Integrating ULTRA and Trip-Based Routing

ATMOS · September 7, 2020
Jonas Sauer, Dorothea Wagner, and Tobias Zündorf
Multi-Modal Journey Planning

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

Institute of Theoretical Informatics
Algorithmics Group
**Multi-Modal Journey Planning**

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

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

Integrating ULTRA and Trip-Based Routing
Jonas Sauer, Dorothea Wagner, and Tobias Zündorf
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)
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)
- 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)
- Source \( s \), target \( t \), and a departure time

**Objective:**
- Find all Pareto-optimal journeys w.r.t. arrival time and number of trips
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)
- Source $s$, target $t$, and a departure time

Objective:
- Find all Pareto-optimal journeys w.r.t. \textit{arrival time} and \textit{number of trips}
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)
- Source $s$, target $t$, and a departure time

Objective:
- Find all Pareto-optimal journeys w.r.t. arrival time and number of trips
Related Work

**ULTRA:** (multimodal journey planning)
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\leadsto$ Shortcuts between stops
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\rightsquigarrow$ Shortcuts between stops
Related Work

**ULTRA**: (multimodal journey planning)
- Complex transfer graph $\rightsquigarrow$ Shortcuts between stops
- Add transfers from source / to target
Related Work

**ULTRA**: (multimodal journey planning)
- Complex transfer graph $\rightsquigarrow$ Shortcuts between stops
- Add transfers from source / to target
Related Work

**ULTRA: (multimodal journey planning)**
- Complex transfer graph \(\rightsquigarrow\) Shortcuts between stops
- Add transfers from source / to target
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\leadsto$ Shortcuts between stops
- Add transfers from source / to target
- Use public transit algorithm
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\rightsquigarrow$ Shortcuts between stops
- Add transfers from source / to target
- Use public transit algorithm

**Trip-Based Routing:** (public transit journey planning)
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\leadsto$ Shortcuts between stops
- Add transfers from source / to target
- Use public transit algorithm

**Trip-Based Routing:** (public transit journey planning)
- Operates on the trips of the network
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\leadsto$ Shortcuts between stops
- Add transfers from source / to target
- Use public transit algorithm

**Trip-Based Routing:** (public transit journey planning)
- Operates on the trips of the network
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\leadsto$ Shortcuts between stops
- Add transfers from source / to target
- Use public transit algorithm

**Trip-Based Routing:** (public transit journey planning)
- Operates on the trips of the network
- Transitive transfer graph $\leadsto$ Shortcuts between stop events
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\sim$ Shortcuts between stops
- Add transfers from source / to target
- Use public transit algorithm

**Trip-Based Routing:** (public transit journey planning)
- Operates on the trips of the network
- Transitive transfer graph $\sim$ Shortcuts between stop events
Related Work

**ULTRA:** (multimodal journey planning)
- Complex transfer graph $\leadsto$ Shortcuts between stops
- Add transfers from source / to target
- Use public transit algorithm

**Trip-Based Routing:** (public transit journey planning)
- Operates on the trips of the network
- Transitive transfer graph $\leadsto$ Shortcuts between stop events
- BFS-like search on the resulting graph
Related Work

**ULTRA**: (multimodal journey planning)
- Complex transfer graph $\rightarrow$ Shortcuts between stops
- Add transfers from source / to target
- Use public transit algorithm

**Trip-Based Routing**: (public transit journey planning)
- Operates on the trips of the network
- Transitive transfer graph $\rightarrow$ Shortcuts between stop events
- BFS-like search on the resulting graph
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses time-independent shortcuts
  - Trip-Based uses time-dependent shortcuts
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:
- Precomputed shortcuts represent transfers
  - ULTRA uses *time-independent* shortcuts
  - Trip-Based uses *time-dependent* shortcuts
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses \textit{time-independent} shortcuts
  - Trip-Based uses \textit{time-dependent} shortcuts
- Preprocessing enumerates journeys with up to two trips
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses \textit{time-independent} shortcuts
  - Trip-Based uses \textit{time-dependent} shortcuts
- Preprocessing enumerates journeys with up to two trips
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses **time-independent** shortcuts
  - Trip-Based uses **time-dependent** shortcuts
- Preprocessing enumerates journeys with up to two trips
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses **time-independent** shortcuts
  - Trip-Based uses **time-dependent** shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first stop

![Diagram showing the integration of ULTRA and Trip-Based Preprocessing](image-url)
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses time-independent shortcuts
  - Trip-Based uses time-dependent shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first stop
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses \textit{time-independent} shortcuts
  - Trip-Based uses \textit{time-dependent} shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first \textit{stop}
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses time-independent shortcuts
  - Trip-Based uses time-dependent shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first stop
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:
- Precomputed shortcuts represent transfers
  - ULTRA uses **time-independent** shortcuts
  - Trip-Based uses **time-dependent** shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first **stop**
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses **time-independent** shortcuts
  - Trip-Based uses **time-dependent** shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first stop
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses time-independent shortcuts
  - Trip-Based uses time-dependent shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first stop
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses time-independent shortcuts
  - Trip-Based uses time-dependent shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first stop
  - Trip-Based compares journeys with the same first trip
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses **time-independent** shortcuts
  - Trip-Based uses **time-dependent** shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first **stop**
  - Trip-Based compares journeys with the same first **trip**
Integrating ULTRA and Trip-Based Preprocessing

Similarities and Differences:

- Precomputed shortcuts represent transfers
  - ULTRA uses *time-independent* shortcuts
  - Trip-Based uses *time-dependent* shortcuts
- Preprocessing enumerates journeys with up to two trips
  - ULTRA compares journeys with the same first stop
  - Trip-Based compares journeys with the same first trip
- Shortcut is required if no better alternative exists
  - For a suitable definition of “better”
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

$s$, 8:00

$o t$
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:
1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

\[ s, 8:00 \]
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:
1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. Trip-based query?
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:
1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
2. Trip-based query?
Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. Trip-based query?
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. Trip-based query?
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:
1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
2. Trip-based query?
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:
1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:
1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target
2. RAPTOR collect routes
   - Find the first trip for every stop and route
Integrated ULTRA-Trip-Based Query

Query Algorithm Outline:

1. Bucket-CH query
   - Find arrival time at all stops
   - Find transfer time from all stops to the target

2. RAPTOR collect routes
   - Find the first trip for every stop and route

3. Trip-Based trip scanning
   - Find all Pareto optimal journeys
### Experimental Evaluation

**Instances:**
- London, Stuttgart, Switzerland, and Germany
- Timetables comprising two days from TfL, GTFS-CH, and DB

<table>
<thead>
<tr>
<th>Network</th>
<th>Stops</th>
<th>Routes</th>
<th>Trips</th>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>20 595</td>
<td>2 107</td>
<td>125 436</td>
<td>183 k</td>
<td>579 k</td>
</tr>
<tr>
<td>Stuttgart</td>
<td>13 583</td>
<td>12 350</td>
<td>91 298</td>
<td>1 166 k</td>
<td>3 680 k</td>
</tr>
<tr>
<td>Switzerland</td>
<td>25 426</td>
<td>13 934</td>
<td>369 006</td>
<td>604 k</td>
<td>1 847 k</td>
</tr>
<tr>
<td>Germany</td>
<td>244 055</td>
<td>231 089</td>
<td>2 387 292</td>
<td>6 872 k</td>
<td>21 372 k</td>
</tr>
</tbody>
</table>
Experimental Evaluation – Preprocessing

Comparing Sequential and Integrated Preprocessing:

- Using ULTRA shortcuts as input for Trip-Based Routing is a bit faster
- The Integrated approach yields significantly fewer shortcuts

<table>
<thead>
<tr>
<th></th>
<th>Stuttgart</th>
<th>London</th>
<th>Switzerland</th>
<th>Germany</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sequential)</td>
<td>4:40</td>
<td>19:15</td>
<td>9:16</td>
<td>7:54:13</td>
</tr>
<tr>
<td>Time (integrated)</td>
<td>5:11</td>
<td>22:24</td>
<td>10:04</td>
<td>9:16:15</td>
</tr>
<tr>
<td>Shortcuts (sequential)</td>
<td>25 865 892</td>
<td>58 301 120</td>
<td>58 807 528</td>
<td>1 072 750 574</td>
</tr>
<tr>
<td>Shortcuts (integrated)</td>
<td>3 900 258</td>
<td>19 856 062</td>
<td>11 646 572</td>
<td>121 676 520</td>
</tr>
</tbody>
</table>
Comparing Sequential and Integrated Preprocessing:

- Using ULTRA shortcuts as input for Trip-Based Routing is a bit faster
- The Integrated approach yields significantly fewer shortcuts

<table>
<thead>
<tr>
<th></th>
<th>Stuttgart</th>
<th>London</th>
<th>Switzerland</th>
<th>Germany</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sequential)</td>
<td>4:40</td>
<td>19:15</td>
<td>9:16</td>
<td>7:54:13</td>
</tr>
<tr>
<td>Time (integrated)</td>
<td>5:11</td>
<td>22:24</td>
<td>10:04</td>
<td>9:16:15</td>
</tr>
<tr>
<td>Increase</td>
<td>1.11</td>
<td>1.16</td>
<td>1.09</td>
<td>1.17</td>
</tr>
<tr>
<td>Shortcuts (sequential)</td>
<td>25 865 892</td>
<td>58 301 120</td>
<td>58 807 528</td>
<td>1 072 750 574</td>
</tr>
<tr>
<td>Shortcuts (integrated)</td>
<td>3 900 258</td>
<td>19 856 062</td>
<td>11 646 572</td>
<td>121 676 520</td>
</tr>
</tbody>
</table>
## Comparing Sequential and Integrated Preprocessing:

- Using ULTRA shortcuts as input for Trip-Based Routing is a bit faster
- The Integrated approach yields significantly fewer shortcuts

<table>
<thead>
<tr>
<th></th>
<th>Stuttgart</th>
<th>London</th>
<th>Switzerland</th>
<th>Germany</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sequential)</td>
<td>4:40</td>
<td>19:15</td>
<td>9:16</td>
<td>7:54:13</td>
</tr>
<tr>
<td>Time (integrated)</td>
<td>5:11</td>
<td>22:24</td>
<td>10:04</td>
<td>9:16:15</td>
</tr>
<tr>
<td>Increase</td>
<td>1.11</td>
<td>1.16</td>
<td>1.09</td>
<td>1.17</td>
</tr>
<tr>
<td>Shortcuts (sequential)</td>
<td>25 865 892</td>
<td>58 301 120</td>
<td>58 807 528</td>
<td>1 072 750 574</td>
</tr>
<tr>
<td>Shortcuts (integrated)</td>
<td>3 900 258</td>
<td>19 856 062</td>
<td>11 646 572</td>
<td>121 676 520</td>
</tr>
<tr>
<td>Reduction</td>
<td>6.63</td>
<td>2.94</td>
<td>5.05</td>
<td>8.82</td>
</tr>
</tbody>
</table>
**Experimental Evaluation – Query**

**Query Overview for London and Germany:**

- **ULTRA-TB (int.)** is the fastest algorithm (even faster than TB on a transitive graph)

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Full graph</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Trips</td>
<td>Shortcuts</td>
<td>B-CH</td>
</tr>
<tr>
<td>London</td>
<td>Trip-Based*</td>
<td>○</td>
<td>22.75</td>
<td>1 376.26</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>34.09</td>
<td>1 545.15</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>24.69</td>
<td>450.50</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
<td>0.93</td>
</tr>
<tr>
<td>Germany</td>
<td>Trip-Based*</td>
<td>○</td>
<td>337.49</td>
<td>16 116.64</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>439.35</td>
<td>38 092.34</td>
<td>25.34</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>204.23</td>
<td>3 149.87</td>
<td>26.12</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
<td>25.68</td>
</tr>
</tbody>
</table>
## Experimental Evaluation – Query

### Query Overview for London and Germany:
- **ULTRA-TB (int.)** is the fastest algorithm *(even faster than TB on a transitive graph)*

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Full graph</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Trips</td>
<td>Shortcuts</td>
</tr>
<tr>
<td>London</td>
<td>Trip-Based*</td>
<td>o</td>
<td>22.75</td>
<td>1376.26</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>34.09</td>
<td>1545.15</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>24.69</td>
<td>450.50</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Germany</td>
<td>Trip-Based*</td>
<td>o</td>
<td>337.49</td>
<td>16116.64</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>439.35</td>
<td>38092.34</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>204.23</td>
<td>3149.87</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
## Experimental Evaluation – Query

### Query Overview for London and Germany:
- **ULTRA-TB (int.)** is the fastest algorithm *(even faster than TB on a transitive graph)*

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Full graph</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Trips</td>
<td>Shortcuts</td>
<td>B-CH</td>
</tr>
<tr>
<td>London</td>
<td>Trip-Based*</td>
<td>○</td>
<td>22.75</td>
<td>1 376.26</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>34.09</td>
<td>1 545.15</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>24.69</td>
<td>450.50</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
<td>0.93</td>
</tr>
<tr>
<td>Germany</td>
<td>Trip-Based*</td>
<td>○</td>
<td>337.49</td>
<td>16 116.64</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>439.35</td>
<td>38 092.34</td>
<td>25.34</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>204.23</td>
<td>3 149.87</td>
<td>26.12</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
<td>25.68</td>
</tr>
</tbody>
</table>
## Experimental Evaluation – Query

### Query Overview for London and Germany:

- **ULTRA-TB (int.)** is the fastest algorithm (even faster than TB on a transitive graph)

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Full graph</th>
<th>Scans [k]</th>
<th>Time [ms]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Trips</td>
<td>Shortcuts</td>
<td>B-CH</td>
</tr>
<tr>
<td>London</td>
<td>Trip-Based*</td>
<td>o</td>
<td>22.75</td>
<td>1376.26</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>34.09</td>
<td>1545.15</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>24.69</td>
<td>450.50</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>-</td>
<td>-</td>
<td>0.93</td>
</tr>
<tr>
<td>Germany</td>
<td>Trip-Based*</td>
<td>o</td>
<td>337.49</td>
<td>16116.64</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>439.35</td>
<td>38092.34</td>
<td>25.34</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>204.23</td>
<td>3149.87</td>
<td>26.12</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
<td>25.68</td>
</tr>
</tbody>
</table>
Experimental Evaluation – Query

Query Overview for London and Germany:
- ULTRA-TB (int.) is the fastest algorithm (even faster than TB on a transitive graph)

<table>
<thead>
<tr>
<th>Network</th>
<th>Algorithm</th>
<th>Full graph</th>
<th>Scans [k]</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Trips</td>
<td>Shortcuts</td>
<td>B-CH</td>
<td>R Collect</td>
<td>TB Scan</td>
</tr>
<tr>
<td>London</td>
<td>Trip-Based*</td>
<td>○</td>
<td>22.75</td>
<td>1376.26</td>
<td>0.01</td>
<td>0.05</td>
<td>6.10</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>34.09</td>
<td>1545.15</td>
<td>0.91</td>
<td>0.80</td>
<td>7.47</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>24.69</td>
<td>450.50</td>
<td>0.90</td>
<td>0.70</td>
<td>4.05</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
<td>0.93</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Germany</td>
<td>Trip-Based*</td>
<td>○</td>
<td>337.49</td>
<td>16116.64</td>
<td>0.01</td>
<td>0.05</td>
<td>116.14</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (seq.)</td>
<td>●</td>
<td>439.35</td>
<td>38092.34</td>
<td>25.34</td>
<td>18.96</td>
<td>151.35</td>
</tr>
<tr>
<td></td>
<td>ULTRA-TB (int.)</td>
<td>●</td>
<td>204.23</td>
<td>3149.87</td>
<td>26.12</td>
<td>19.13</td>
<td>46.38</td>
</tr>
<tr>
<td></td>
<td>ULTRA-RAPTOR</td>
<td>●</td>
<td>–</td>
<td>–</td>
<td>25.68</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Conclusion

Our Contribution:
- We proposed the ULTRA-Trip-Based algorithm
  - About 4 times faster than best previous algorithm
- Trip-Based shortcuts
- ULTRA journey enumeration
- Integrated query

About 4 times faster than best previous algorithm
Conclusion

Our Contribution:
- We proposed the ULTRA-Trip-Based algorithm
- About 4 times faster than best previous algorithm
- Trip-Based shortcuts
- ULTRA journey enumeration

Future Work:
- Optimize more criteria
- Handle delays