On Compact Representation and Robustness of Transfer Patterns in Public Transportation Routing

Master's Thesis

Jonas Sternisko

Institut für Informatik
Albert-Ludwigs-Universität Freiburg

22 March, 2013

Outline

Transfer Pattern Routing

Compact Representation of Transfer Patterns

Robustness of Transfer Patterns

Outline

Transfer Pattern Routing

Compact Representation of Transfer Patterns

Robustness of Transfer Patterns

Outline

Transfer Pattern Routing

Transfer Patterns

- Example Freiburg $\rightarrow$ Munich

  [F, Karlsruhe, M], [F, Titisee, Ulm, M]

State-of-the-art routing algorithm (Hannah Bast et al. [1])

- Optimal transfer patterns
- Efficient direct connection queries

Modeling timetable data

- Time-expanded graph
- Realistic routing adds transfer buffer, walking (Robert Geisberger [2])
- Multi-variate cost model: time of travel, number of transfers
- Pareto-optimal paths by multi-label Dijkstra

Transfer Patterns

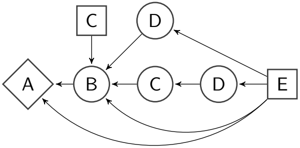
Computation and Storage

Perform a full Dijkstra for every station

- Backtrack optimal paths from each destination
- Transfers along paths yield patterns

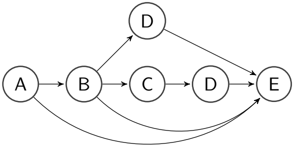
Store patterns as Directed Acyclic Graph

- Reversed, prefix-free
- One DAG per station
- Example for patterns 'ABC', 'AE', 'ABE', 'ABDE', 'ABCDE'



Transfer Patterns

Search



Query Graph

- Construction from patterns
- Arcs present direct connections

Efficient search

- Direct connection queries for arc relaxation
- List-intersection based algorithm

Pareto-optimal paths within a few ms

Transfer Pattern Routing

Compact Representation of Transfer Patterns

Robustness of Transfer Patterns

Motivation

Size of the information

- Hardware requirements
- Access speed
- Future: increased number of patterns (multi-modal route planning)

Motivation

Size of the information $\rightarrow$ *Compact representation*

- Hardware requirements
- Access speed
- Future: increased number of patterns
  (multi-modal route planning)

First Approach

Routing with first transfers

Idea

- Store only the first transfer instead of full patterns
- At search time, recursively construct query graph
- Example Freiburg → Munich

Problems

- Requires same precomputation...
- ...but discards most information
- Less informed, search space twice as large → slower
- Only small advantage in space

First Approach

Routing with first transfers

Idea

- Store only the first transfer instead of full patterns
- At search time, recursively construct query graph
- Example Freiburg → Munich

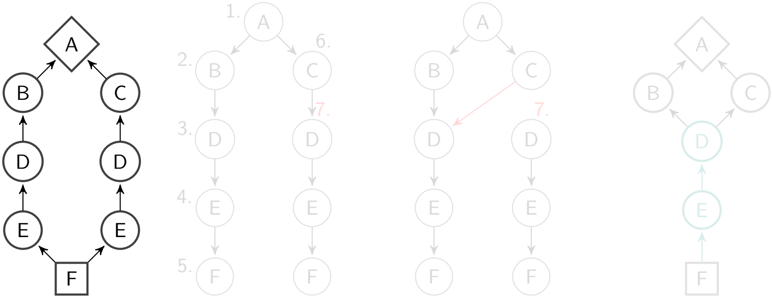
Problems

- Requires same precomputation...
- ...but discards most information
- Less informed, search space twice as large → slower
- Only small advantage in space

Removing Redundancy (1)

Equal suffixes

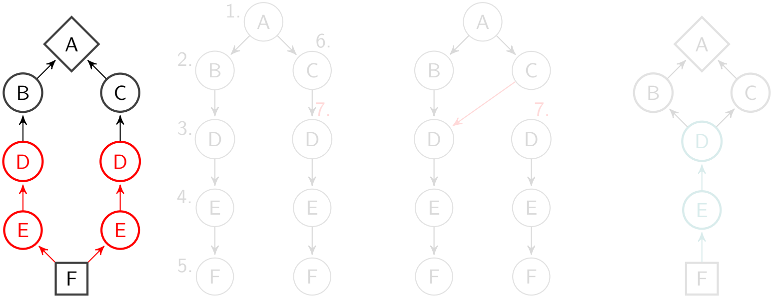
- DAG is prefix-free
- Detect and remove equal suffixes



Removing Redundancy (1)

Equal suffixes

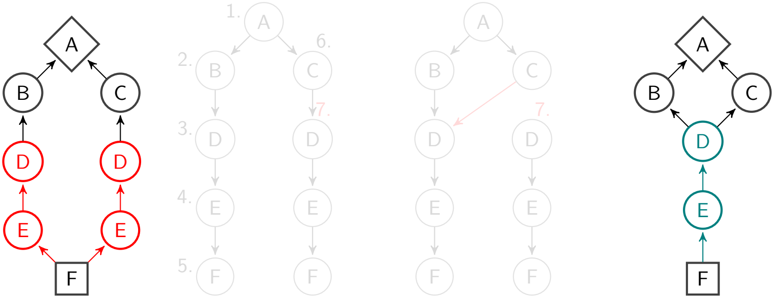
- DAG is prefix-free
- Detect and remove equal suffixes



Removing Redundancy (1)

Equal suffixes

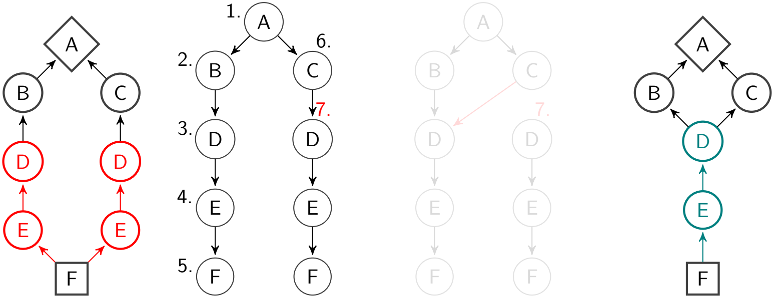
- DAG is prefix-free
- Detect and remove equal suffixes



Removing Redundancy (1)

Equal suffixes

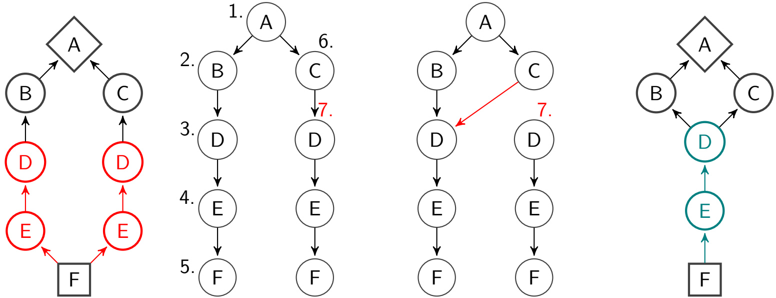
- DAG is prefix-free
- Detect and remove equal suffixes



Removing Redundancy (1)

Equal suffixes

- DAG is prefix-free
- Detect and remove equal suffixes



Removing Redundancy (2)

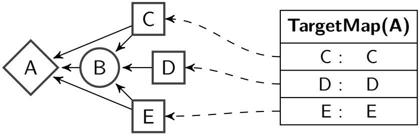
Entry points

Information of destination nodes

- Station id + successors

Destination map determines station id as well

- Merge destination nodes with equal successors



Removing Redundancy (2)

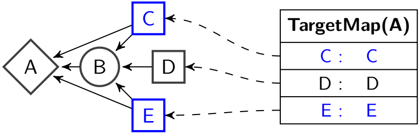
Entry points

Information of destination nodes

- Station id + successors

Destination map determines station id as well

- Merge destination nodes with equal successors



Removing Redundancy (2)

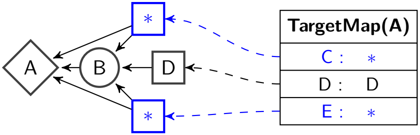
Entry points

Information of destination nodes

- Station id + successors

Destination map determines station id as well

- Merge destination nodes with equal successors



Removing Redundancy (2)

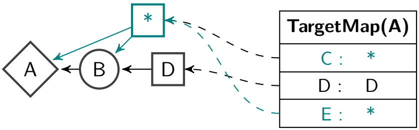
Entry points

Information of destination nodes

- Station id + successors

Destination map determines station id as well

- Merge destination nodes with equal successors



Removing Redundancy (3)

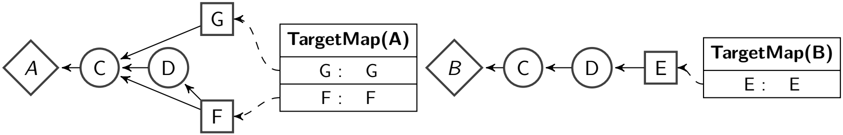
Joint Graph

Observation

- Information of the departure node is redundant
- Context determines station id

Let all patterns share a common root

- Joint DAG resolves redundancy between all DAGs
- Station id is assigned at query time
- Other techniques can be applied on top of that



Removing Redundancy (3)

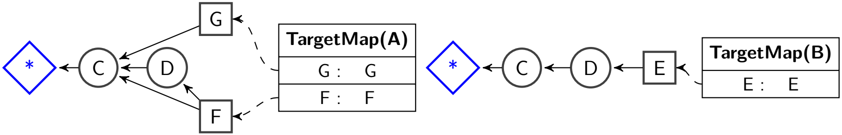
Joint Graph

Observation

- Information of the departure node is redundant
- Context determines station id

Let all patterns share a common root

- Joint DAG resolves redundancy between all DAGs
- Station id is assigned at query time
- Other techniques can be applied on top of that



Removing Redundancy (3)

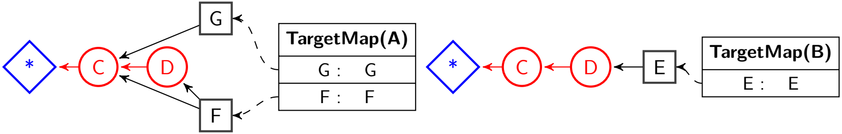
Joint Graph

Observation

- Information of the departure node is redundant
- Context determines station id

Let all patterns share a common root

- Joint DAG resolves redundancy between all DAGs
- Station id is assigned at query time
- Other techniques can be applied on top of that



Removing Redundancy (3)

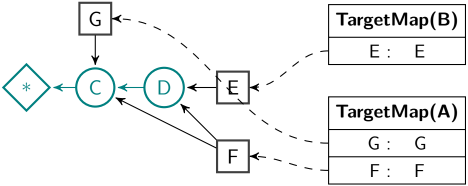
Joint Graph

Observation

- Information of the departure node is redundant
- Context determines station id

Let all patterns share a common root

- Joint DAG resolves redundancy between all DAGs
- Station id is assigned at query time
- Other techniques can be applied on top of that



Removing Redundancy (3)

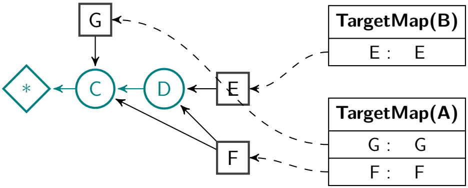
Joint Graph

Observation

- Information of the departure node is redundant
- Context determines station id

Let all patterns share a common root

- Joint DAG resolves redundancy between all DAGs
- Station id is assigned at query time
- Other techniques can be applied on top of that



Results

Compact Representation

- Computed transfer patterns for Hawaii, Detroit, Toronto, NYC
- Measure size of 4 representations
- Example: Toronto at 1000m walking distance (162M patterns, 5.8 patterns per destination in average) [1]

Toronto@1000m: 162.0M, 5.8

	TP	TP^c	*jDAG*	$jDAG^c$
# internal nodes	24.2M	20.6M	3.2M	3.0M
# destination nodes	27.9M	11.7M	27.9M	7.2M
# arcs	186.2M	110.0M	165.2M	78.3M
Memory size (Byte)	1.6G	1.1G	1.3G	659.8M
Byte/pattern	9.8	6.5	7.8	4.1

[1] TP^c: techniques 1 + 2, *jDAG*: technique 3, $jDAG^c$: 1 + 2 + 3

Results

Compact Representation

Evaluation (general)

- Removing equal suffixes saves 10–20% internal nodes
- Merging destination nodes removes 50–80% nodes and a lot of arcs
- Joint DAG shrinks the internal structures by factor $\sim$8
- Approaches combine very well
- Destination maps become dominant part of the data

Transfer Pattern Routing

Compact Representation of Transfer Patterns

Robustness of Transfer Patterns

Motivation

Transfer patterns vs. real-time updates

Time-consuming precomputation

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

Realistic applications

- Frequent updates: delay
  - Traffic jam, strike, cow on the track, ...
- Can the patterns still guarantee optimal responses?

Motivation

Transfer patterns vs. real-time updates

Time-consuming precomputation → *Robustness*

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

Realistic applications

- Frequent updates: delay
  - Traffic jam, strike, cow on the track, ...
- Can the patterns still guarantee optimal responses?

Delay model

Scenarios

- Delay fixed percentage of trips
- Random, exponentially distributed offset
- After random insertion stop

Scenario	Share of trips and average delay
Low	25% : 5 min
Medium	25% : 15 min
High	25% : 50 min
Switzerland	10% : 5 min, 3% : 15 min, 1% : 50 min
Germany	20% : 5 min, 10% : 15 min, 5% : 50 min
India	40% : 5 min, 40% : 15 min, 20% : 50 min

Experimental setup

- Compute transfer patterns
- Update network
- Answer random location queries
  - Original transfer pattern, updated direct connection data
  - Dijkstra on updated network
- Compare and classify responses

Class	Difference d to optimal path costs c^*
OPTIMAL	$d = 0$
ALMOST OPTIMAL A	$d \leq 5\text{min} \wedge \frac{d}{c^*} \leq 5\%$
ALMOST OPTIMAL B	$d \leq 10\text{min} \wedge \frac{d}{c^*} \leq 10\%$
FAILING	otherwise

Results

Robustness

Example: 50,000 successful queries on Toronto

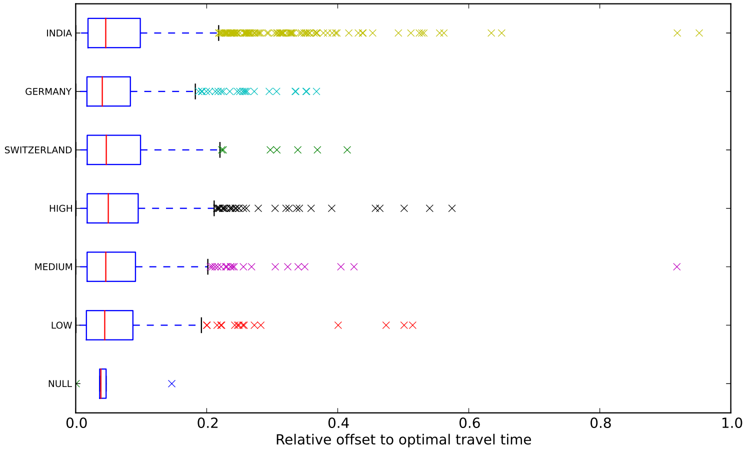
- Classification of responses

	optimal	almost a	almost b	failing
Null	99.98%	0.00%	0.00%	0.00%
Low	99.73%	0.12%	0.04%	0.08%
Medium	99.59%	0.20%	0.06%	0.13%
High	99.49%	0.27%	0.07%	0.16%
Switzerland	99.82%	0.09%	0.02%	0.05%
Germany	99.54%	0.22%	0.06%	0.16%
India	97.85%	1.12%	0.31%	0.70%

Results

Robustness

- Suboptimal responses: time of travel



Results

Robustness

Evaluation (general)

- Never more than 5% suboptimal queries
- Majority of suboptimal responses is almost optimal
- Even under worst scenario INDIA
- But: A few critical outliers

Summary

Compact Representation

Contribution

- Understanding sources of redundancy
- Several techniques reducing data size, maintain accessibility
- Store twice as many patterns in the same memory

Future work

- Dominant destination maps
  - Joint destination maps
  - Invest in hub selection strategies
- Space-efficient implementation

Summary

Robustness

Contribution

- Indication for robustness
- Even for extreme scenarios
- Quality guarantee for transfer pattern routing

Future work

- Acquire and test with real data
- Dependency from precomputation parameters
- How to improve robustness?

Bibliography

- Hannah Bast, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev, and Fabien Viger.
  Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns.
  In Mark de Berg and Ulrich Meyer, editors, *ESA (1)*, volume 6346 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2010.

- Robert Geisberger.
  Advanced Route Planning in Transportation Networks.
  PhD thesis, KIT, 2011.