Delay-Robustness of Transfer Patterns in Public Transportation Route Planning
ATMOS 2013, Sophia Antipolis

Hannah Bast, Jonas Sternisko, Sabine Storandt

Department of Computer Science
Albert-Ludwigs-University Freiburg

September 5, 2013
13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization and Systems
Transfer pattern of a route

:=

Sequence of stations where vehicle is changed
Transfer pattern of a route

\[:=\]

Sequence of stations where vehicle is changed

Example: Route A → H has transfer pattern ACGH.
Transfer Pattern Routing
Transfer Pattern Routing

Given the optimal patterns from *Paris* to *Antibes*:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A
Transfer Pattern Routing

Given the optimal patterns from *Paris* to *Antibes*:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(1) Construct query graph from patterns
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(1) Construct query graph from patterns
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(1) Construct query graph from patterns
Transfer Pattern Routing

Given the optimal patterns from *Paris* to *Antibes*:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(1) Construct query graph from patterns
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Marseille, Nice, A
- P, Aix-en-Provence, A

(2) Search on the graph
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(2) Search on the graph

Direct Connection Query:
Paris@9:00 → Avignon
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(2) Search on the graph

Direct Connection Query:
Paris@9:00 → Aix-en-Provence
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(2) Search on the graph

Direct Connection Query:
Paris@9:00 → Antibes
Transfer Pattern Routing

Given the optimal patterns from *Paris* to *Antibes*:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(2) Search on the graph

Direct Connection Query:
Paris@9:00 → Marseille
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(2) Search on the graph

Direct Connection Query: Avignon@11:40 → Antibes
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(2) Search on the graph

Direct Connection Query:
Avignon@11:40 → Antibes
Transfer Pattern Routing

Given the optimal patterns from *Paris* to *Antibes*:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(2) Search on the graph
Transfer Pattern Routing

Given the optimal patterns from Paris to Antibes:

- Paris, Antibes
- P, Avignon, A
- P, Aix-en-Provence, A
- P, Marseille, Nice, A

(2) Search on the graph

Diagram showing connections and times between Paris, Avignon, Aix-en-Provence, Antibes, Marseille, and Nice.
Transfer Pattern Routing

Public Transportation Route Planning 🚌 🎐 🚤
- Find a set of Pareto-optimal routes for a query \( A@t \rightarrow B \) given a set of timetables \( tt \)
- Different structure than Road Networks, different algorithms

Transfer Pattern Routing
- Bast et al. ESA 2010
- Given the optimal transfer patterns, search the best route among them
- Multi-criteria
- Very fast even on huge networks (e.g. Northern America)
- State-of-the-art, Google Maps
Transfer Pattern Routing

Components

Data computed from timetables $tt$
- Transfer patterns $TP_{tt}$
- Direct connection datastructure $DC_{tt}$

At query time
- Dijkstra on query graph built from patterns
- Relax arcs using direct connection queries
Transfer Patterns vs. Real-Time Updates

Time-consuming precomputation
- Computation of patterns in $O(N^2)$
- Heuristics: important stations, limits
- Still very long

Route planning applications
- Steady flow of updates during operation
- Example: Delay of trips
  - Traffic jam, strike, blocked rails, ...

No guarantee for optimal responses when trips are delayed
- Transfer patterns are computed on the original network
- Any update may introduce a new optimal pattern
Handling real-time updates with transfer patterns

Observation
- Typical updates introduce only small changes to the timetables, $tt \rightarrow tt'$
- Computing $TP_{tt'}$ is expensive, $DC_{tt'}$ not.

Main idea
- Update only the direct connection data $DC_{tt'}$
- Search on the original transfer patterns $TP_{tt}$

Expectation
- Correct responses in most cases
- What happens elsewhere?
How much does the quality of results suffer?

Experimental Setup

- Compute transfer patterns
- Apply delay scenario to dataset \((tt \rightarrow tt')\)
- Answer N queries using \(TP_{tt}, DC_{tt'}\)
- Compare results with ground truth

<table>
<thead>
<tr>
<th>Delay Scenario</th>
<th>Average delay</th>
<th>(E\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 min</td>
<td>15 min</td>
</tr>
<tr>
<td>Low</td>
<td>25%</td>
<td>-</td>
</tr>
<tr>
<td>Medium</td>
<td>-</td>
<td>25%</td>
</tr>
<tr>
<td>High</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mix Low</td>
<td>10%</td>
<td>3%</td>
</tr>
<tr>
<td>Mix Normal</td>
<td>20%</td>
<td>10%</td>
</tr>
<tr>
<td>Mix Chaos</td>
<td>40%</td>
<td>40%</td>
</tr>
</tbody>
</table>
## Experiment 1

### Classification of responses

Example: 50,000 queries on New York City (2.3 M departures)

<table>
<thead>
<tr>
<th></th>
<th>OPTIMAL</th>
<th>ALMOST A</th>
<th>ALMOST B</th>
<th>BAD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$d = 0$</td>
<td>$d \leq 5 \text{ min}$</td>
<td>$d \leq 10 \text{ min}$</td>
<td>$\wedge \frac{d}{c^*} \leq 0.05$</td>
</tr>
<tr>
<td><strong>NULL</strong></td>
<td>99.99%</td>
<td>0.01%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td><strong>LOW</strong></td>
<td>99.87%</td>
<td>0.09%</td>
<td>0.00%</td>
<td>0.04%</td>
</tr>
<tr>
<td><strong>MEDIUM</strong></td>
<td>99.68%</td>
<td>0.19%</td>
<td>0.03%</td>
<td>0.10%</td>
</tr>
<tr>
<td><strong>HIGH</strong></td>
<td>99.72%</td>
<td>0.17%</td>
<td>0.02%</td>
<td>0.09%</td>
</tr>
<tr>
<td><strong>Mix Low</strong></td>
<td>99.93%</td>
<td>0.05%</td>
<td>0.00%</td>
<td>0.02%</td>
</tr>
<tr>
<td><strong>Mix Normal</strong></td>
<td>99.89%</td>
<td>0.07%</td>
<td>0.01%</td>
<td>0.03%</td>
</tr>
<tr>
<td><strong>Mix Chaos</strong></td>
<td>99.19%</td>
<td>0.57%</td>
<td>0.07%</td>
<td>0.17%</td>
</tr>
</tbody>
</table>

$c := \text{travel time}, \ d := c^* - c$
Experiment 1
Suboptimal responses

- Low
- Medium
- High
- Mix Low
- Mix Normal
- Mix Chaos

Relative offset to optimal travel time

- N = 46
- N = 149
- N = 144
- N = 24
- N = 41
- N = 332
Experiment 2

Directed Delay

- Repeat a fixed query
- Delay the vehicles of the optimal connections
- Until the response becomes suboptimal

![Bar chart showing the number of optimal queries over the number of iterations.]

- Iteration 0: 5000
- Iteration 1: 5000
- Iteration 2: 4927
- Iteration 3: 4806
- Iteration 4: 4661
- Iteration 5: 4455
- Iteration 6: 4258
- Iteration 7: 3996
- Iteration 8: 3737
- Iteration 9: 3460
Summary
Delay-Robustness of Transfer Patterns

Contribution
▶ Strategy to adapt to timetable updates
▶ Empirical proof for robustness to delay

In the paper
▶ Updating the direct connection data in real-time
▶ Detailed experimental evaluation
▶ Analysis of reasons for robustness
▶ Discussion of possible improvements

Find it here: http://drops.dagstuhl.de/opus/volltexte/2013/4243/pdf/5.pdf