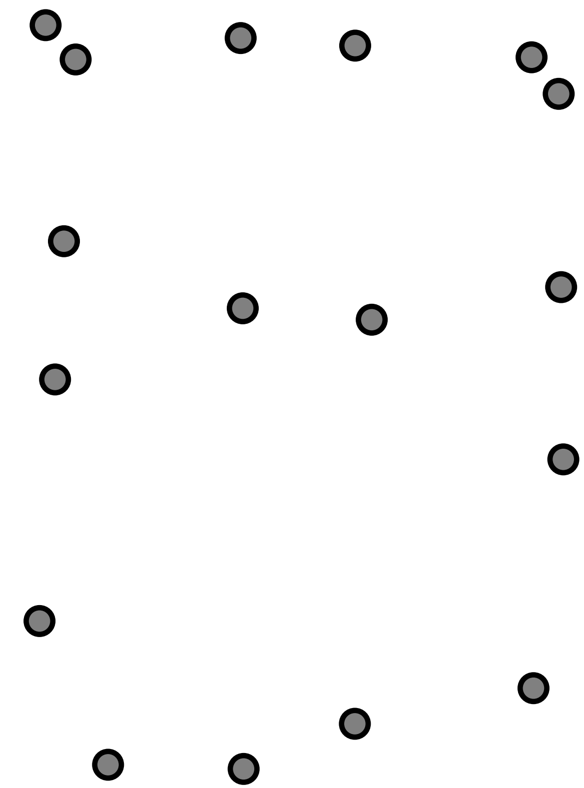
Given:

- Public transit network (timetable)

Problem Statement

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)

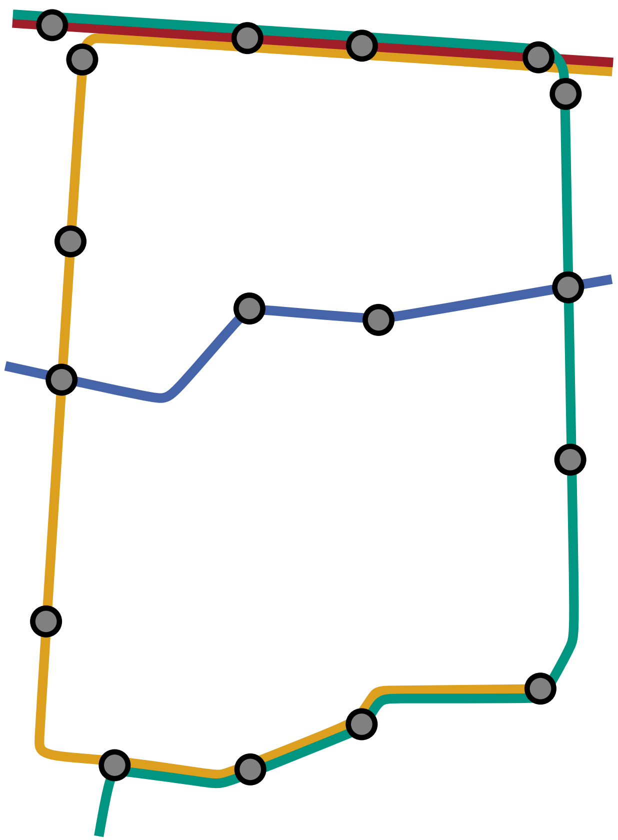


Problem Statement

Karlsruhe Institute of Technology

Given:

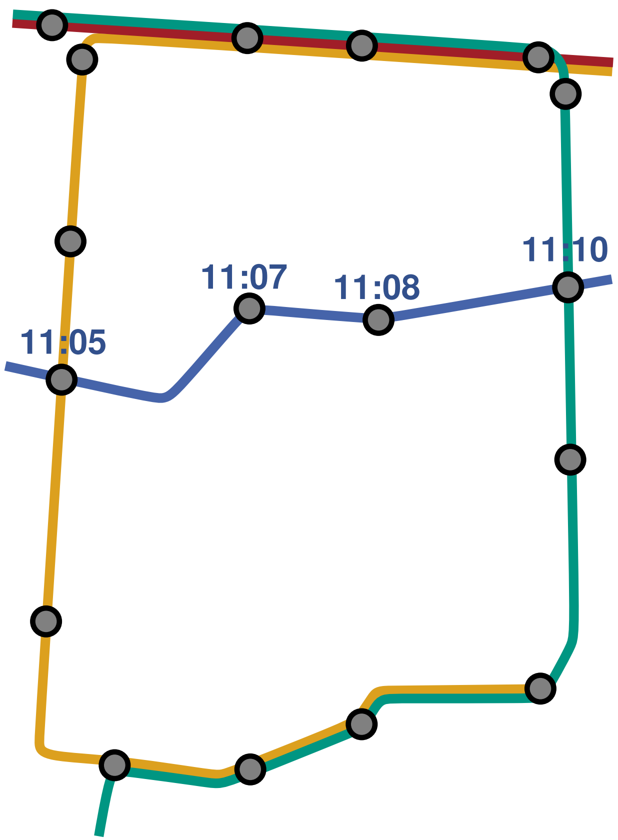
- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)



Problem Statement

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)

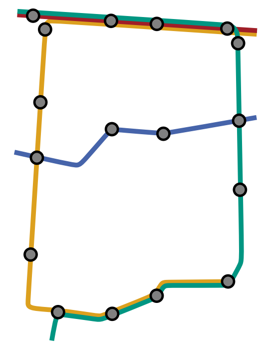


Problem Statement

Karlsruhe Institute of Technology

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)



- Transfer graph (non-schedule-based)

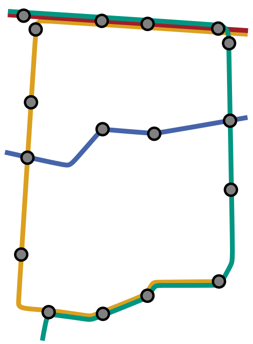
Institute of Theoretical Informatics
Algorithmics Group

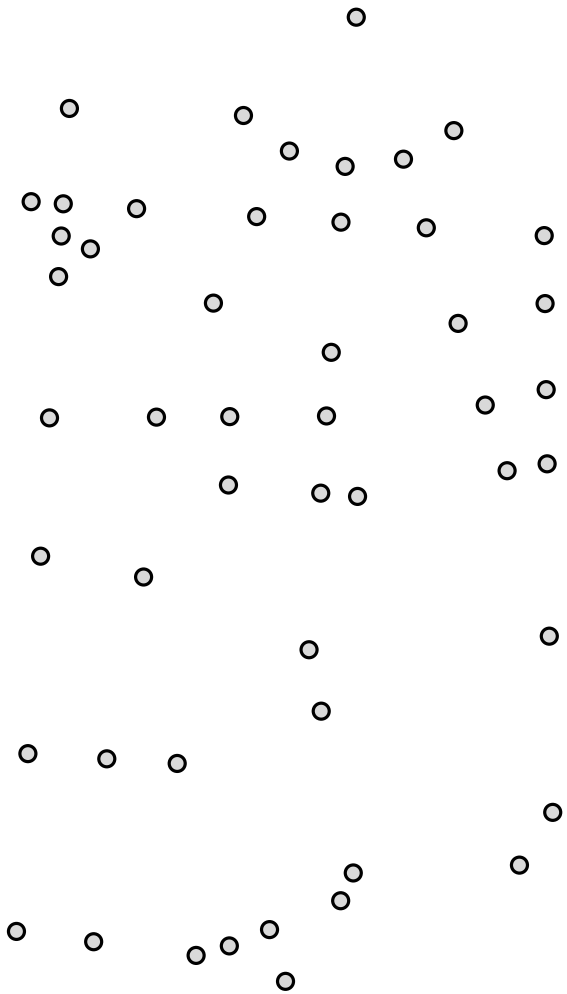
Problem Statement

Karlsruhe Institute of Technology

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)
- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)



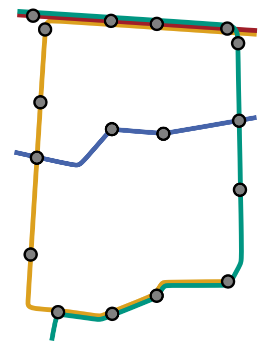


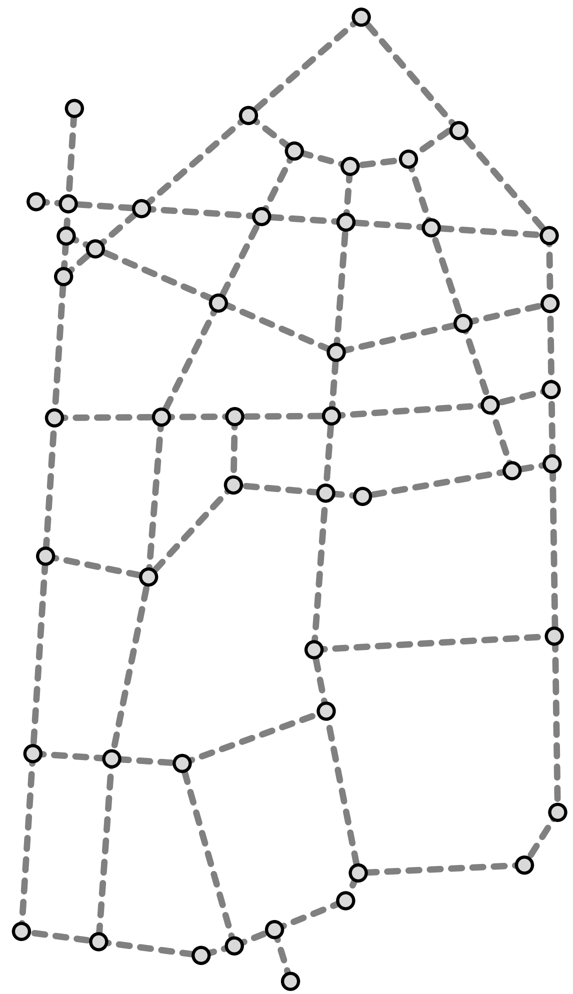
Institute of Theoretical Informatics
Algorithmics Group

Problem Statement

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)
- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)
  - Edges (roads, paths)





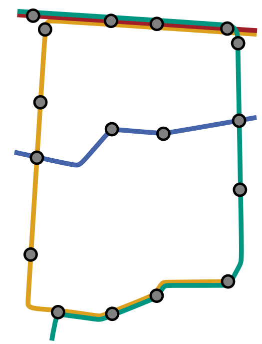
Institute of Theoretical Informatics
Algorithmics Group

Problem Statement

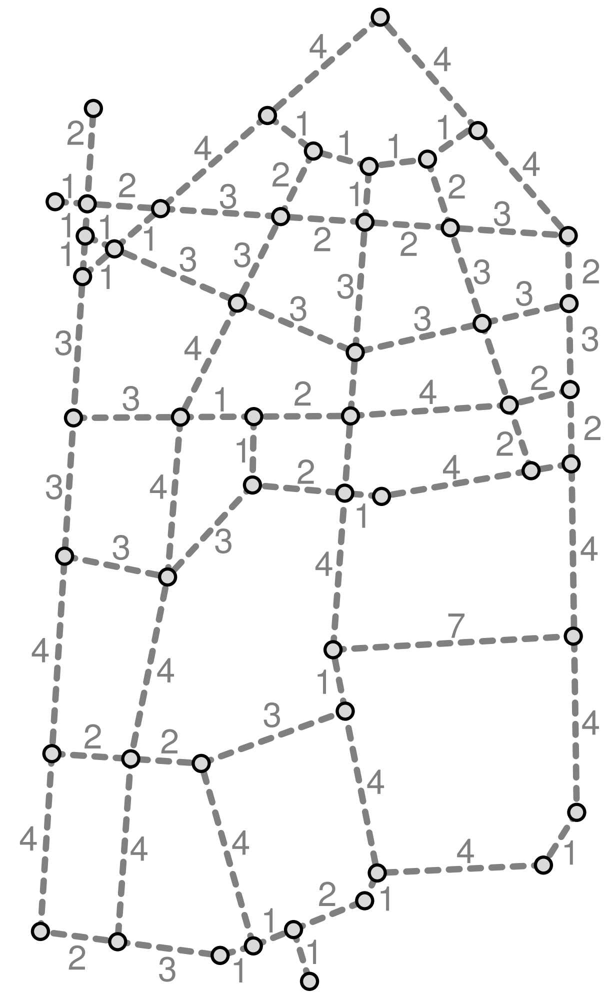
Karlsruhe Institute of Technology

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)



- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)
  - Edges (roads, paths)
  - Transfer times (e.g., walking)

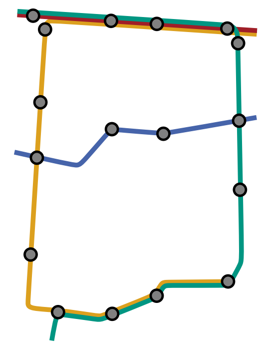


Institute of Theoretical Informatics
Algorithmics Group

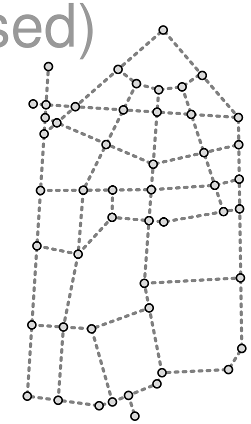
Problem Statement

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)



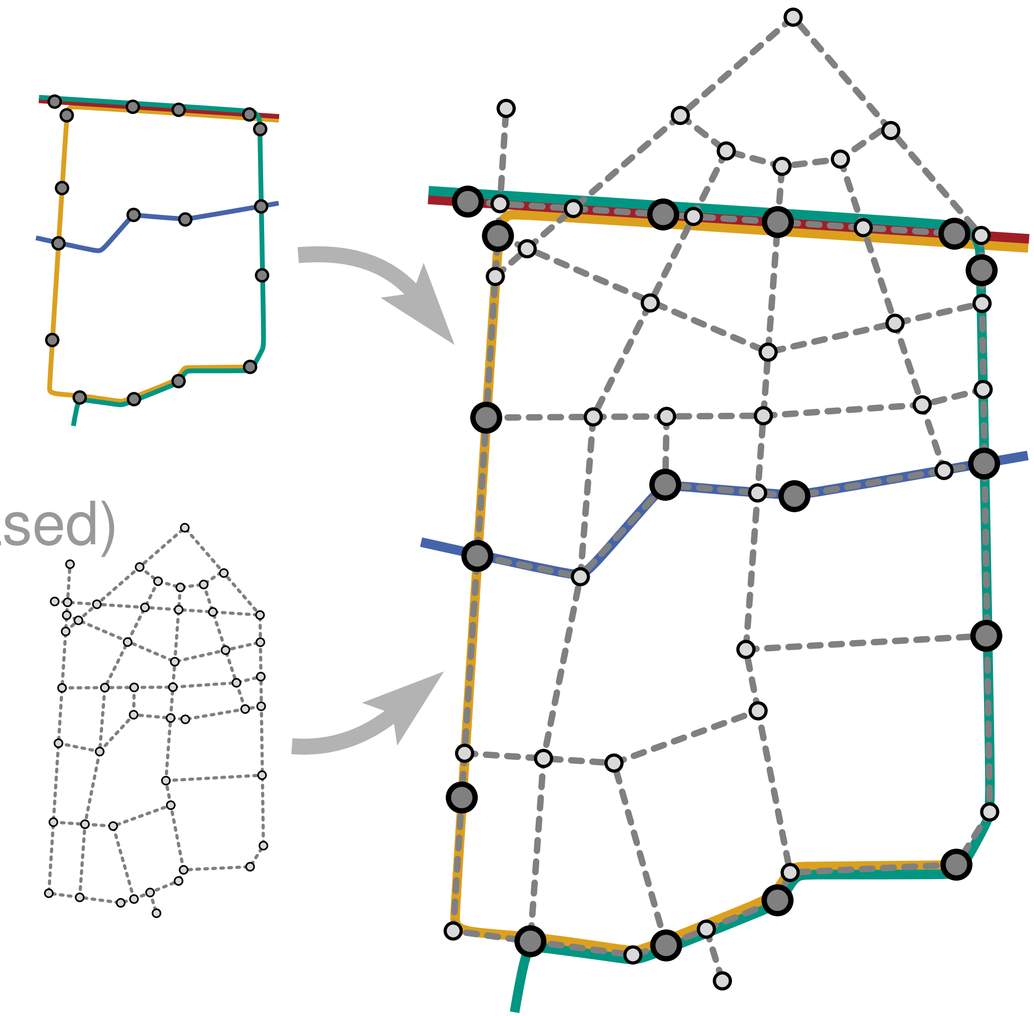
- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)
  - Edges (roads, paths)
  - Transfer times (e.g., walking)



Problem Statement

Given:

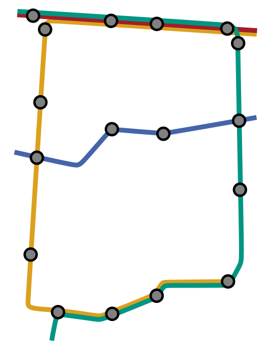
- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)
- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)
  - Edges (roads, paths)
  - Transfer times (e.g., walking)



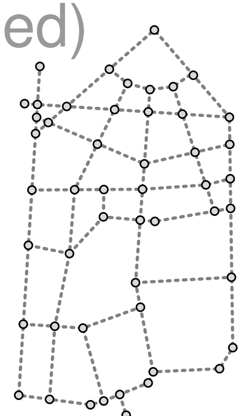
Problem Statement

Given:

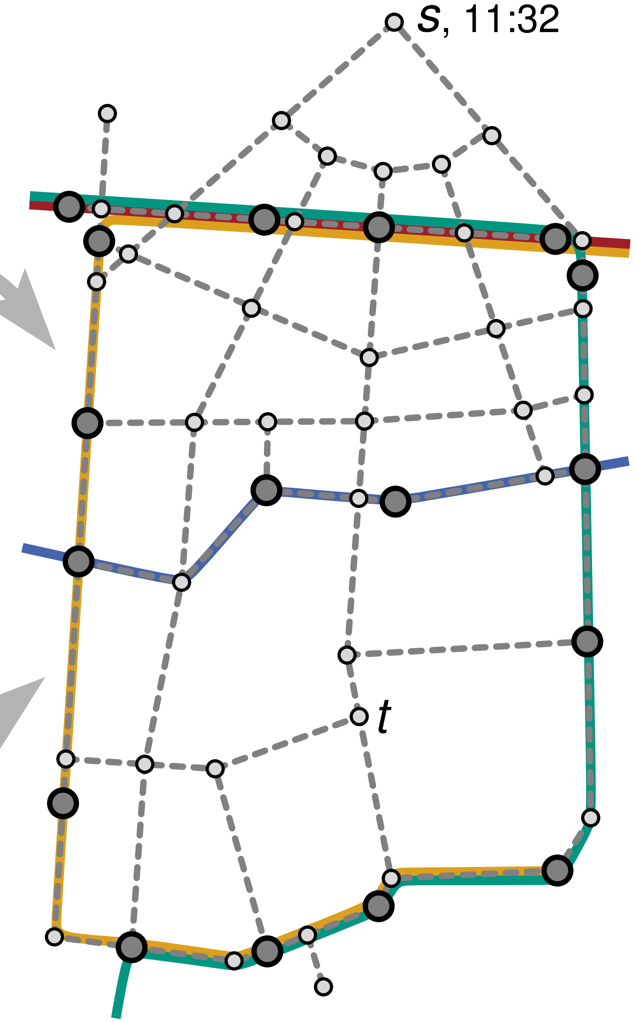
- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)



- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)
  - Edges (roads, paths)
  - Transfer times (e.g., walking)



- Source *s*, target *t*, and a departure time



Institute of Theoretical Informatics
Algorithmics Group

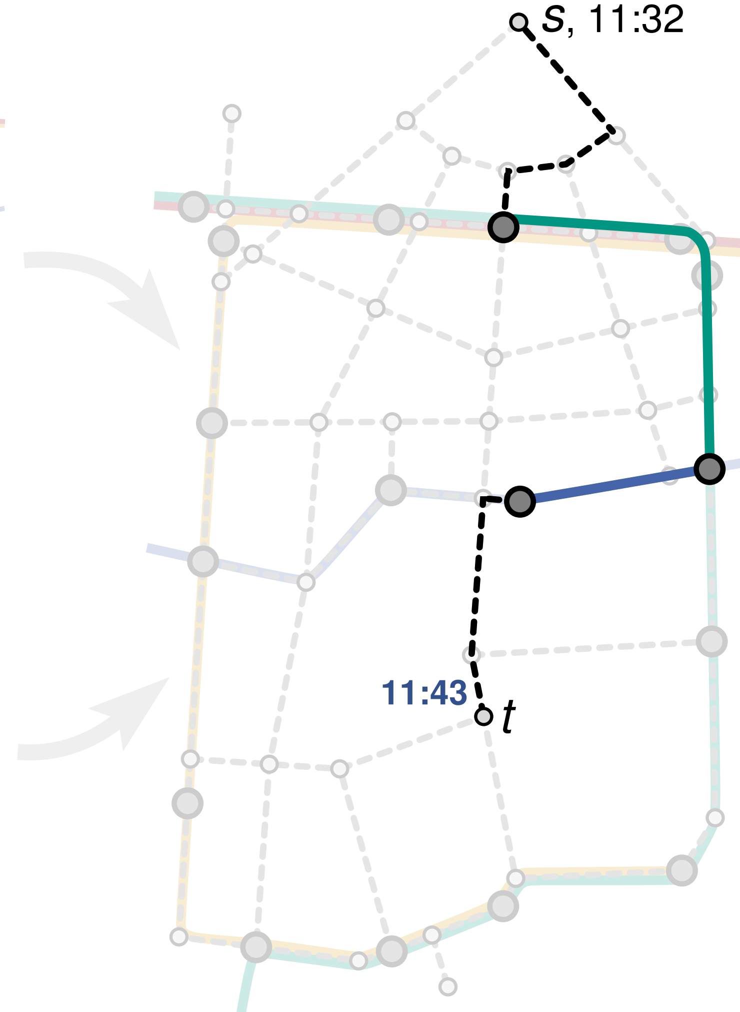
Problem Statement

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)

- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)
  - Edges (roads, paths)
  - Transfer times (e.g., walking)

- Source s, target t, and a departure time



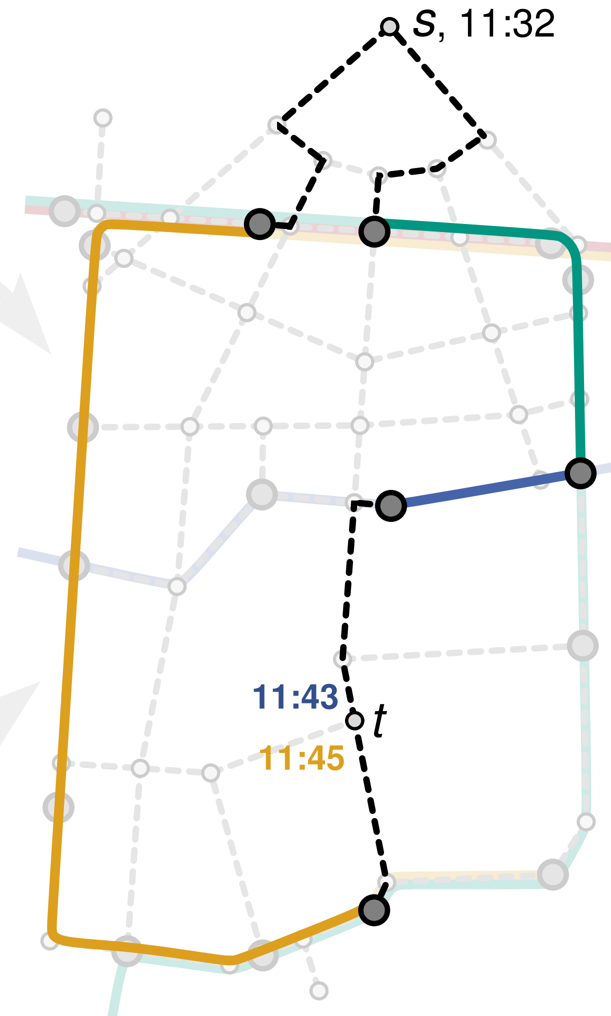
Problem Statement

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)

- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)
  - Edges (roads, paths)
  - Transfer times (e.g., walking)

- Source *s*, target *t*, and a departure time



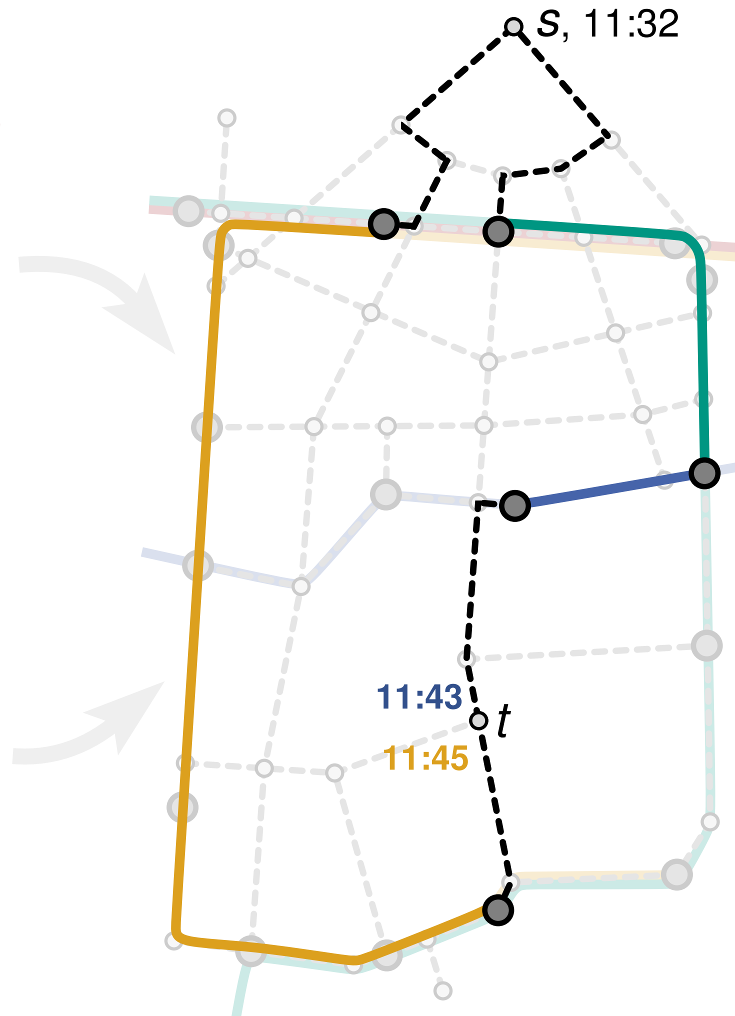
Problem Statement

Given:

- Public transit network (timetable)
  - Stops (bus stops, stations)
  - Routes (bus lines, train lines)
  - Trips (schedule of a vehicle)

- Transfer graph (non-schedule-based)
  - Vertices (crossings, places)
  - Edges (roads, paths)
  - Transfer times (e.g., walking)

- Source *s*, target *t*, and a departure time



Objective:

- Find a Pareto-set of journeys w.r.t. **arrival time** and **number of trips**

Institute of Theoretical Informatics
Algorithmics Group

Related Work

Public Transit:

- Restricted transfers, only between a few stops
- Transitively closed transfer graph:
  - RAPTOR (Delling et al. '14), CSA (Dibbelt et al. '14), Trip-Based (Witt '16)
  - Only feasible for up to 15 minutes of walking
- Only evaluated with limited transfers:
  - Transfer Patterns (Bast et al. '16), Frequency-Based (Bast, Storandt '14)

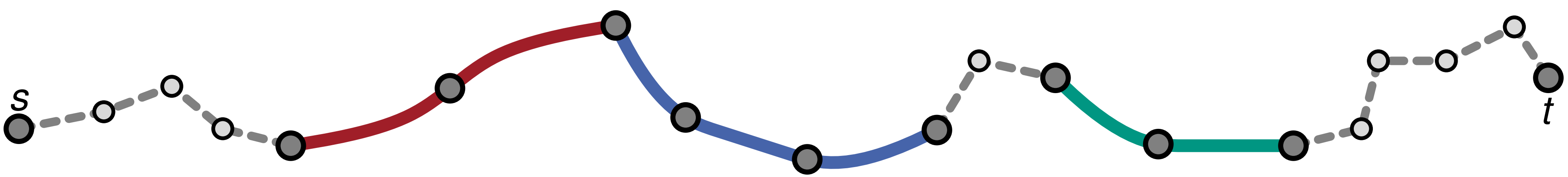
Multi-Modal:

- Interlace RAPTOR and Dijkstra: MCR (Delling et al '13)
- Has significant impact on travel times: (5% London – 40% Switzerland)
  - Profile-MCR (Wagner, Zündorf '17)
  - HLRaptor, HLCSA (Phan, Viennot '19)

Our Approach

Observation: (Sauer 2018)

- Long transfers are mostly useful at the source/target
- Transfers between two public transit routes are mostly short



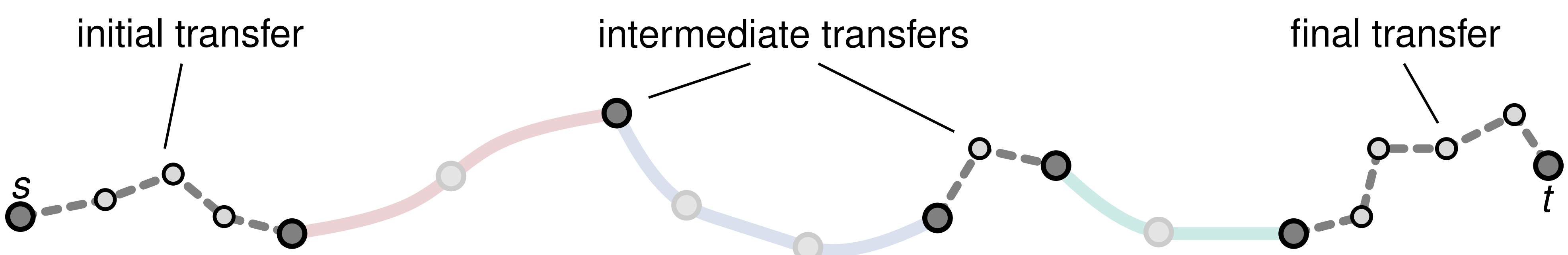
Our Approach

Observation: (Sauer 2018)

- Long transfers are mostly useful at the source/target
- Transfers between two public transit routes are mostly short

Idea:

- Process transfers differently based on their position in a journey
- We distinguish:
  - Initial transfers
  - Final transfers
  - Intermediate transfers



Our Approach – Transfer Handling

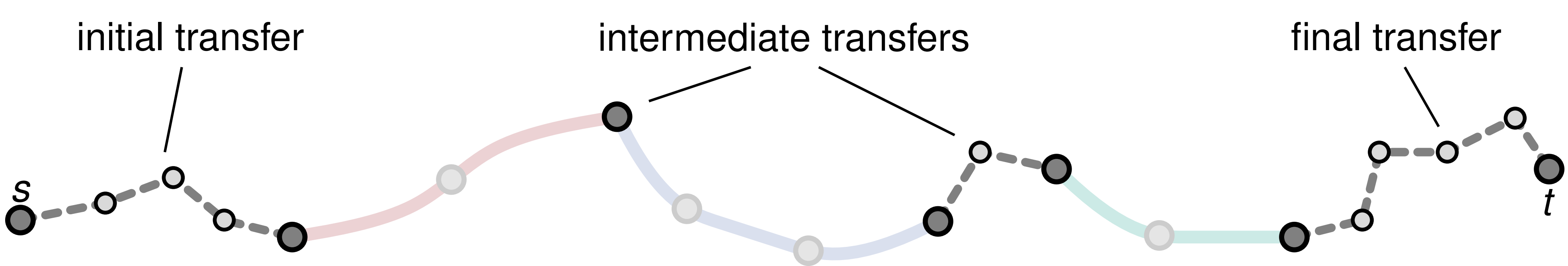
Karlsruhe Institute of Technology

Initial/Final Transfers:

- – Frequent
- – Often long

Intermediate Transfers:

- + Rare
- + Mostly short



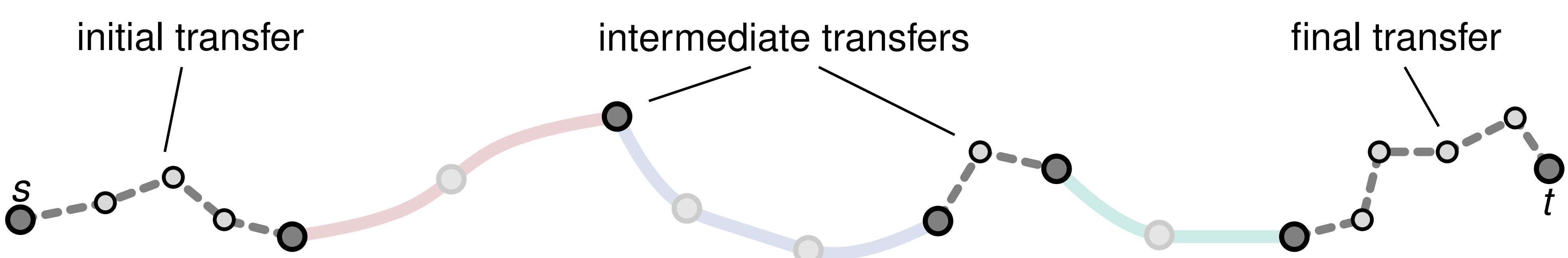
Our Approach – Transfer Handling

Initial/Final Transfers:

- − Frequent
- − Often long
- + One endpoint known (*s* or *t*)

Intermediate Transfers:

- + Rare
- + Mostly short
- − Both endpoints unknown



Our Approach – Transfer Handling

Initial/Final Transfers:

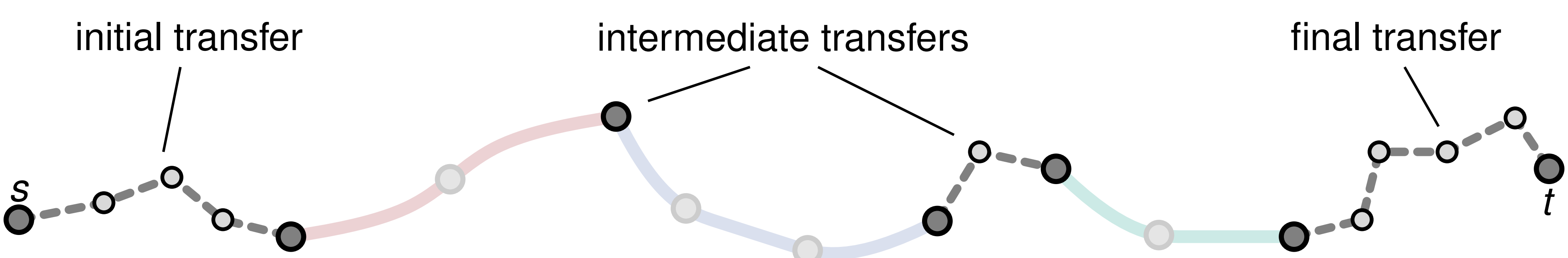
- − Frequent
- − Often long
- + One endpoint known (*s* or *t*)

- Use fast one-to-many queries
- Bucket-CH

Intermediate Transfers:

- + Rare
- + Mostly short
- − Both endpoints unknown

- Precompute **all** of them
- One-hop transfers during query



Our Approach – Transfer Handling

Initial/Final Transfers:

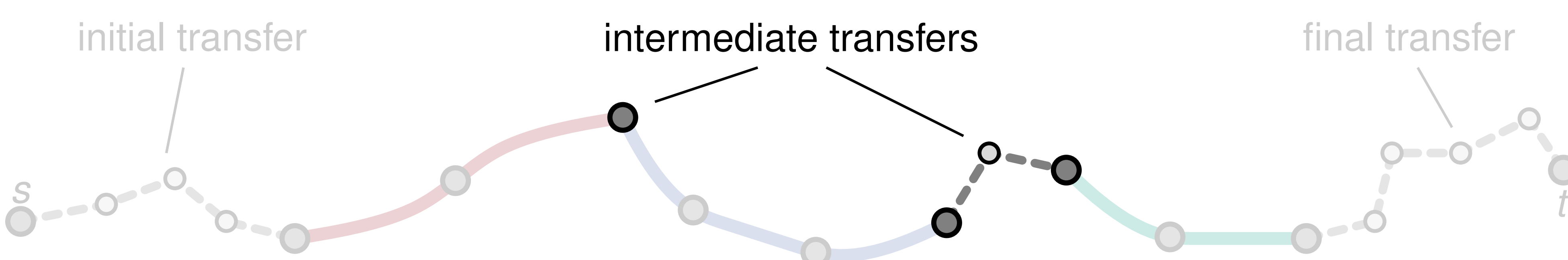
- − Frequent
- − Often long
- \+ One endpoint known (*s* or *t*)

- Use fast one-to-many queries
- Bucket-CH

Intermediate Transfers:

- \+ Rare
- \+ Mostly short
- − Both endpoints unknown

- Precompute **all** of them
- One-hop transfers during query

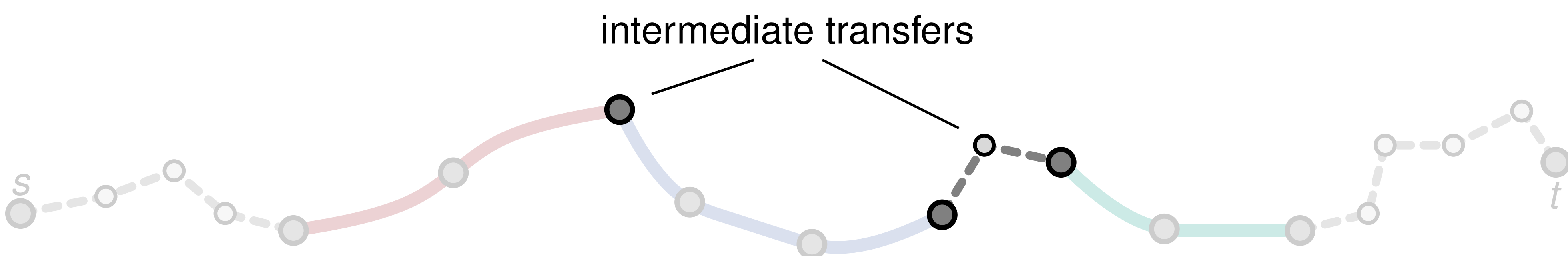


Our Approach – The Preprocessing Phase

Karlsruhe Institute of Technology

First Idea:

- Enumerate all Pareto-optimal journeys
- Collect the transfers used by them

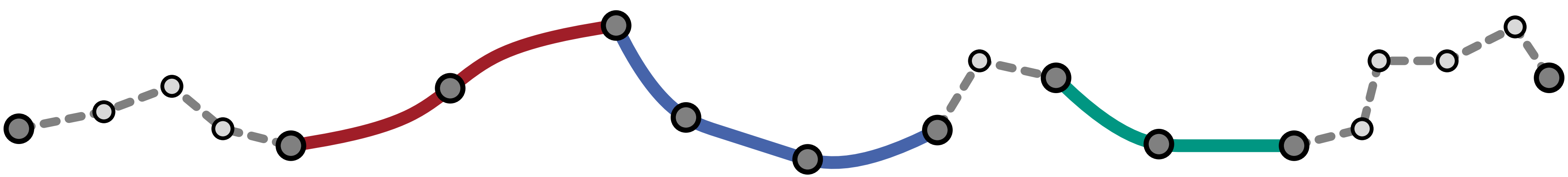


Our Approach – The Preprocessing Phase

Karlsruhe Institute of Technology

First Idea:

- Enumerate all Pareto-optimal journeys
- Collect the transfers used by them



Our Approach – The Preprocessing Phase

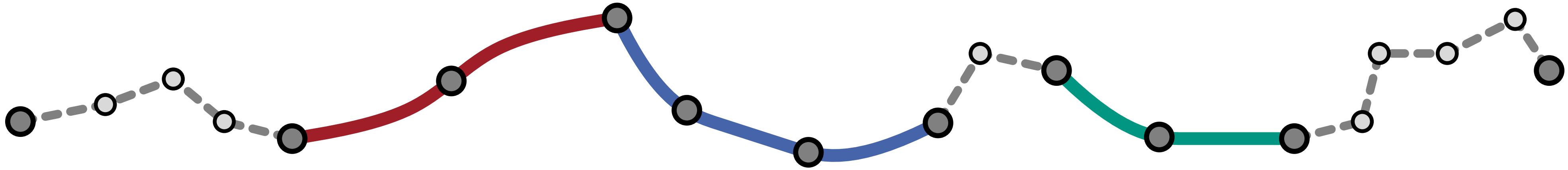
Karlsruhe Institute of Technology

First Idea:

- Enumerate all Pareto-optimal journeys
- Collect the transfers used by them

Improvements:

- Exploit the **subpath property**
- Enumerating journeys with exactly 2 trips is sufficient



Our Approach – The Preprocessing Phase

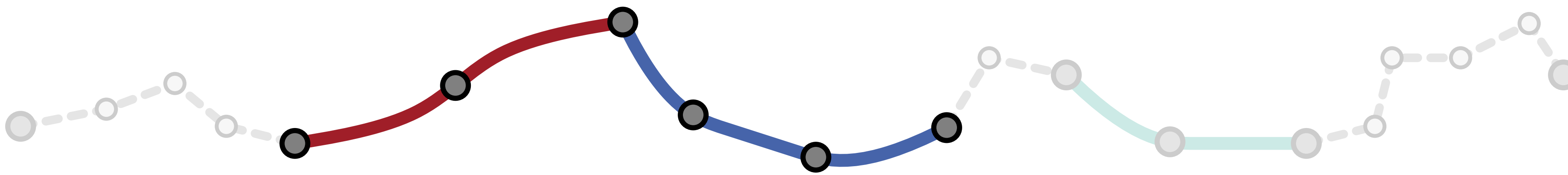
Karlsruhe Institute of Technology

First Idea:

- Enumerate all Pareto-optimal journeys
- Collect the transfers used by them

Improvements:

- Exploit the **subpath property**
- Enumerating journeys with exactly 2 trips is sufficient



Institute of Theoretical Informatics
Algorithmics Group

Our Approach – The Preprocessing Phase

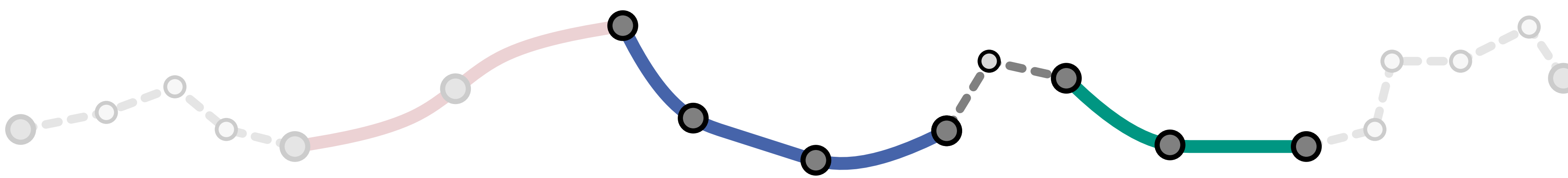
Karlsruhe Institute of Technology

First Idea:

- Enumerate all Pareto-optimal journeys
- Collect the transfers used by them

Improvements:

- Exploit the **subpath property**
- Enumerating journeys with exactly 2 trips is sufficient



Institute of Theoretical Informatics
Algorithmics Group

Our Approach – The Preprocessing Phase

First Idea:

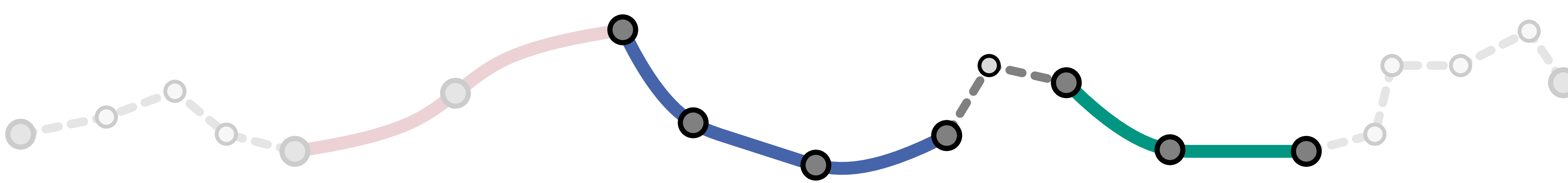
- Enumerate all Pareto-optimal journeys
- Collect the transfers used by them

Improvements:

- Exploit the **subpath property**
- Enumerating journeys with exactly 2 trips is sufficient

Implementation using multi-modal multi-criteria RAPTOR (MCR):

- RAPTOR runs in rounds, adding one trip per round
- Run range MCR from each stop restricted to two rounds



Our Approach – The Preprocessing Phase

First Idea:

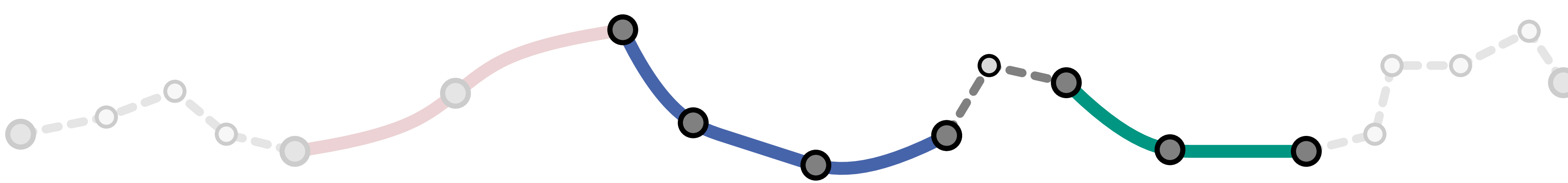
- Enumerate all Pareto-optimal journeys
- Collect the transfers used by them

Improvements:

- Exploit the **subpath property**
- Enumerating journeys with exactly 2 trips is sufficient

Implementation using multi-modal multi-criteria RAPTOR (MCR):

- RAPTOR runs in rounds, adding one trip per round
- Run range MCR from each stop restricted to two rounds

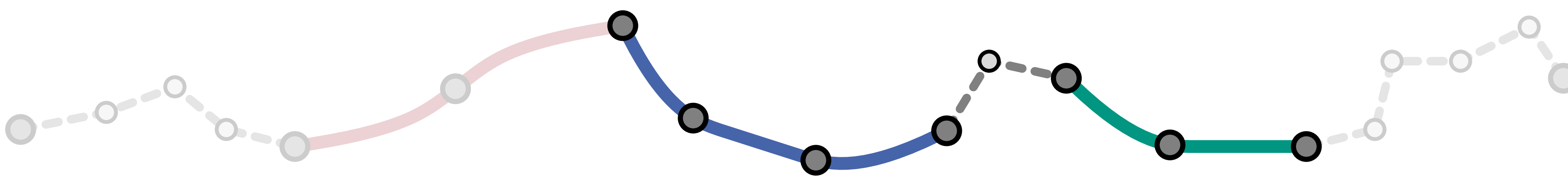


Our Approach – Shortcut Reduction

Karlsruhe Institute of Technology

Observation:

- Collecting all Pareto-optimal 2-trip journeys is superfluous
- A minimal dominating set of journeys is sufficient



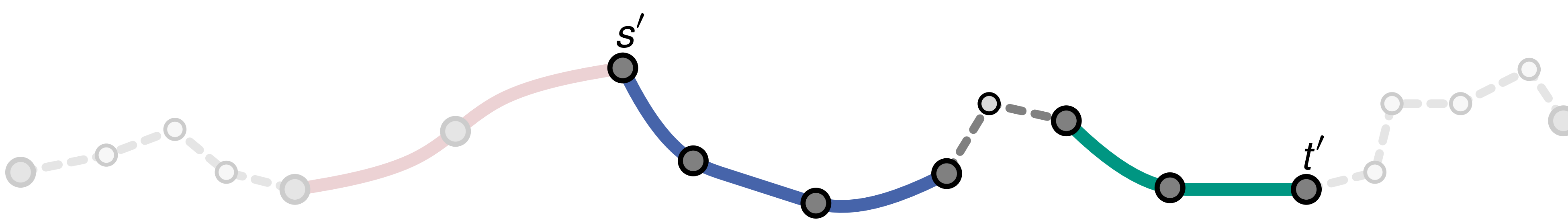
Our Approach – Shortcut Reduction

Observation:

- Collecting all Pareto-optimal 2-trip journeys is superfluous
- A minimal dominating set of journeys is sufficient

Implementation:

- Differentiate two types of journeys:
  - **Candidate** journeys have the form: trip – transfer – trip
  - **Witness** journeys are all other journeys with at most 2 trips



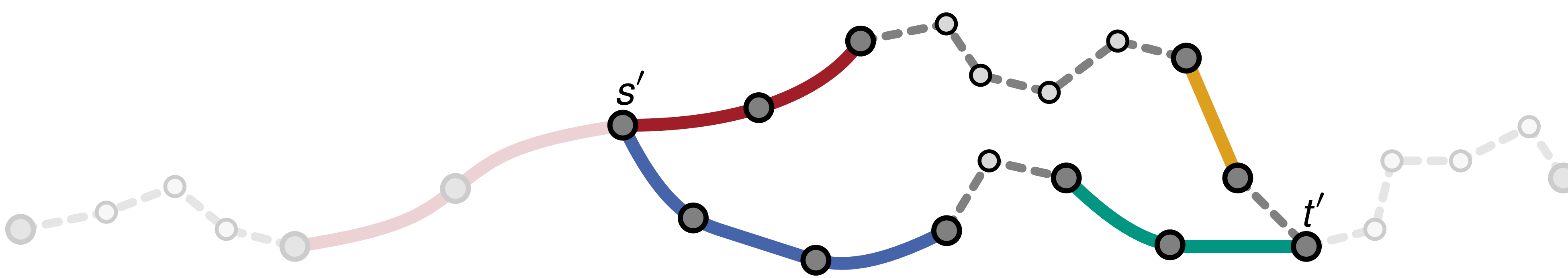
Our Approach – Shortcut Reduction

Observation:

- Collecting all Pareto-optimal 2-trip journeys is superfluous
- A minimal dominating set of journeys is sufficient

Implementation:

- Differentiate two types of journeys:
  - **Candidate** journeys have the form: trip – transfer – trip
  - **Witness** journeys are all other journeys with at most 2 trips



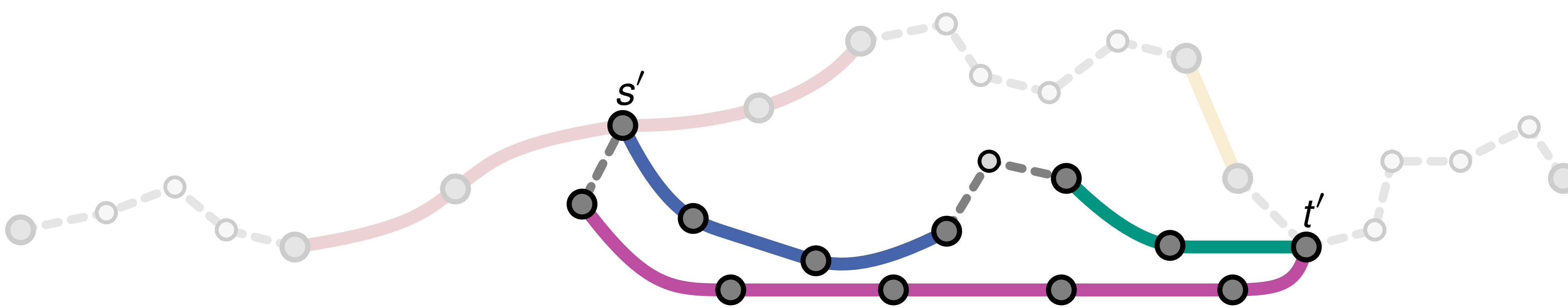
Our Approach – Shortcut Reduction

Observation:

- Collecting all Pareto-optimal 2-trip journeys is superfluous
- A minimal dominating set of journeys is sufficient

Implementation:

- Differentiate two types of journeys:
  - **Candidate** journeys have the form: trip – transfer – trip
  - **Witness** journeys are all other journeys with at most 2 trips



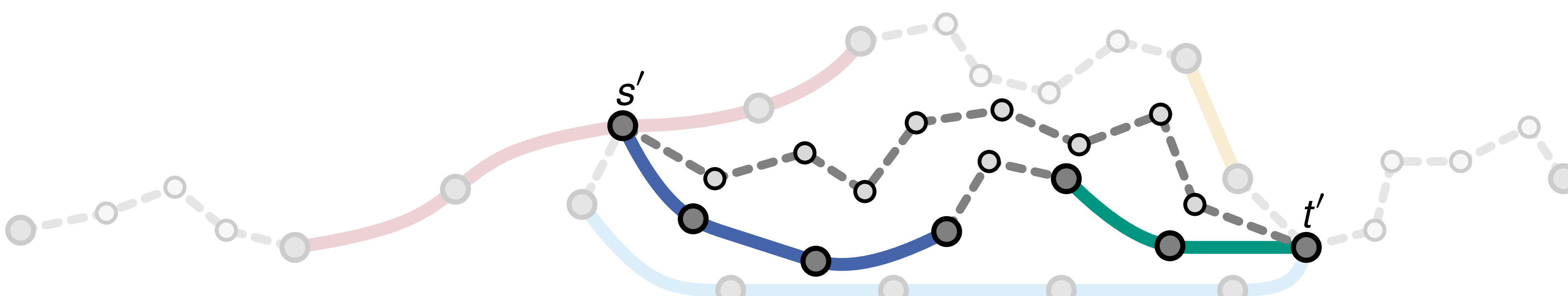
Our Approach – Shortcut Reduction

Observation:

- Collecting all Pareto-optimal 2-trip journeys is superfluous
- A minimal dominating set of journeys is sufficient

Implementation:

- Differentiate two types of journeys:
  - **Candidate** journeys have the form: trip – transfer – trip
  - **Witness** journeys are all other journeys with at most 2 trips



Our Approach – Shortcut Reduction

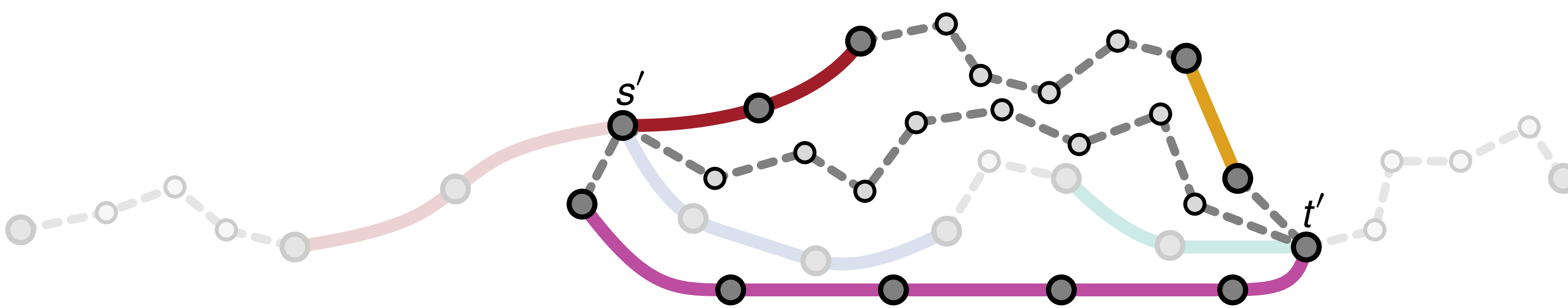
Observation:

- Collecting all Pareto-optimal 2-trip journeys is superfluous
- A minimal dominating set of journeys is sufficient

Implementation:

- Differentiate two types of journeys:
  - **Candidate** journeys have the form: trip – transfer – trip
  - **Witness** journeys are all other journeys with at most 2 trips

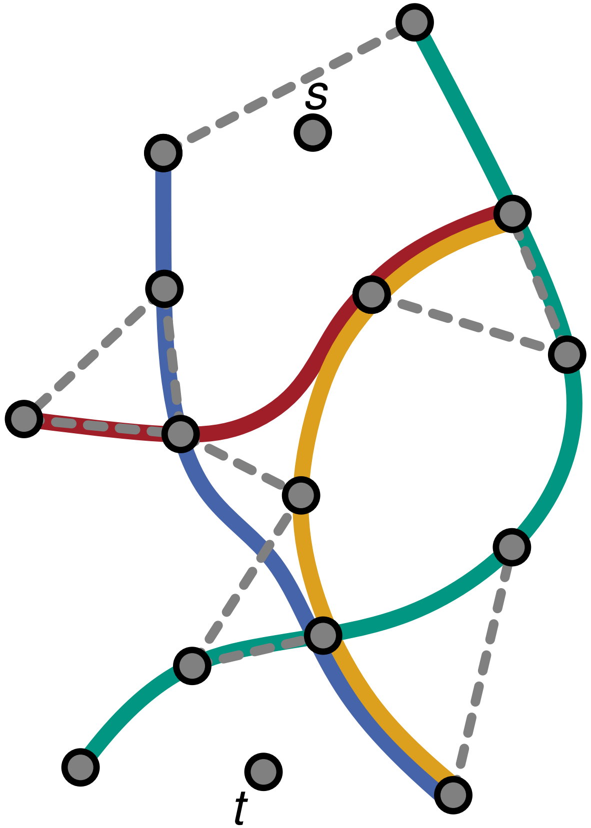
A Witness dominates a Candidate $\Leftrightarrow$ No shortcut needed



ULTRA Query Algorithms

Query Algorithm Outline:

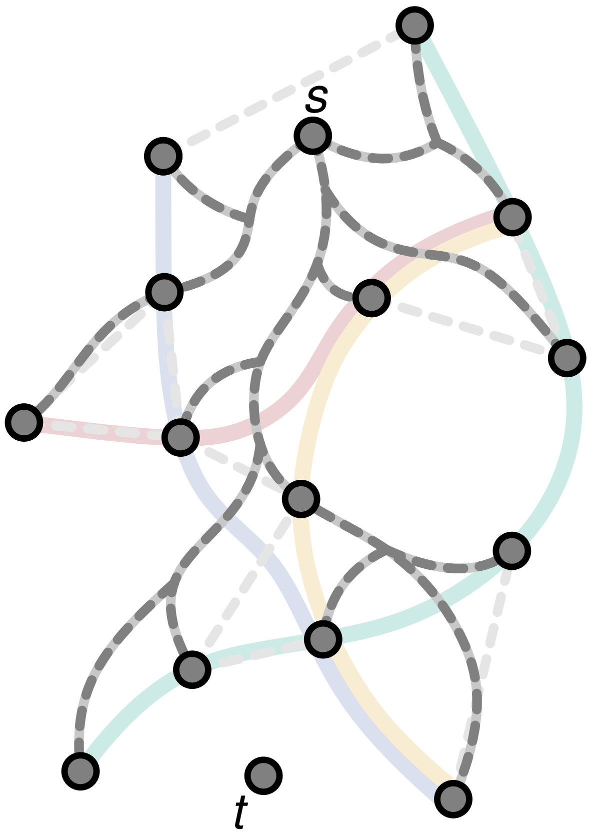
- Build a temporary transfer graph *G* including:
  - Preprocessed shortcuts
  - Initial transfers
  - Final transfers
- Run any query algorithm on: (Timetable, *G*)



ULTRA Query Algorithms

Query Algorithm Outline:

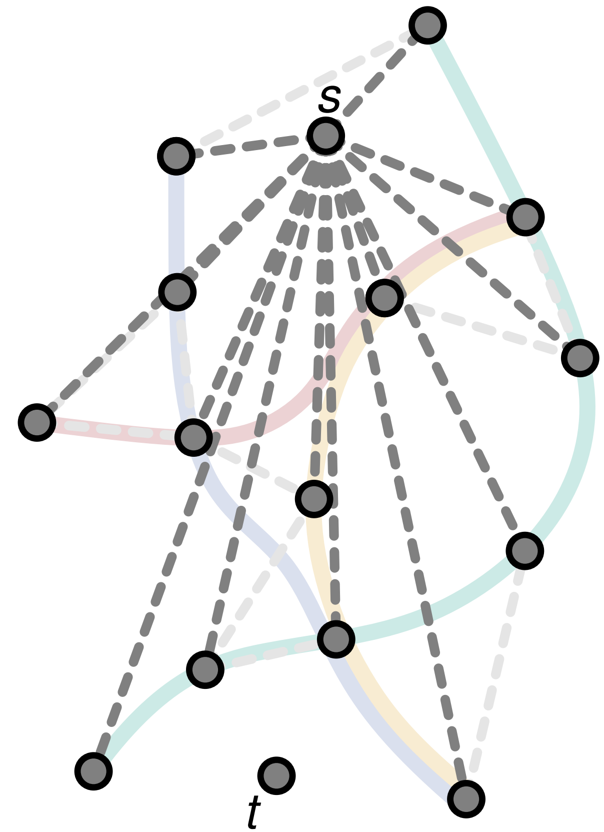
- Build a temporary transfer graph *G* including:
  - Preprocessed shortcuts
  - Initial transfers
  - Final transfers
- Run any query algorithm on: (Timetable, *G*)
- Use one-to-many query for initial/final transfers
  - One = source/target
  - Many = public transit stops
  - Fast implementation: Bucket-CH



ULTRA Query Algorithms

Query Algorithm Outline:

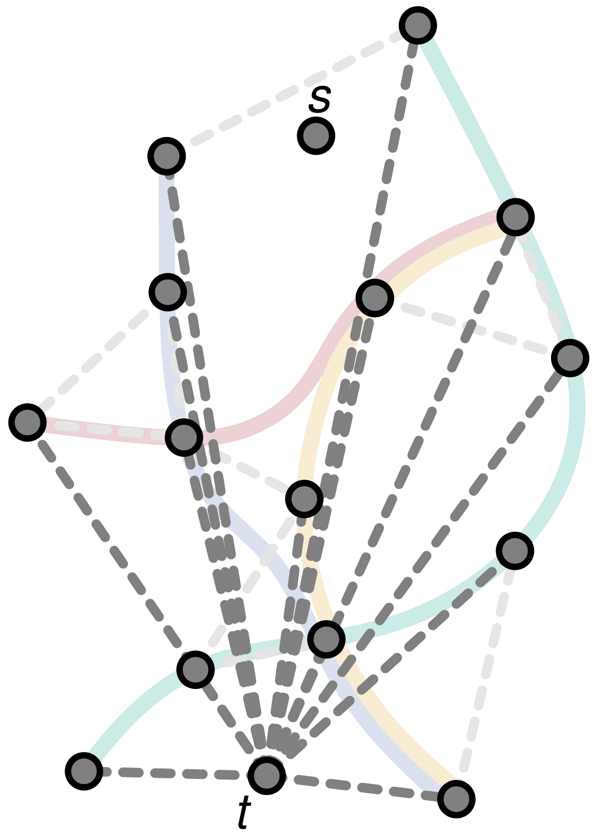
- Build a temporary transfer graph *G* including:
  - Preprocessed shortcuts
  - Initial transfers
  - Final transfers
- Run any query algorithm on: (Timetable, *G*)
- Use one-to-many query for initial/final transfers
  - One = source/target
  - Many = public transit stops
  - Fast implementation: Bucket-CH



ULTRA Query Algorithms

Query Algorithm Outline:

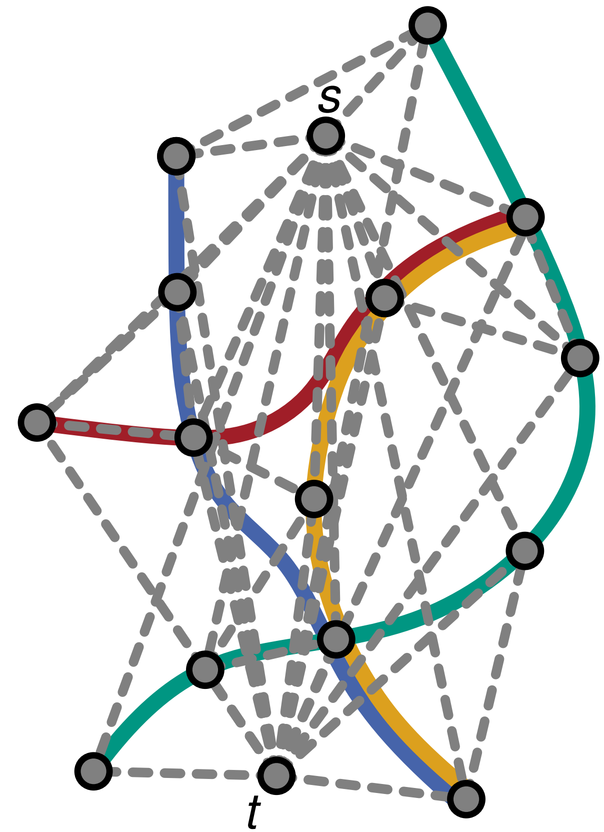
- Build a temporary transfer graph *G* including:
  - Preprocessed shortcuts
  - Initial transfers
  - Final transfers
- Run any query algorithm on: (Timetable, *G*)
- Use one-to-many query for initial/final transfers
  - One = source/target
  - Many = public transit stops
  - Fast implementation: Bucket-CH



ULTRA Query Algorithms

Query Algorithm Outline:

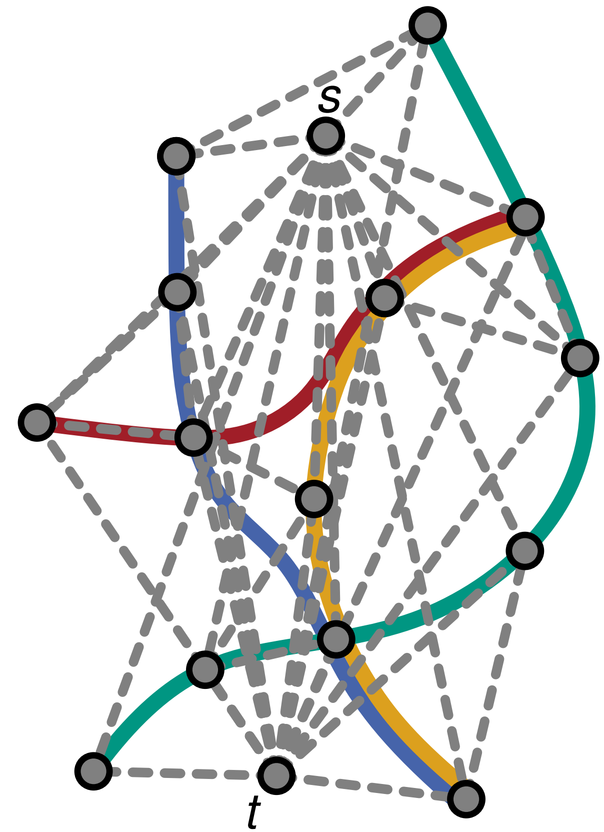
- Build a temporary transfer graph *G* including:
  - Preprocessed shortcuts
  - Initial transfers
  - Final transfers
- Run any query algorithm on: (Timetable, *G*)
- Use one-to-many query for initial/final transfers
  - One = source/target
  - Many = public transit stops
  - Fast implementation: Bucket-CH



ULTRA Query Algorithms

Query Algorithm Outline:

- Build a temporary transfer graph *G* including:
  - Preprocessed shortcuts
  - Initial transfers
  - Final transfers
- Run any query algorithm on: (Timetable, *G*)
- Use one-to-many query for initial/final transfers
  - One = source/target
  - Many = public transit stops
  - Fast implementation: Bucket-CH



Observations:

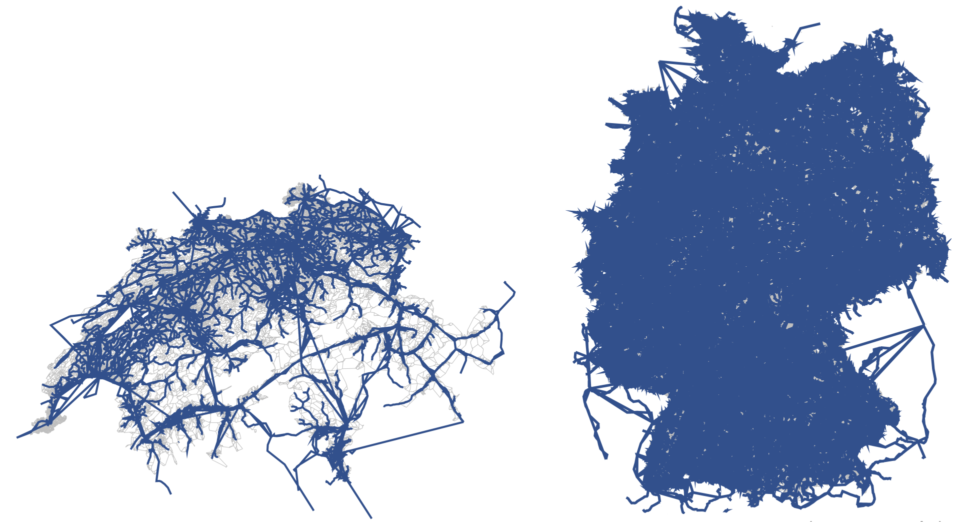
- Approach is independent of the used public transit query algorithm
- Knowing the algorithm can enable direct integration

Experimental Evaluation

Karlsruhe Institute of Technology

Instances:

- Timetables comprising two days
  - Switzerland (GTFS feed)
  - Germany (from DB)
- OpenStreetMap transfer graphs
  - Streets and pedestrian zones
  - Speed limits
- Transitive graphs for comparison
  - Limited maximum distance
  - Avg. degree ≈ 100



(not to scale)

Network	Switzerland	Germany
Stops	25 426	244 055
Routes	13 934	231 089
Trips	369 534	2 387 297
Stop events	4 740 929	48 495 169
Vertices	604 167	6 872 105
Full edges	1 847 140	21 372 360
Transitive edges	4 687 016	22 645 480

Experimental Evaluation – Preprocessing

Karlsruhe Institute of Technology

Setup:

- Switzerland with different speeds for the transfer graph
- In parallel on a machine with 16 cores

Result:

- Parallel speed-ups for walking as transfer mode (4.5 km/h)

Number of threads	1	2	4	8	16
Preprocessing time [mm:ss]	2:00:56	58:03	31:11	17:29	10:12
Speed-up factor	1	2.08	3.88	6.92	11.85

Experimental Evaluation – Preprocessing

Karlsruhe Institute of Technology

Setup:

- Switzerland with different speeds for the transfer graph
- In parallel on a machine with 16 cores

Result:

- Parallel speed-ups for walking as transfer mode (4.5 km/h)

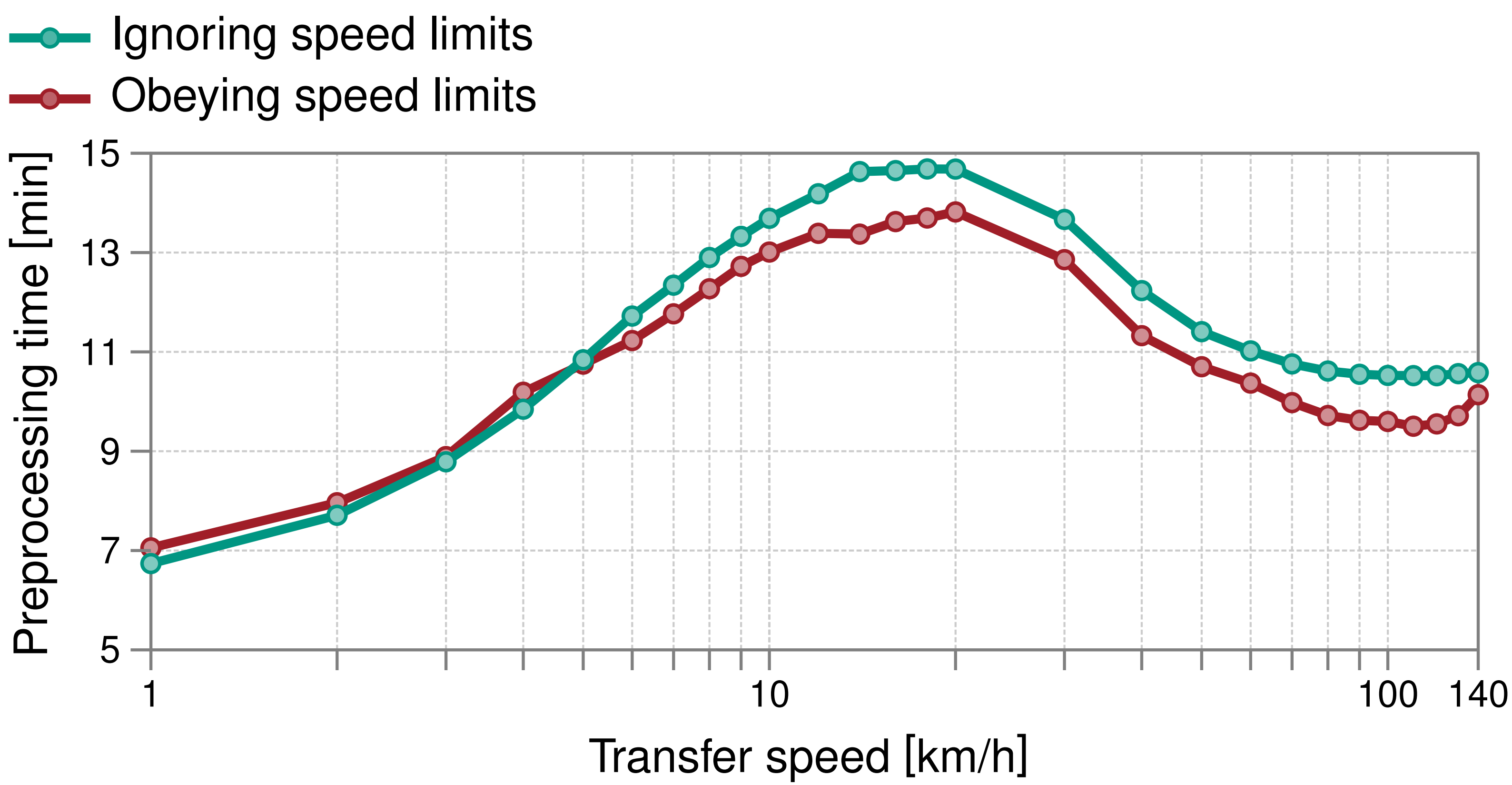
Number of threads	1	2	4	8	16
Preprocessing time [mm:ss]	2:00:56	58:03	31:11	17:29	10:12
Speed-up factor	1	2.08	3.88	6.92	11.85

Experimental Evaluation – Preprocessing

Setup:

- Switzerland with different speeds for the transfer graph
- In parallel on a machine with 16 cores

Result:



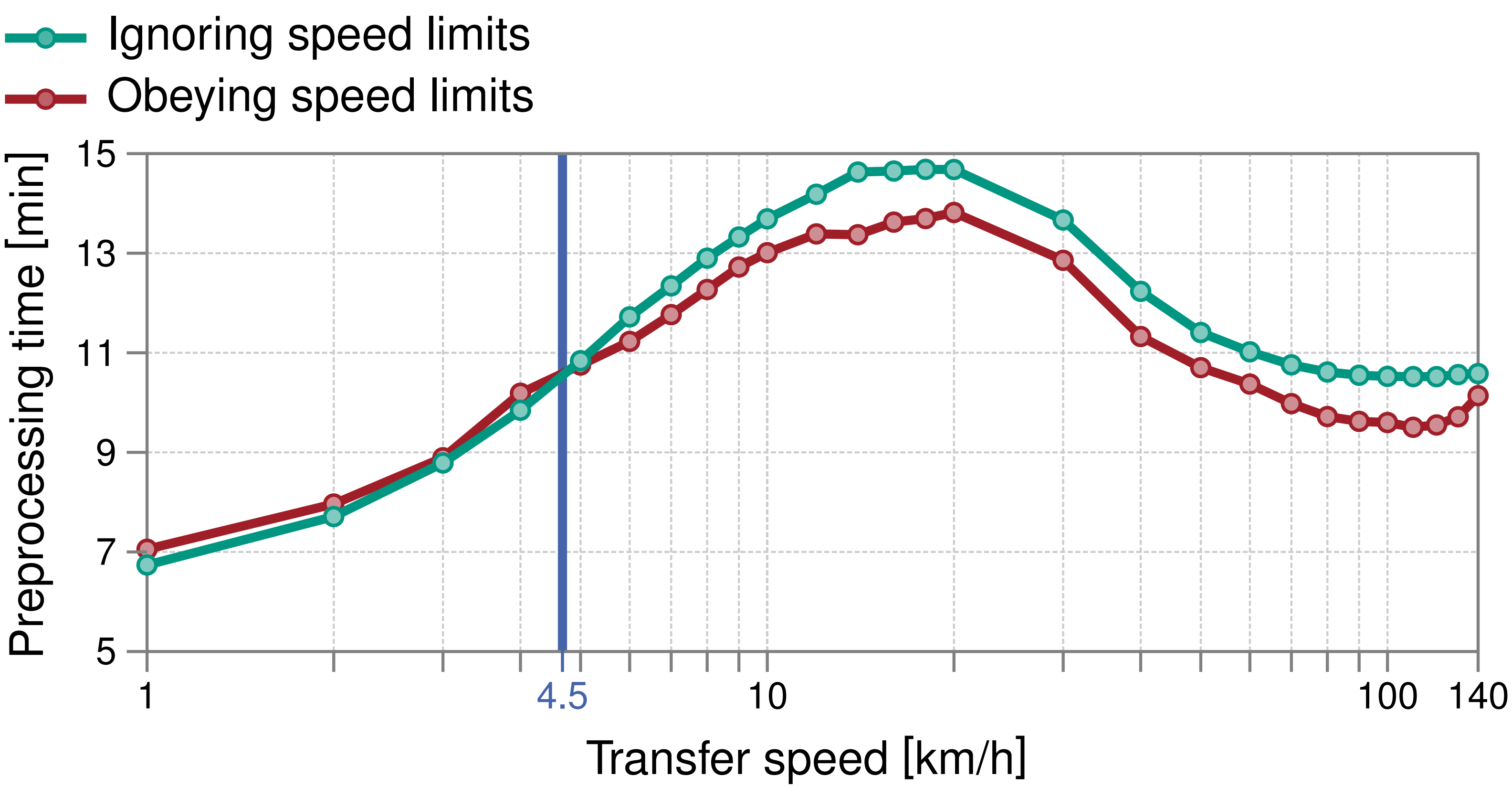
Institute of Theoretical Informatics
Algorithmics Group

Experimental Evaluation – Preprocessing

Setup:

- Switzerland with different speeds for the transfer graph
- In parallel on a machine with 16 cores

Result:

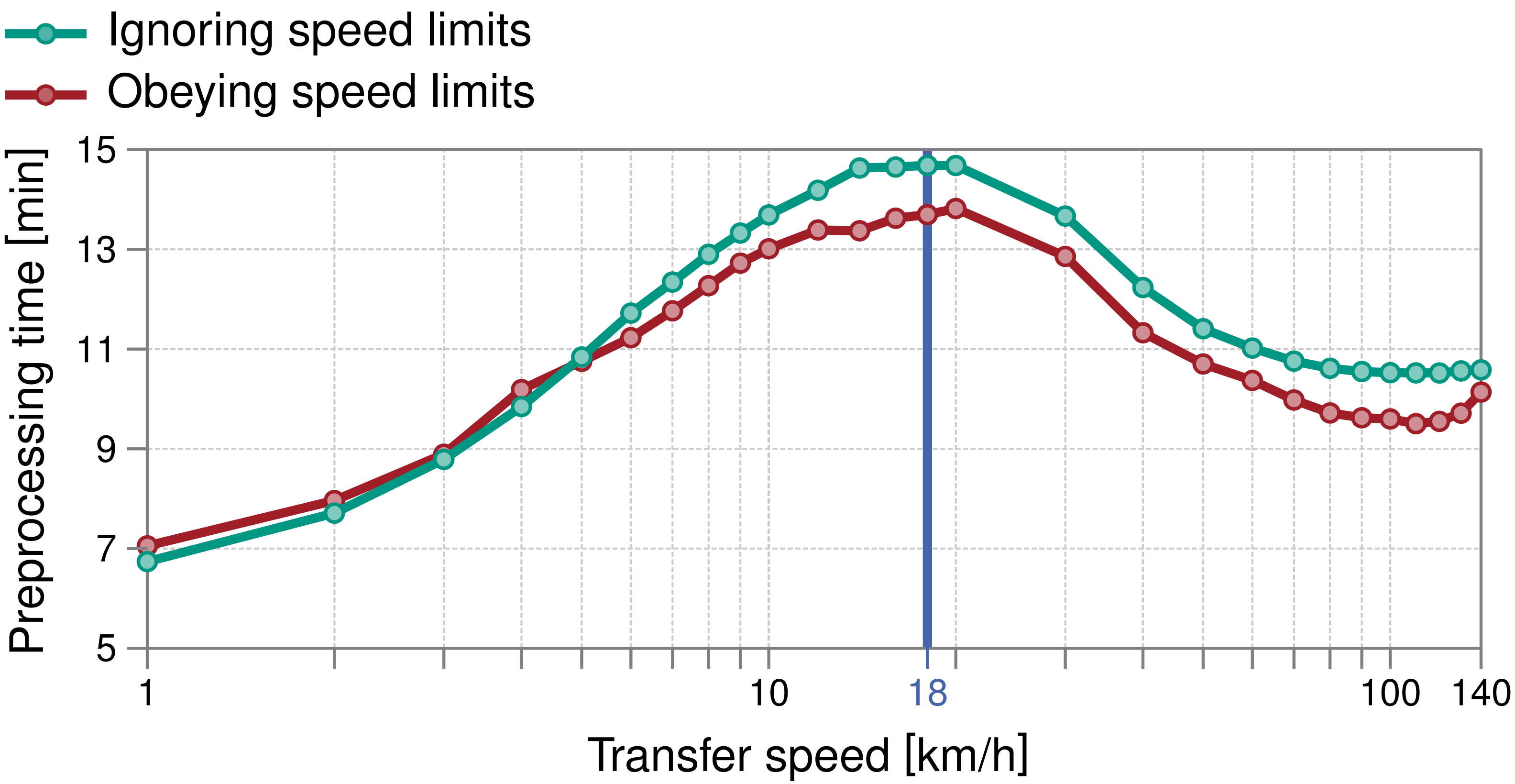


Experimental Evaluation – Preprocessing

Setup:

- Switzerland with different speeds for the transfer graph
- In parallel on a machine with 16 cores

Result:



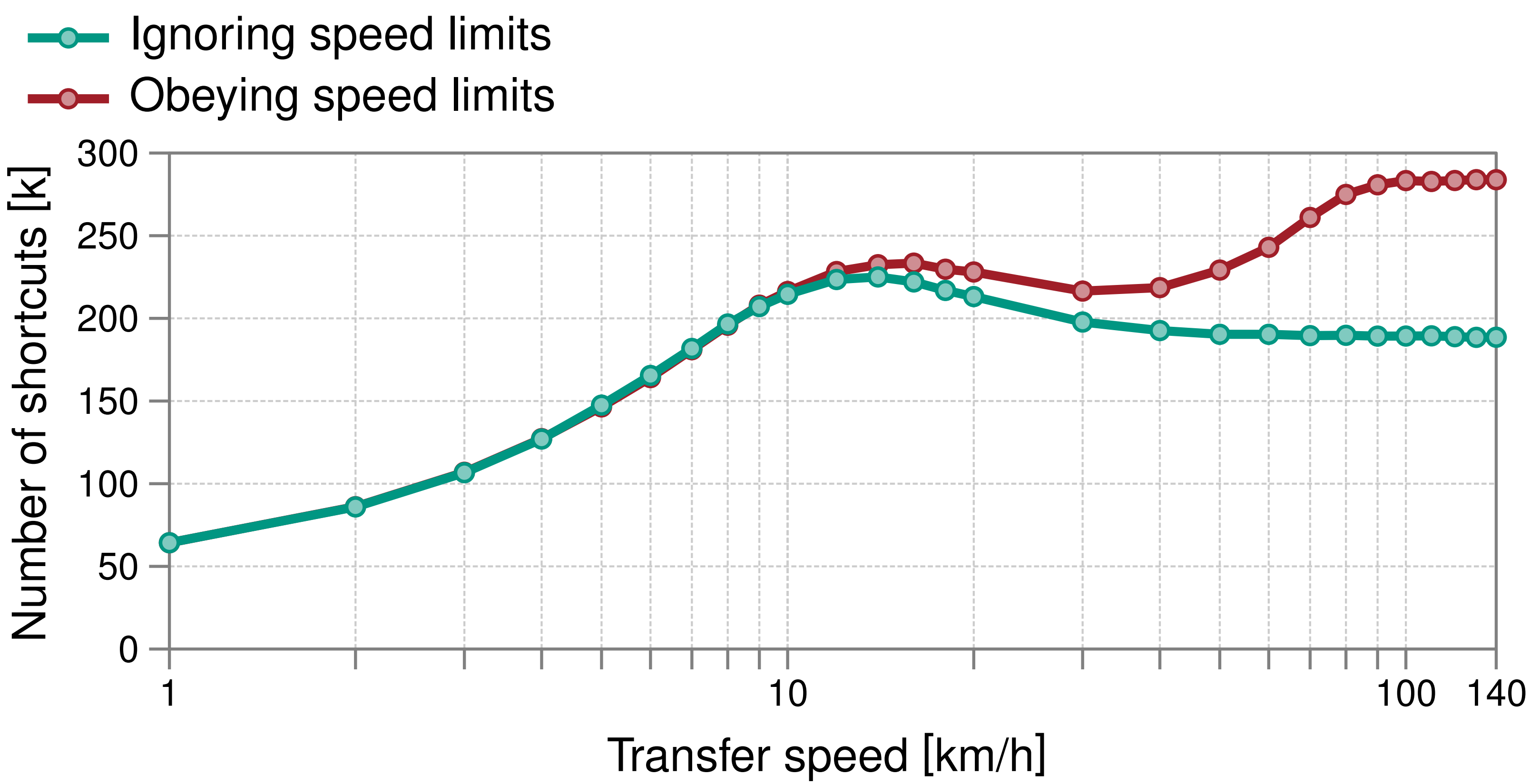
Institute of Theoretical Informatics
Algorithmics Group

Experimental Evaluation – Preprocessing

Setup:

- Switzerland with different speeds for the transfer graph

Result:

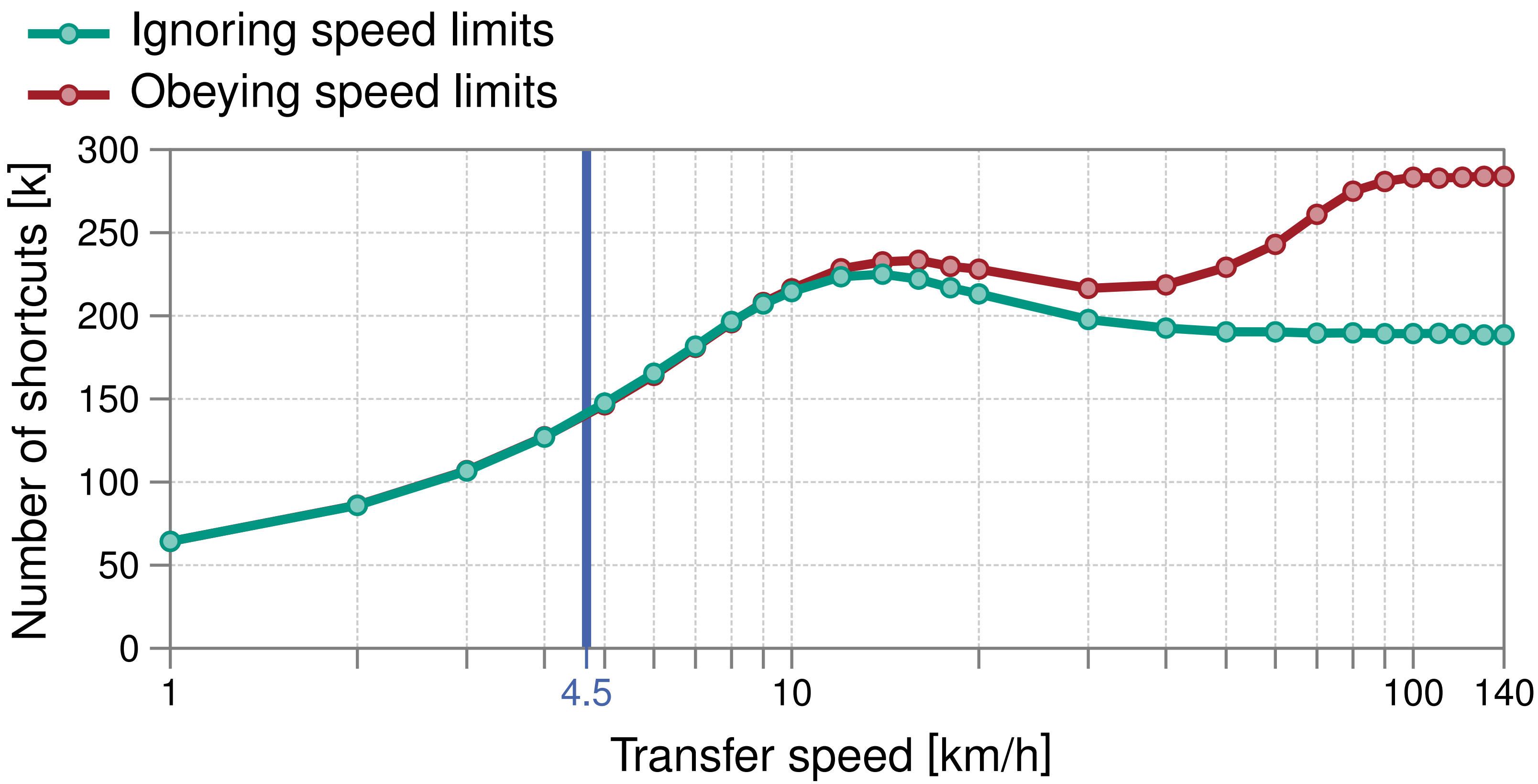


Experimental Evaluation – Preprocessing

Setup:

- Switzerland with different speeds for the transfer graph

Result:



Institute of Theoretical Informatics
Algorithmics Group

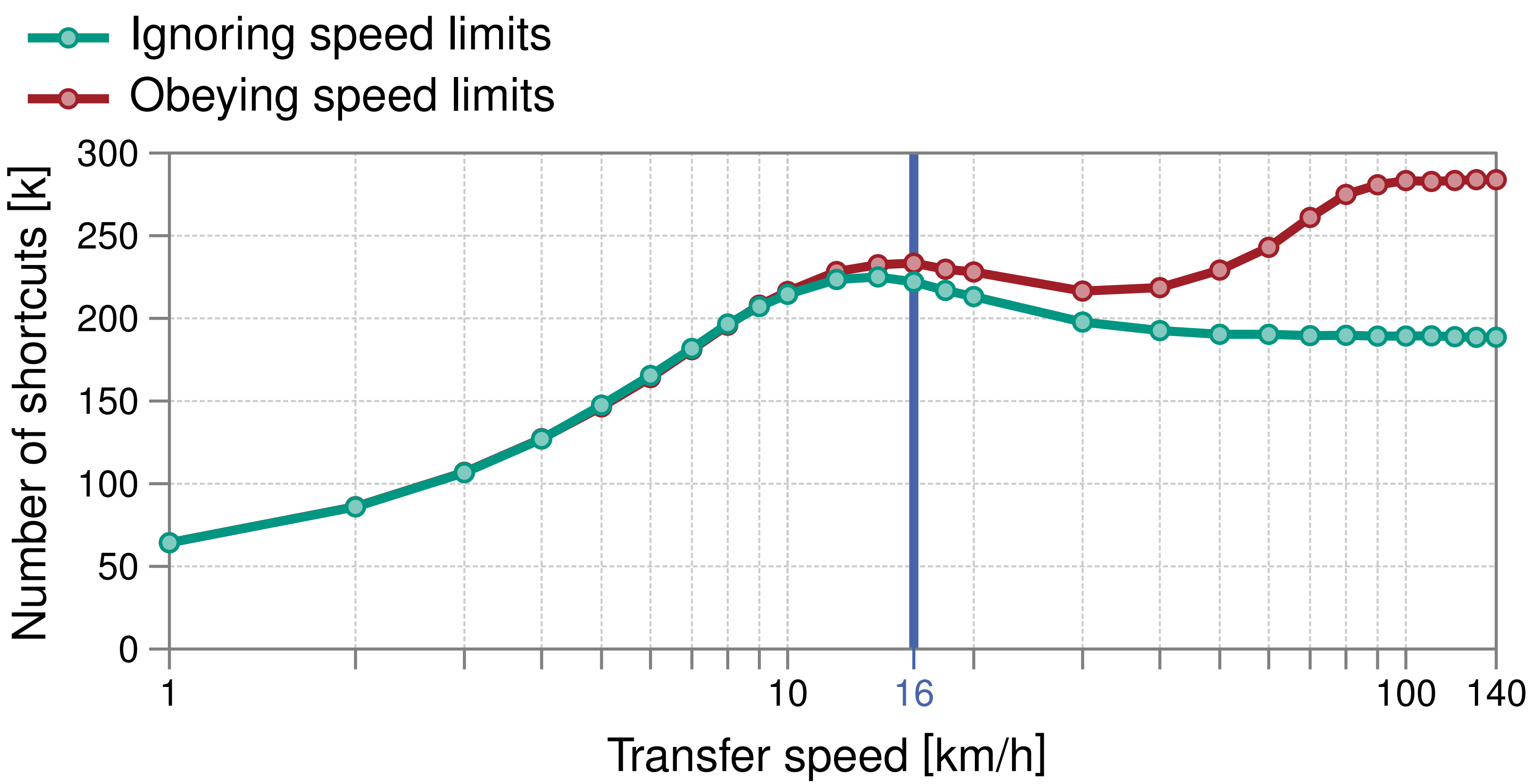
Experimental Evaluation – Preprocessing

Karlsruhe Institute of Technology

Setup:

- Switzerland with different speeds for the transfer graph

Result:



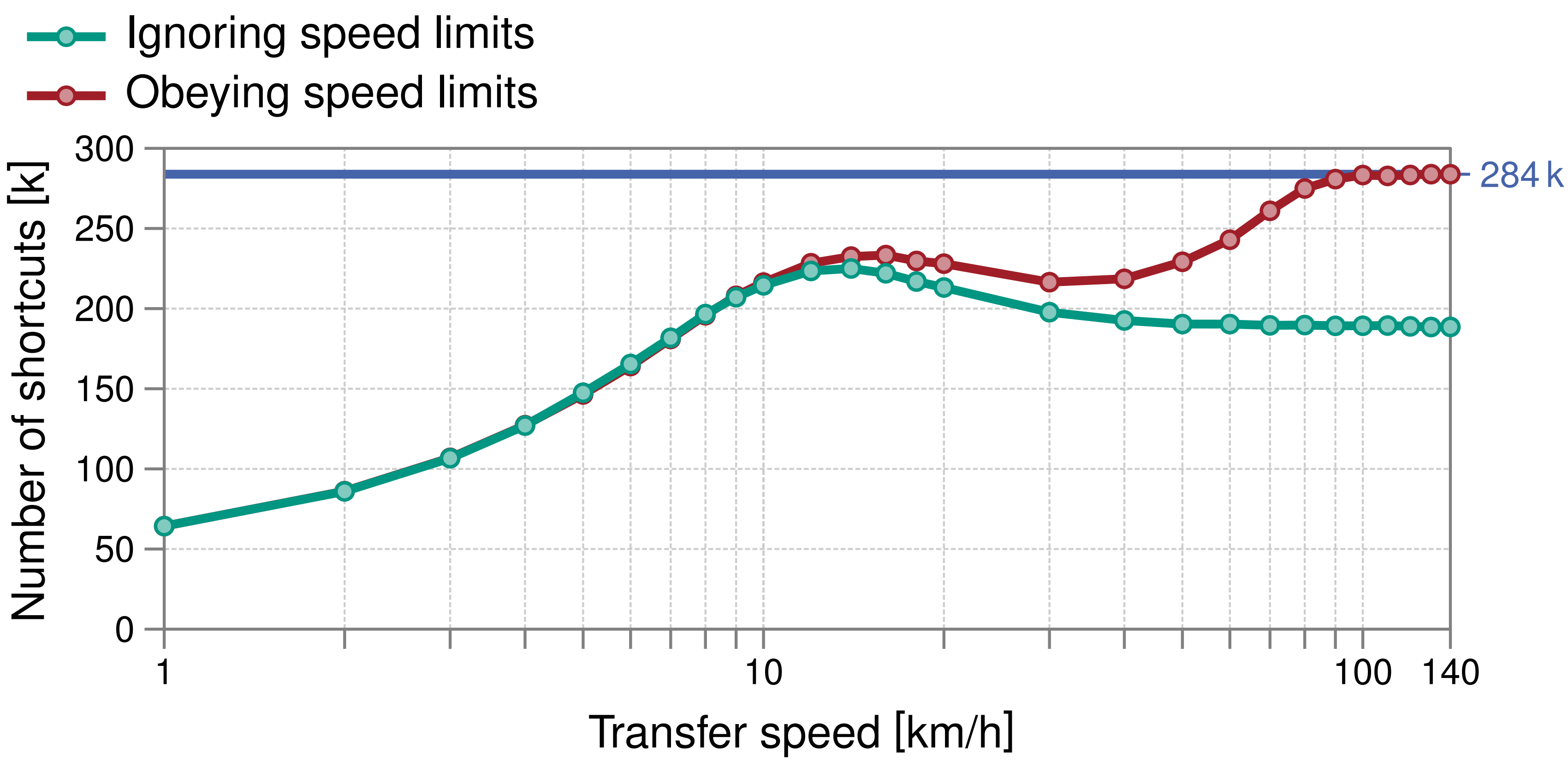
Experimental Evaluation – Preprocessing

Karlsruhe Institute of Technology

Setup:

- Switzerland with different speeds for the transfer graph

Result:



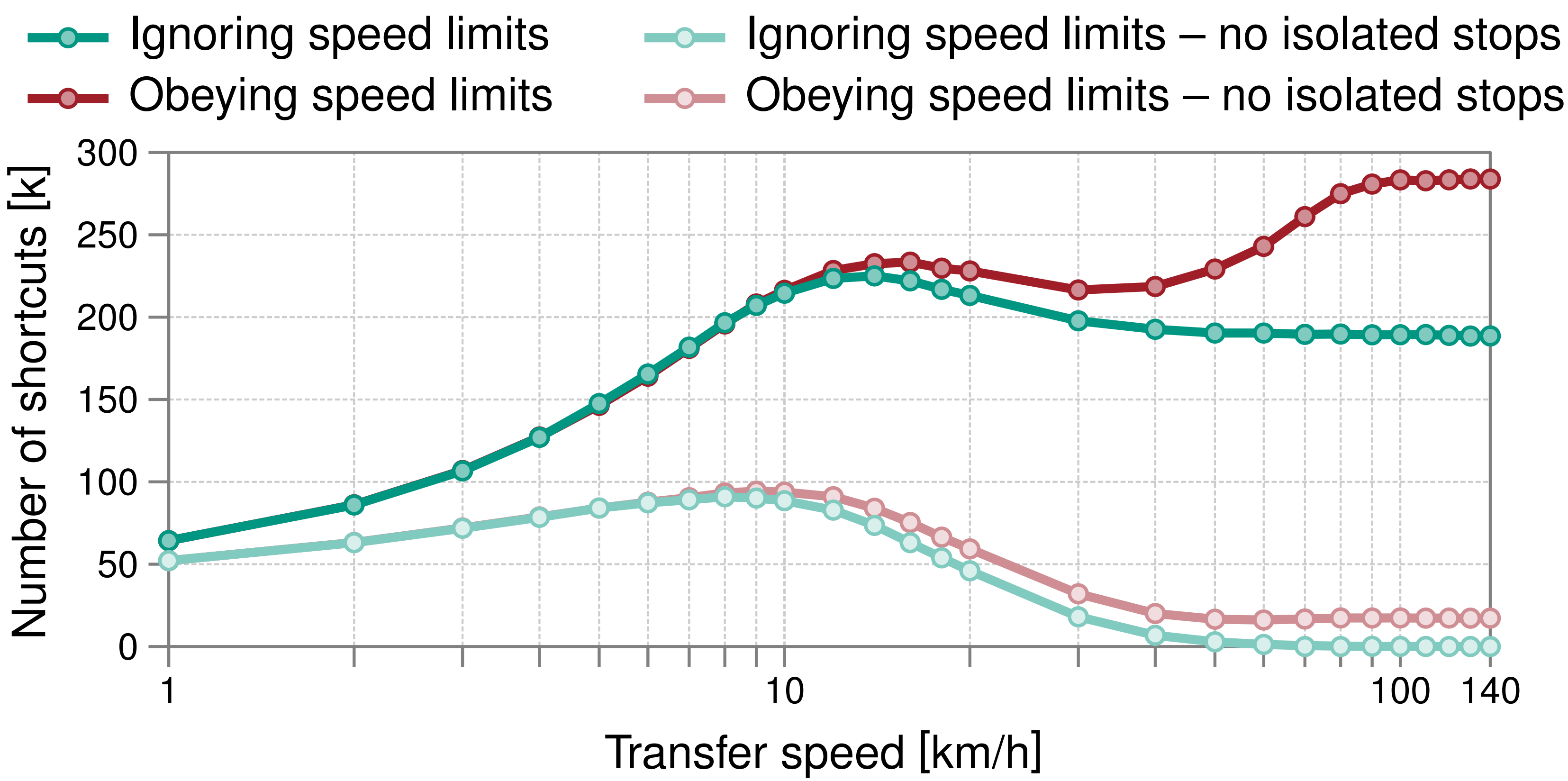
Experimental Evaluation – Preprocessing

Karlsruhe Institute of Technology

Setup:

- Switzerland with different speeds for the transfer graph
- Isolated stops / imperfect data was filtered out

Result:



 Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf

Institute of Theoretical Informatics
Algorithmics Group

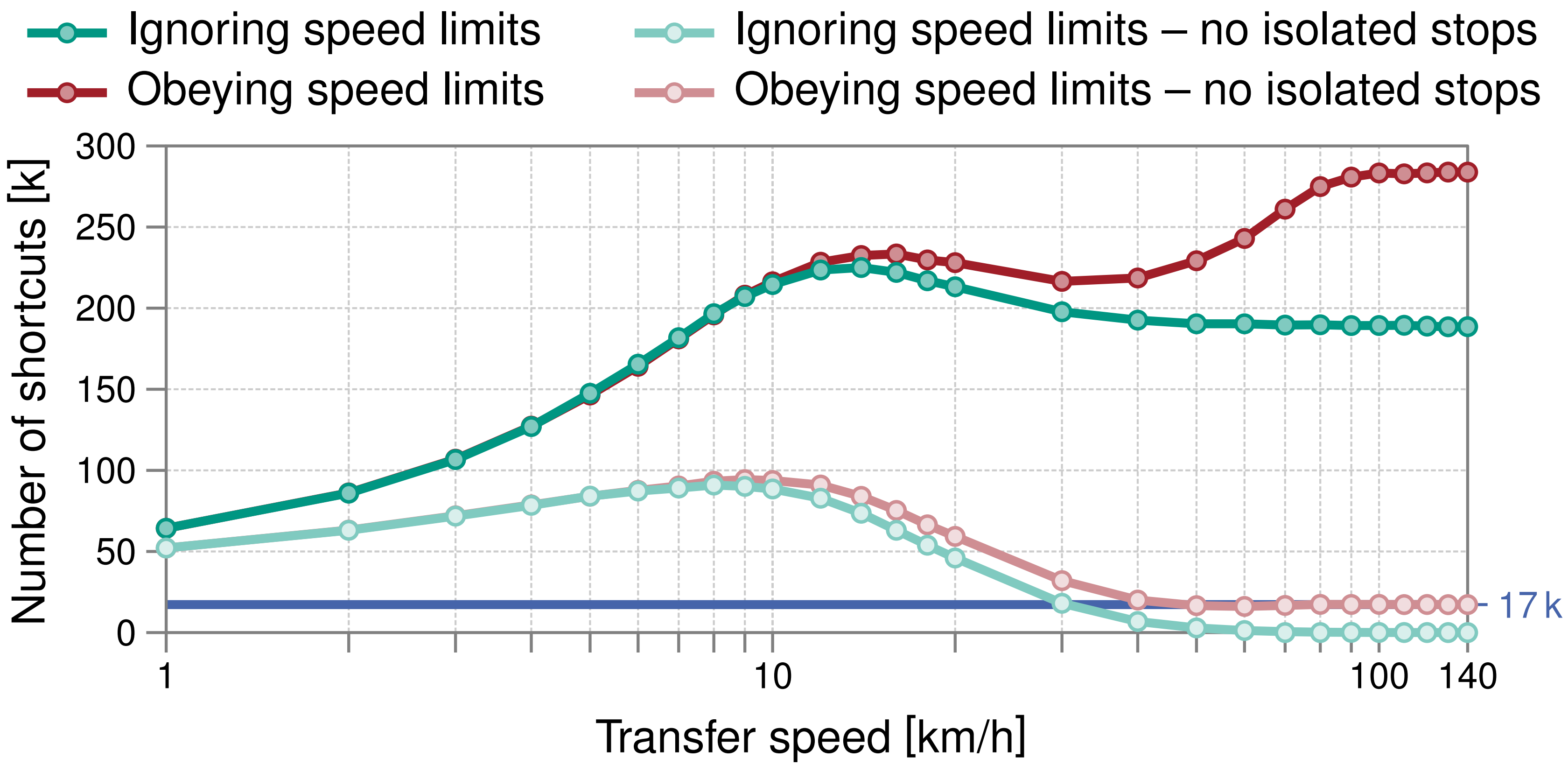
Experimental Evaluation – Preprocessing

Karlsruhe Institute of Technology

Setup:

- Switzerland with different speeds for the transfer graph
- Isolated stops / imperfect data was filtered out

Result:



Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf

Institute of Theoretical Informatics
Algorithmics Group

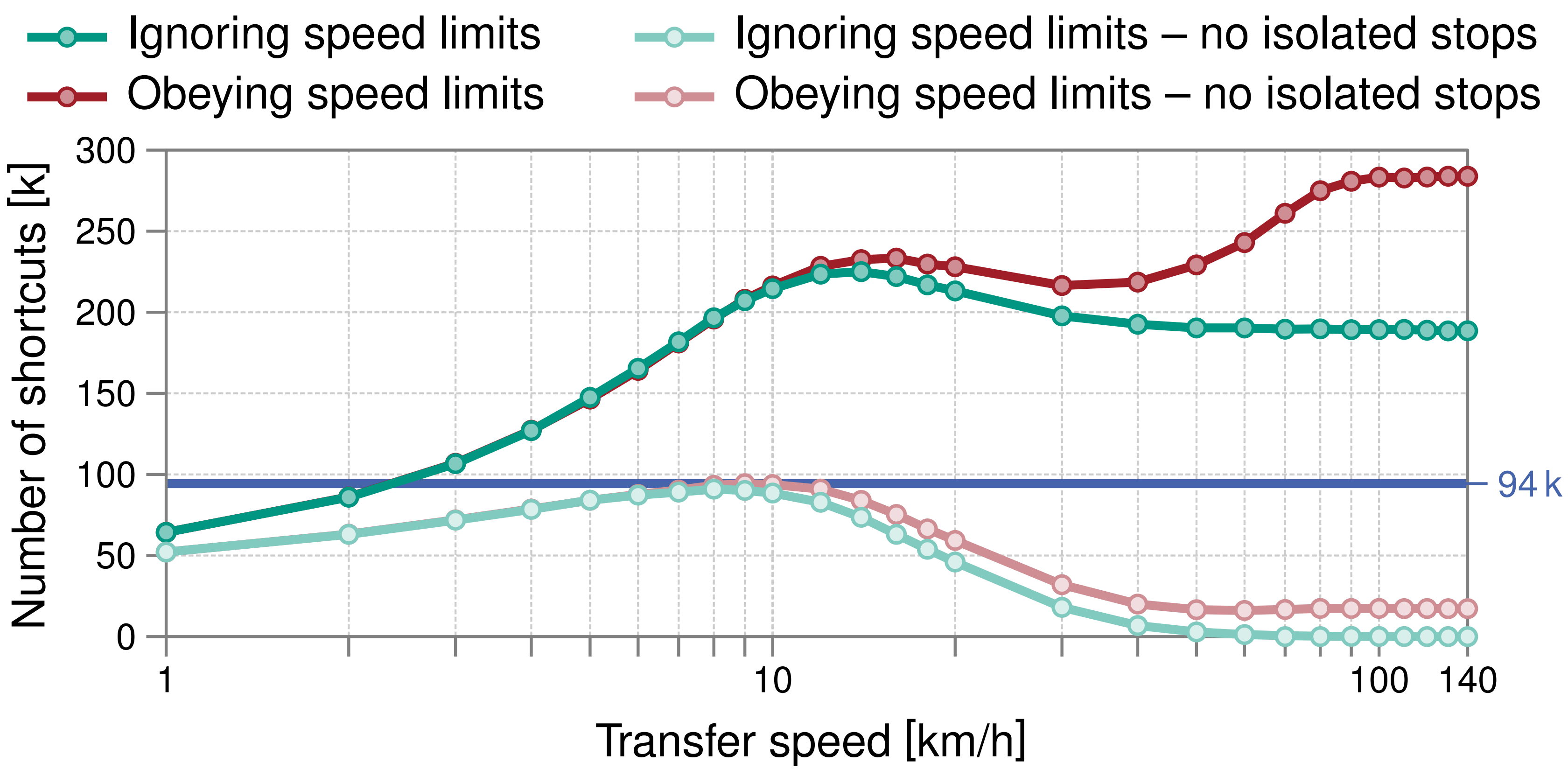
Experimental Evaluation – Preprocessing

Karlsruhe Institute of Technology

Setup:

- Switzerland with different speeds for the transfer graph
- Isolated stops / imperfect data was filtered out

Result:



Experimental Evaluation – ULTRA-CSA

Karlsruhe Institute of Technology

Setup:

- CSA-based queries, optimizing only arrival time
- MCSA interleaves CSA with Dijkstra's algorithm
- Query type for CSA*: stop-to-stop
- Query type for MCSA, ULTRA-CSA: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]		
		Connections	Edges	Init.	Scan	Total
Switzerland (4.5 km/h)	CSA*	124.7	1 294	0.1	6.0	6.2
	MCSA	85.3	244	9.9	9.0	18.8
	ULTRA-CSA	84.7	80	1.3	4.2	5.6
Germany (4.5 km/h)	CSA*	2 564.0	6 269	1.7	145.8	147.5
	MCSA	1 527.8	3 182	148.2	185.9	334.1
	ULTRA-CSA	1 523.4	933	23.3	119.7	143.0

Institute of Theoretical Informatics
Algorithmics Group

Experimental Evaluation – ULTRA-CSA

Karlsruhe Institute of Technology

Setup:

- CSA-based queries, optimizing only arrival time
- MCSA interleaves CSA with Dijkstra's algorithm
- Query type for CSA*: stop-to-stop
- Query type for MCSA, ULTRA-CSA: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]		
		Connections	Edges	Init.	Scan	Total
Switzerland (4.5 km/h)	CSA*	124.7	1 294	0.1	6.0	6.2
	MCSA	85.3	244	9.9	9.0	18.8
	ULTRA-CSA	84.7	80	1.3	4.2	5.6
Germany (4.5 km/h)	CSA*	2 564.0	6 269	1.7	145.8	147.5
	MCSA	1 527.8	3 182	148.2	185.9	334.1
	ULTRA-CSA	1 523.4	933	23.3	119.7	143.0

Experimental Evaluation – ULTRA-CSA

Setup:

- CSA-based queries, optimizing only arrival time
- MCSA interleaves CSA with Dijkstra's algorithm
- Query type for CSA*: stop-to-stop
- Query type for MCSA, ULTRA-CSA: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]		
		Connections	Edges	Init.	Scan	Total
Switzerland (4.5 km/h)	CSA*	124.7	1 294	0.1	6.0	6.2
	MCSA	85.3	244	9.9	9.0	18.8
	ULTRA-CSA	84.7	80	1.3	4.2	5.6
Germany (4.5 km/h)	CSA*	2 564.0	6 269	1.7	145.8	147.5
	MCSA	1 527.8	3 182	148.2	185.9	334.1
	ULTRA-CSA	1 523.4	933	23.3	119.7	143.0

Experimental Evaluation – ULTRA-CSA

Setup:

- CSA-based queries, optimizing only arrival time
- MCSA interleaves CSA with Dijkstra's algorithm
- Query type for CSA*: stop-to-stop
- Query type for MCSA, ULTRA-CSA: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]		
		Connections	Edges	Init.	Scan	Total
Switzerland (4.5 km/h)	CSA*	124.7	1 294	0.1	6.0	6.2
	MCSA	85.3	244	9.9	9.0	18.8
	ULTRA-CSA	84.7	80	1.3	4.2	5.6
Germany (4.5 km/h)	CSA*	2 564.0	6 269	1.7	145.8	147.5
	MCSA	1 527.8	3 182	148.2	185.9	334.1
	ULTRA-CSA	1 523.4	933	23.3	119.7	143.0

Experimental Evaluation – ULTRA-RAPTOR

Karlsruhe Institute of Technology

Setup:

- RAPTOR-based queries, optimizing arrival time and number of trips
- MR-∞ is MCR with unlimited walking
- Query type for RAPTOR*: stop-to-stop
- Query type for MR-∞, ULTRA-RAPTOR: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]				
		Routes	Edges	Init.	Collect	Scan	Relax	Total
Switzerland (4.5 km/h)	RAPTOR*	27.2	3 527	0.0	3.7	6.4	7.8	18.4
	MR-∞	34.9	769	11.6	5.9	8.2	12.3	39.3
	ULTRA-RAPTOR	37.7	148	1.6	4.9	7.9	1.9	16.7
Germany (4.5 km/h)	RAPTOR*	480.4	25 798	0.0	166.9	178.0	85.1	436.5
	MR-∞	555.8	12 571	191.1	250.7	202.2	272.2	944.1
	ULTRA-RAPTOR	610.6	2 224	26.8	204.5	202.9	37.0	477.8

Experimental Evaluation – ULTRA-RAPTOR

Setup:

- RAPTOR-based queries, optimizing arrival time and number of trips
- MR-∞ is MCR with unlimited walking
- Query type for RAPTOR*: stop-to-stop
- Query type for MR-∞, ULTRA-RAPTOR: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]				
		Routes	Edges	Init.	Collect	Scan	Relax	Total
Switzerland (4.5 km/h)	RAPTOR*	27.2	3 527	0.0	3.7	6.4	7.8	18.4
	MR-∞	34.9	769	11.6	5.9	8.2	12.3	39.3
	ULTRA-RAPTOR	37.7	148	1.6	4.9	7.9	1.9	16.7
Germany (4.5 km/h)	RAPTOR*	480.4	25 798	0.0	166.9	178.0	85.1	436.5
	MR-∞	555.8	12 571	191.1	250.7	202.2	272.2	944.1
	ULTRA-RAPTOR	610.6	2 224	26.8	204.5	202.9	37.0	477.8

Experimental Evaluation – ULTRA-RAPTOR

Setup:

- RAPTOR-based queries, optimizing arrival time and number of trips
- MR-∞ is MCR with unlimited walking
- Query type for RAPTOR*: stop-to-stop
- Query type for MR-∞, ULTRA-RAPTOR: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]				
		Routes	Edges	Init.	Collect	Scan	Relax	Total
Switzerland (4.5 km/h)	RAPTOR*	27.2	3 527	0.0	3.7	6.4	7.8	18.4
	MR-∞	34.9	769	11.6	5.9	8.2	12.3	39.3
	ULTRA-RAPTOR	37.7	148	1.6	4.9	7.9	1.9	16.7
Germany (4.5 km/h)	RAPTOR*	480.4	25 798	0.0	166.9	178.0	85.1	436.5
	MR-∞	555.8	12 571	191.1	250.7	202.2	272.2	944.1
	ULTRA-RAPTOR	610.6	2 224	26.8	204.5	202.9	37.0	477.8

Experimental Evaluation – ULTRA-RAPTOR

Karlsruhe Institute of Technology

Setup:

- RAPTOR-based queries, optimizing arrival time and number of trips
- MR-∞ is MCR with unlimited walking
- Query type for RAPTOR*: stop-to-stop
- Query type for MR-∞, ULTRA-RAPTOR: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]				
		Routes	Edges	Init.	Collect	Scan	Relax	Total
Switzerland (4.5 km/h)	RAPTOR*	27.2	3 527	0.0	3.7	6.4	7.8	18.4
	MR-∞	34.9	769	11.6	5.9	8.2	12.3	39.3
	ULTRA-RAPTOR	37.7	148	1.6	4.9	7.9	1.9	16.7
Germany (4.5 km/h)	RAPTOR*	480.4	25 798	0.0	166.9	178.0	85.1	436.5
	MR-∞	555.8	12 571	191.1	250.7	202.2	272.2	944.1
	ULTRA-RAPTOR	610.6	2 224	26.8	204.5	202.9	37.0	477.8

Experimental Evaluation – ULTRA-RAPTOR

Karlsruhe Institute of Technology

Setup:

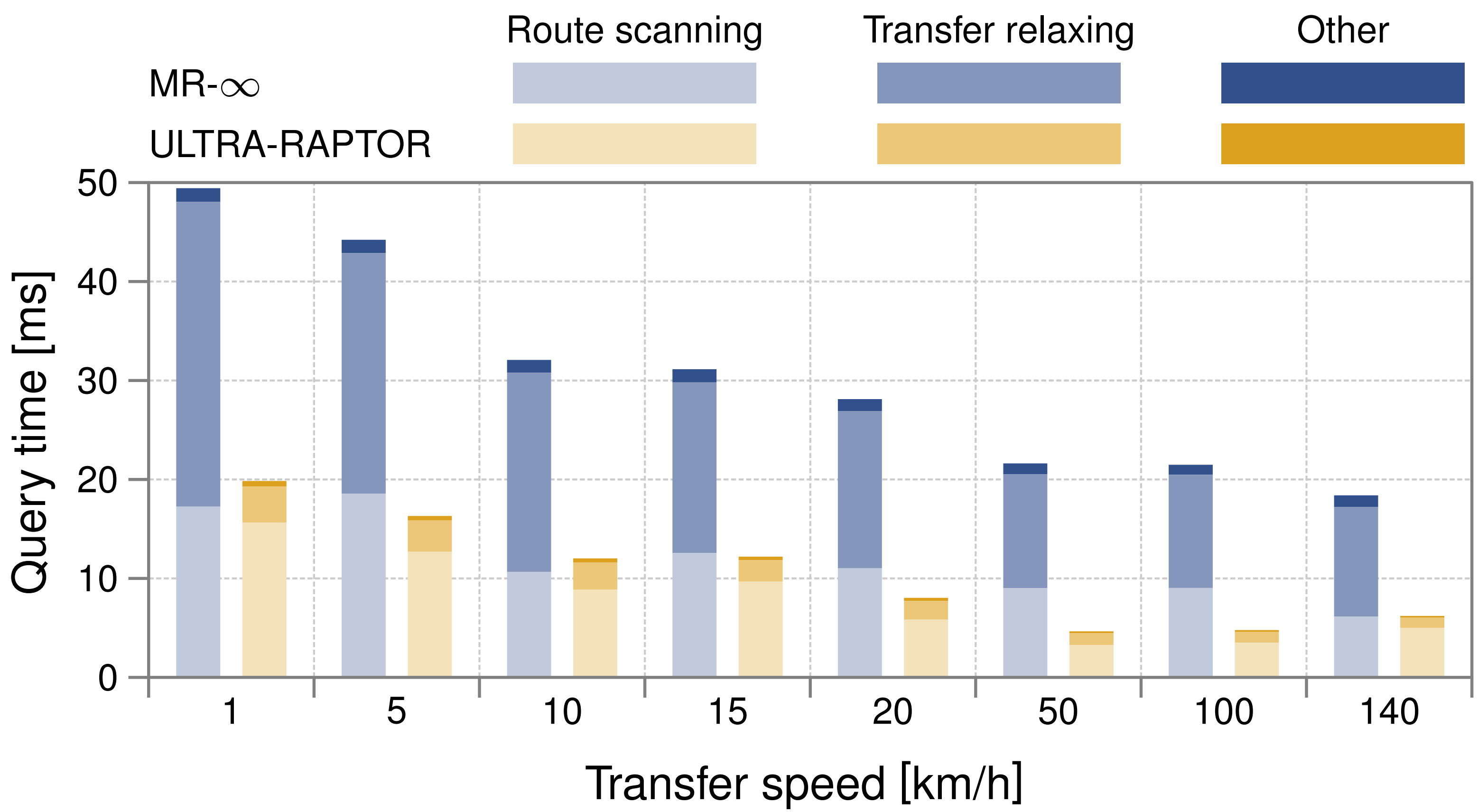
- RAPTOR-based queries, optimizing arrival time and number of trips
- MR-∞ is MCR with unlimited walking
- Query type for RAPTOR*: stop-to-stop
- Query type for MR-∞, ULTRA-RAPTOR: vertex-to-vertex

Network	Algorithm	Scans [k]		Time [ms]				
		Routes	Edges	Init.	Collect	Scan	Relax	Total
Switzerland (4.5 km/h)	RAPTOR*	27.2	3 527	0.0	3.7	6.4	7.8	18.4
	MR-∞	34.9	769	11.6	5.9	8.2	12.3	39.3
	ULTRA-RAPTOR	37.7	148	1.6	4.9	7.9	1.9	16.7
Germany (4.5 km/h)	RAPTOR*	480.4	25 798	0.0	166.9	178.0	85.1	436.5
	MR-∞	555.8	12 571	191.1	250.7	202.2	272.2	944.1
	ULTRA-RAPTOR	610.6	2 224	26.8	204.5	202.9	37.0	477.8

Experimental Evaluation – ULTRA-RAPTOR

Setup:

- RAPTOR-based queries, optimizing arrival time and number of trips



Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf

Institute of Theoretical Informatics
Algorithmics Group

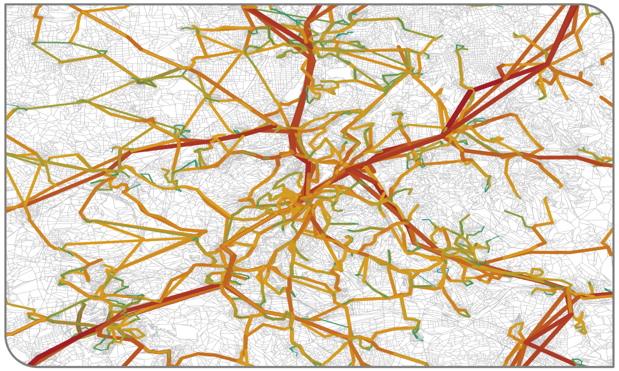
Ongoing and Future Work

Extending the ULTRA Preprocessing:

- Compute shortcuts for more criteria (price, transfer distance, ...)
- Accelerate the preprocessing phase
- Consider complicated transfer scenarios (bike sharing stations)

Utilizing the ULTRA Shortcuts:

- Multi-modal public transit traffic assignments
- Other query algorithms (Trip-Based, ...)
- One-to-many queries



Thank you for your attention

Institute of Theoretical Informatics
Algorithmics Group