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

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

Precomputation

- Transit graph
- Walk graph
- Car graph
- Multi-modal graph
- Profile queries
- Transfer patterns
- DAG

At query time

- Query graph
- Direct connection
- Evaluation
- Optimal paths
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: (30min, 1) (40min, 0) incomparable
  - (30min, 1) better than (40min, 2)

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

Precomputation

Transit graph → Walk graph → Car graph → Multi-modal graph → Profile queries → Transfer patterns → DAG

Query graph → Evaluation → Optimal paths

At query time

Direct connection
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
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 < 20 min, 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

<table>
<thead>
<tr>
<th></th>
<th>Vitoria</th>
<th>Freiburg</th>
<th>Austin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stations</td>
<td>333</td>
<td>1,381</td>
<td>2,709</td>
</tr>
<tr>
<td>Lines</td>
<td>40</td>
<td>569</td>
<td>228</td>
</tr>
<tr>
<td>Trips</td>
<td>2,733</td>
<td>2,328</td>
<td>4,852</td>
</tr>
<tr>
<td>Nodes</td>
<td>2.8K</td>
<td>20.5K</td>
<td>27.9K</td>
</tr>
<tr>
<td>Arcs</td>
<td>11.4K</td>
<td>53.8K</td>
<td>96.9K</td>
</tr>
</tbody>
</table>
Precomputation

- Labels generated by random profile queries

<table>
<thead>
<tr>
<th></th>
<th>Vitoria</th>
<th>Freiburg</th>
<th>Austin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transit</td>
<td>155K</td>
<td>101K</td>
<td>652K</td>
</tr>
<tr>
<td>Transit + walk</td>
<td>476K</td>
<td>352K</td>
<td>2,013K</td>
</tr>
<tr>
<td>Transit + walk + car</td>
<td>4,526K</td>
<td>7,695K</td>
<td>128,593K</td>
</tr>
</tbody>
</table>

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

- Average profile query times

<table>
<thead>
<tr>
<th></th>
<th>Vitoria</th>
<th>Freiburg</th>
<th>Austin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile query time (min)</td>
<td>1.97</td>
<td>0.57</td>
<td>2,634.55</td>
</tr>
</tbody>
</table>

- 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
  - \( \sim 99\% \) quality preserved

- **Recall**
  - Fraction of relevant paths that are retrieved
  - Decreased due to TNT
  - Median pure car duration 17.6 min \( \leq \) lower than limit (20 min)

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP vs. Dijkstra</td>
<td>99.1 %</td>
<td>94.0 %</td>
</tr>
<tr>
<td>TP + TNT vs.</td>
<td>98.9 %</td>
<td>40.3 %</td>
</tr>
<tr>
<td>Dijkstra</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Number of labels and transfer patterns

- Comparison: Before and after TNT filter

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vitoria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Labels</td>
<td>430</td>
<td>104</td>
</tr>
<tr>
<td>TP</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>Freiburg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Labels</td>
<td>421</td>
<td>32</td>
</tr>
<tr>
<td>TP</td>
<td>145</td>
<td>13</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th></th>
<th>Vitoria</th>
<th>Freiburg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry stations</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Build QG (ms)</td>
<td>0.16</td>
<td>0.40</td>
</tr>
<tr>
<td>Evaluation QG (ms)</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Build path (ms)</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Total time (ms)</td>
<td><strong>11</strong></td>
<td><strong>15</strong></td>
</tr>
</tbody>
</table>

- 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?

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
Location-to-location query graph

DAG station G
GFC, GFBC, GDH

DAG station A

Transfer Pattern Algorithm
Efficient Multi-modal Route Planning with Transfer Patterns
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
### Average number of labels and profile query time

<table>
<thead>
<tr>
<th></th>
<th>Vitoria</th>
<th>Freiburg</th>
<th>Austin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile query time (min)</td>
<td>1.97</td>
<td>0.57</td>
<td>2,634.55</td>
</tr>
<tr>
<td>Number of labels</td>
<td>1.94M</td>
<td>1.44M</td>
<td>128.59M</td>
</tr>
</tbody>
</table>