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

ESA · September 11, 2019
Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf
Multi-Modal Route Planning

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

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, ...)
- But also:
  - Allow secondary transfer mode
  - Non-schedule-based
    (walking, bike, e-scooter, ...)
Problem Statement

Given:

- Public transit network (timetable)
Problem Statement

Given:
- Public transit network (timetable)
- Stops (bus stops, stations)
Problem Statement

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

Given:
- Public transit network (timetable)
- Stops (bus stops, stations)
- Routes (bus lines, train lines)
- Trips (schedule of a vehicle)
- Transfer graph (non-schedule-based)
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)
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)
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)
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
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
Related Work

Public Transit:

- Restricted transfers, only between a few stops
- Transitivity 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

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

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

First Idea:
- Enumerate all Pareto-optimal journeys
- Collect the transfers used by them
Our Approach – The Preprocessing Phase

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

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 ⇔ 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: $(\text{Timetable}, G)$
- 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: $(\text{Timetable}, \ G)$
- 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: $(\text{Timetable}, G)$
- 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: $(\text{Timetable, } G)$
- 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

### 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 $\approx 100$

<table>
<thead>
<tr>
<th>Network</th>
<th>Switzerland</th>
<th>Germany</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stops</td>
<td>25 426</td>
<td>244 055</td>
</tr>
<tr>
<td>Routes</td>
<td>13 934</td>
<td>231 089</td>
</tr>
<tr>
<td>Trips</td>
<td>369 534</td>
<td>2 387 297</td>
</tr>
<tr>
<td>Stop events</td>
<td>4 740 929</td>
<td>48 495 169</td>
</tr>
<tr>
<td>Vertices</td>
<td>604 167</td>
<td>6 872 105</td>
</tr>
<tr>
<td>Full edges</td>
<td>1 847 140</td>
<td>21 372 360</td>
</tr>
<tr>
<td>Transitive edges</td>
<td>4 687 016</td>
<td>22 645 480</td>
</tr>
</tbody>
</table>
Experimental Evaluation – Preprocessing

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)

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing time [mm:ss]</td>
<td>2:00:56</td>
<td>58:03</td>
<td>31:11</td>
<td>17:29</td>
<td>10:12</td>
</tr>
<tr>
<td>Speed-up factor</td>
<td>1</td>
<td>2.08</td>
<td>3.88</td>
<td>6.92</td>
<td>11.85</td>
</tr>
</tbody>
</table>
Experimental Evaluation – Preprocessing

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)

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing time [mm:ss]</td>
<td>2:00:56</td>
<td>58:03</td>
<td>31:11</td>
<td>17:29</td>
<td>10:12</td>
</tr>
<tr>
<td>Speed-up factor</td>
<td>1</td>
<td>2.08</td>
<td>3.88</td>
<td>6.92</td>
<td>11.85</td>
</tr>
</tbody>
</table>
Experimental Evaluation – Preprocessing

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

Result:

- Ignoring speed limits
- Obeying speed limits

![Graph showing preprocessing time against transfer speed](image)
Experimental Evaluation – Preprocessing

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

Result:

![Graph showing preprocessing time vs. transfer speed (km/h)](image)

- Ignoring speed limits
- Obeying speed limits

Preprocessing time [min]

Transfer speed [km/h]
Experimental Evaluation – Preprocessing

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

Result:

![Graph showing preprocessing time by transfer speed with two lines: one for ignoring speed limits and one for obeying speed limits.](image)

- **Preprocessing time [min]**
  - **Transfer speed [km/h]**
  - **Setup:**
    - Switzerland with different speeds for the transfer graph
    - In parallel on a machine with 16 cores
  - **Result:**
    - Ignoring speed limits
    - Obeying speed limits
  - The graph shows that preprocessing time increases as transfer speed increases, with a peak at around 18 km/h, and decreases thereafter.
Experimental Evaluation – Preprocessing

Setup:
- Switzerland with different speeds for the transfer graph

Result:

Setup:
- Switzerland with different speeds for the transfer graph

Result:

- Ignoring speed limits
- Obeying speed limits

![Graph showing the number of shortcuts vs. transfer speed]

- Number of shortcuts [k]
- Transfer speed [km/h]
- Setup: Switzerland with different speeds for the transfer graph
- Result: Ignoring speed limits vs. Obeying speed limits
Experimental Evaluation – Preprocessing

Setup:
- Switzerland with different speeds for the transfer graph

Result:

- Graph showing the number of shortcuts as a function of the transfer speed, with two lines representing 'Ignoring speed limits' and 'Obeying speed limits'.
Experimental Evaluation – Preprocessing

Setup:
- Switzerland with different speeds for the transfer graph

Result:

![Graph showing number of shortcuts vs. transfer speed]

- Ignoring speed limits
- Obeying speed limits
Experimental Evaluation – Preprocessing

Setup:
- Switzerland with different speeds for the transfer graph

Result:

![Graph showing the number of shortcuts vs. transfer speed]

- Ignoring speed limits
- Obeying speed limits

Number of shortcuts [k]

Transfer speed [km/h]

Setup:

Result:

Switzerland with different speeds for the transfer graph

Number of shortcuts [k]

Transfer speed [km/h]

284 k
Experimental Evaluation – Preprocessing

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

Result:

- Ignoring speed limits
- Obeying speed limits
- Ignoring speed limits – no isolated stops
- Obeying speed limits – no isolated stops

Graph:
- Number of shortcuts vs. transfer speed
- Setup: Switzerland with different speeds for the transfer graph
- Result: Isolated stops / imperfect data was filtered out
Experimental Evaluation – Preprocessing

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

Result:

![Graph showing the relationship between transfer speed and number of shortcuts](image)

- Ignoring speed limits
- Obeying speed limits
- Ignoring speed limits – no isolated stops
- Obeying speed limits – no isolated stops

Number of shortcuts [k]
Transfer speed [km/h]
Experimental Evaluation – Preprocessing

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

Result:
- Ignoring speed limits
- Obeying speed limits
- Ignoring speed limits – no isolated stops
- Obeying speed limits – no isolated stops

The graph shows the number of shortcuts [k] against the transfer speed [km/h]. The setup includes different speeds for the transfer graph in Switzerland and filtering out isolated stops. The result indicates a significant increase in shortcuts with increasing transfer speed for both obeying and ignoring speed limits. Notably, the setup without isolated stops shows a 94k shortcut increase at higher speeds, compared to the setup with isolated stops.
## 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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Connections</td>
<td>Edges</td>
</tr>
<tr>
<td>Switzerland</td>
<td>CSA*</td>
<td>124.7</td>
<td>1294</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td>MCSA</td>
<td>85.3</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>ULTRA-CSA</td>
<td>84.7</td>
<td>80</td>
</tr>
<tr>
<td>Germany</td>
<td>CSA*</td>
<td>2564.0</td>
<td>6269</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td>MCSA</td>
<td>1527.8</td>
<td>3182</td>
</tr>
<tr>
<td></td>
<td>ULTRA-CSA</td>
<td>1523.4</td>
<td>933</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Connections</td>
<td>Edges</td>
</tr>
<tr>
<td>Switzerland</td>
<td>CSA*</td>
<td>124.7</td>
<td>1 294</td>
</tr>
<tr>
<td></td>
<td>MCSA</td>
<td>85.3</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>ULTRA-CSA</td>
<td>84.7</td>
<td>80</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td></td>
<td>2 564.0</td>
<td>6 269</td>
</tr>
<tr>
<td></td>
<td>CSA*</td>
<td>1 527.8</td>
<td>3 182</td>
</tr>
<tr>
<td></td>
<td>MCSA</td>
<td>1 523.4</td>
<td>933</td>
</tr>
<tr>
<td></td>
<td>ULTRA-CSA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Connections</td>
<td>Edges</td>
</tr>
<tr>
<td>Switzerland</td>
<td>CSA*</td>
<td>124.7</td>
<td>1 294</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td>MCSA</td>
<td>85.3</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>ULTRA-CSA</td>
<td>84.7</td>
<td>80</td>
</tr>
<tr>
<td>Germany</td>
<td>CSA*</td>
<td>2 564.0</td>
<td>6 269</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td>MCSA</td>
<td>1 527.8</td>
<td>3 182</td>
</tr>
<tr>
<td></td>
<td>ULTRA-CSA</td>
<td>1 523.4</td>
<td>933</td>
</tr>
</tbody>
</table>
## 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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Connections</td>
<td>Edges</td>
</tr>
<tr>
<td>Switzerland</td>
<td>CSA*</td>
<td>124.7</td>
<td>1294</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td>MCSA</td>
<td>85.3</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>ULTRA-CSA</td>
<td>84.7</td>
<td>80</td>
</tr>
<tr>
<td>Germany</td>
<td>CSA*</td>
<td>2564.0</td>
<td>6269</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td>MCSA</td>
<td>1527.8</td>
<td>3182</td>
</tr>
<tr>
<td></td>
<td>ULTRA-CSA</td>
<td>1523.4</td>
<td>933</td>
</tr>
</tbody>
</table>
## 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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Routes Edges Init. Collect Scan Relax Total</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Switzerland (4.5 km/h)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>RAPTOR</strong>*</td>
<td>27.2</td>
<td>3 527</td>
</tr>
<tr>
<td></td>
<td><strong>MR-∞</strong></td>
<td>34.9</td>
<td>769</td>
</tr>
<tr>
<td></td>
<td><strong>ULTRA-RAPTOR</strong></td>
<td>37.7</td>
<td>148</td>
</tr>
<tr>
<td><strong>Germany (4.5 km/h)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>RAPTOR</strong>*</td>
<td>480.4</td>
<td>25 798</td>
</tr>
<tr>
<td></td>
<td><strong>MR-∞</strong></td>
<td>555.8</td>
<td>12 571</td>
</tr>
<tr>
<td></td>
<td><strong>ULTRA-RAPTOR</strong></td>
<td>610.6</td>
<td>2 224</td>
</tr>
</tbody>
</table>
## 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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Routes</th>
<th>Edges</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Init.</td>
<td>Collect</td>
</tr>
<tr>
<td>Switzerland</td>
<td>RAPTOR*</td>
<td>27.2</td>
<td>3527</td>
<td>0.0</td>
<td>3.7</td>
</tr>
<tr>
<td></td>
<td>MR-∞</td>
<td>34.9</td>
<td>769</td>
<td>11.6</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>37.7</td>
<td>148</td>
<td>1.6</td>
<td>4.9</td>
</tr>
<tr>
<td>Germany</td>
<td>RAPTOR*</td>
<td>480.4</td>
<td>25798</td>
<td>0.0</td>
<td>166.9</td>
</tr>
<tr>
<td></td>
<td>MR-∞</td>
<td>555.8</td>
<td>12571</td>
<td>191.1</td>
<td>250.7</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>610.6</td>
<td>2224</td>
<td>26.8</td>
<td>204.5</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Routes</td>
<td>Edges</td>
</tr>
<tr>
<td>Switzerland (4.5 km/h)</td>
<td>RAPTOR*</td>
<td>27.2</td>
<td>3 527</td>
</tr>
<tr>
<td></td>
<td>MR-∞</td>
<td>34.9</td>
<td>769</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>37.7</td>
<td>148</td>
</tr>
<tr>
<td>Germany (4.5 km/h)</td>
<td>RAPTOR*</td>
<td>480.4</td>
<td>25 798</td>
</tr>
<tr>
<td></td>
<td>MR-∞</td>
<td>555.8</td>
<td>12 571</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>610.6</td>
<td>2 224</td>
</tr>
</tbody>
</table>
## 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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Routes</td>
<td>Edges</td>
</tr>
<tr>
<td>Switzerland</td>
<td>RAPTOR*</td>
<td>27.2</td>
<td>3527</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td>MR-∞</td>
<td>34.9</td>
<td>769</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>37.7</td>
<td>148</td>
</tr>
<tr>
<td>Germany</td>
<td>RAPTOR*</td>
<td>480.4</td>
<td>25798</td>
</tr>
<tr>
<td>(4.5 km/h)</td>
<td>MR-∞</td>
<td>555.8</td>
<td>12571</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>610.6</td>
<td>2224</td>
</tr>
</tbody>
</table>
Experimental Evaluation – ULTRA-RAPTOR

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

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Routes</td>
<td>Edges</td>
</tr>
<tr>
<td>Switzerland (4.5 km/h)</td>
<td>RAPTOR*</td>
<td>27.2</td>
<td>3 527</td>
</tr>
<tr>
<td></td>
<td>MR-$\infty$</td>
<td>34.9</td>
<td>769</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>37.7</td>
<td>148</td>
</tr>
<tr>
<td>Germany (4.5 km/h)</td>
<td>RAPTOR*</td>
<td>480.4</td>
<td>25 798</td>
</tr>
<tr>
<td></td>
<td>MR-$\infty$</td>
<td>555.8</td>
<td>12 571</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>610.6</td>
<td>2 224</td>
</tr>
</tbody>
</table>
Experimental Evaluation – ULTRA-RAPTOR

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

![Bar chart showing query times for different transfer speeds and setup configurations: MR-∞, Route scanning, Transfer relaxing, Other.]
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