Karlsruhe Institute of Technology

Trip-Based Public Transit Routing

Sascha Witt – *sascha.witt@kit.edu*

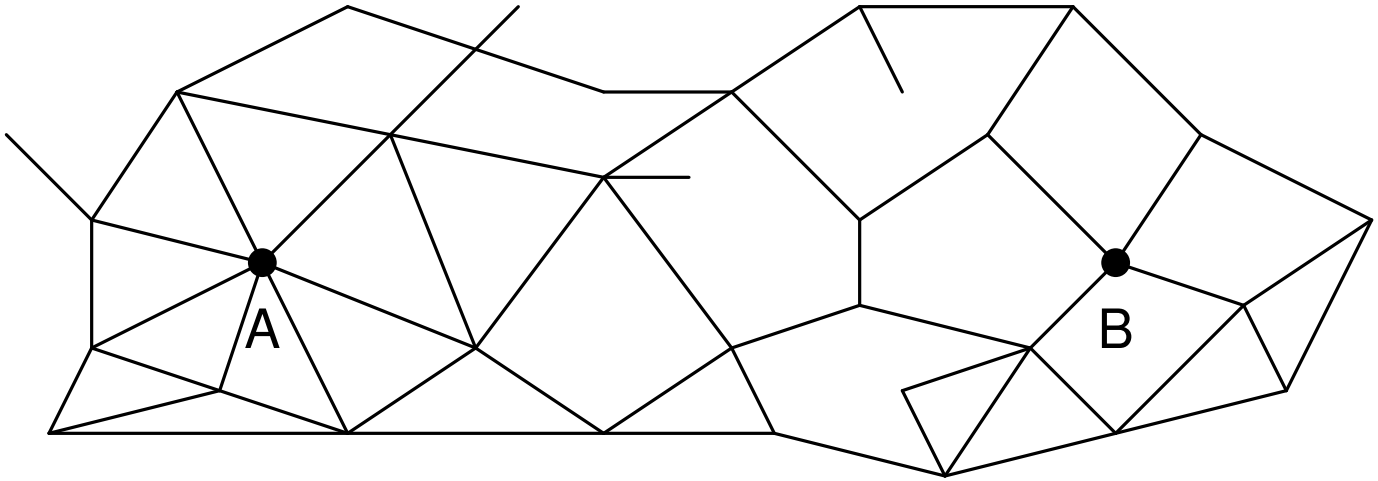
Institute of Theoretical Informatics – Algorithmics

Agenda

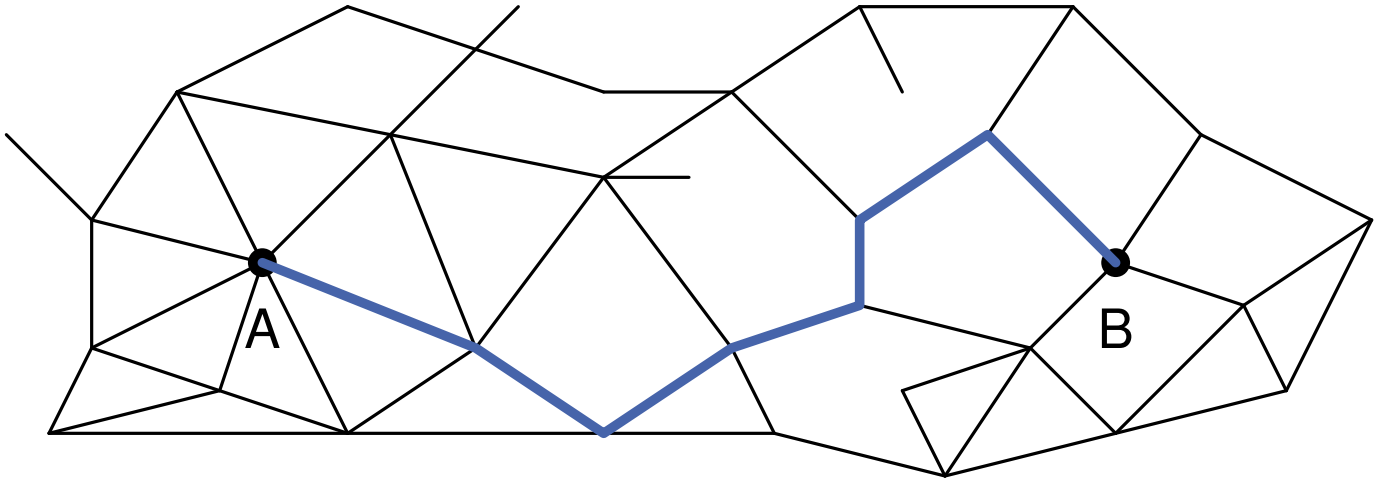
- Introduction
- Public Transit Routing
- Trip-Based Model and Algorithm
- Experiments
- Conclusion

Introduction
Routing 101

Introduction
Routing 101



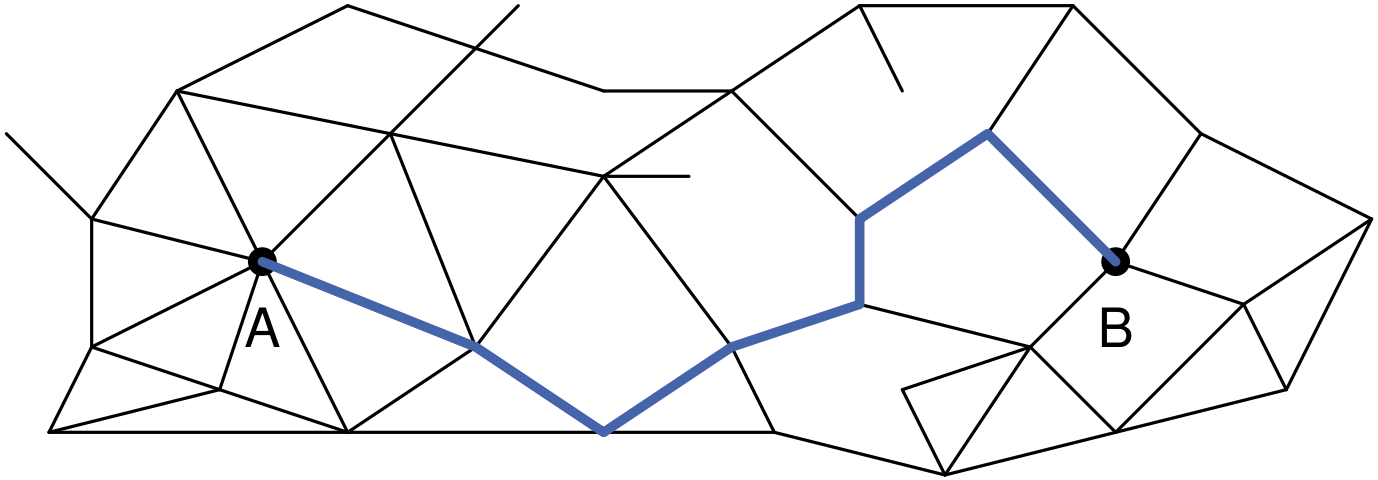
Introduction
Routing 101



Introduction
Routing 101


Goal-directed


Separator-based

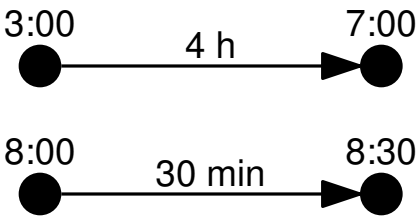



Hierarchical


Bounded-Hop

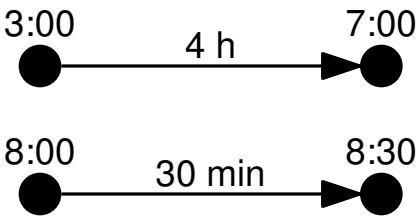
Introduction
Public Transit Routing

- Inherently time-dependent: Travel times depend on departure time

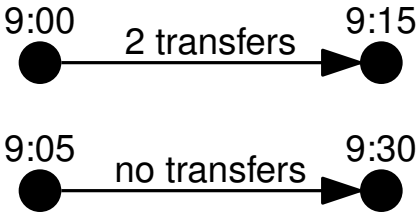


Introduction
Public Transit Routing

- Inherently time-dependent: Travel times depend on departure time



- Multiple natural problem variants
  - Earliest arrival queries
  - Profile (range) queries
  - Multi-criteria queries (e.g., number of transfers taken)



Public Transit Routing

Terminology

Public Transit Routing

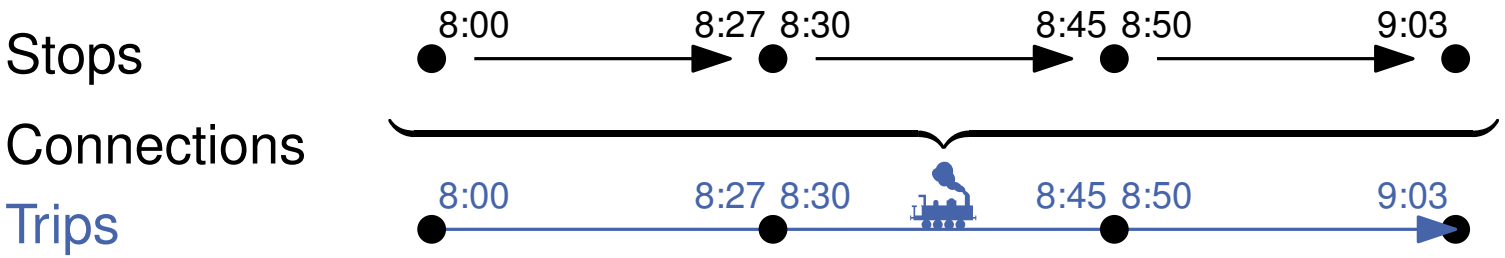
Terminology

Stops

Connections

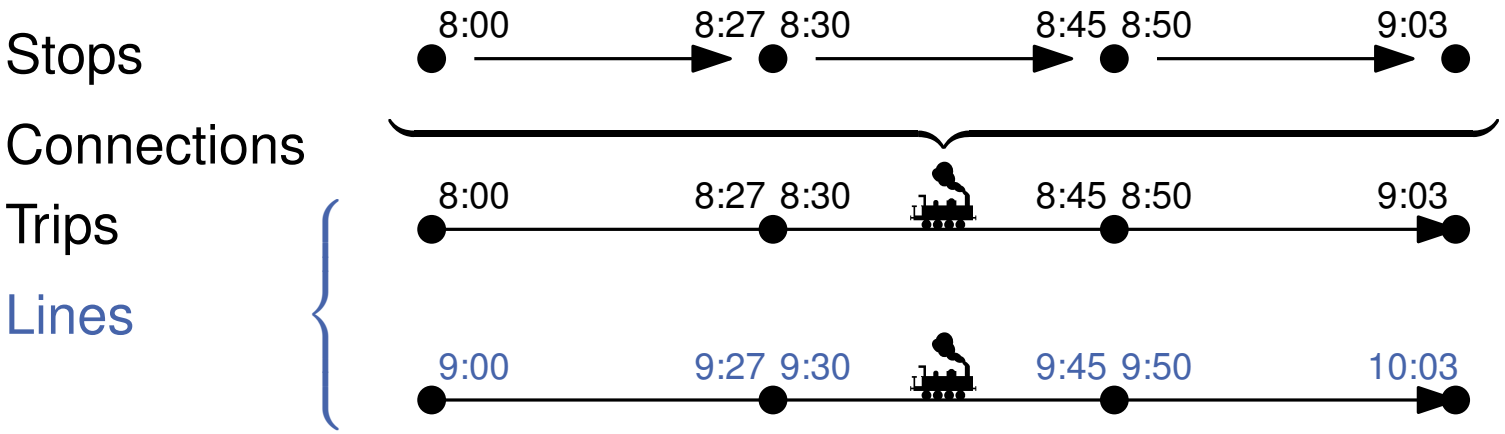
Public Transit Routing

Terminology



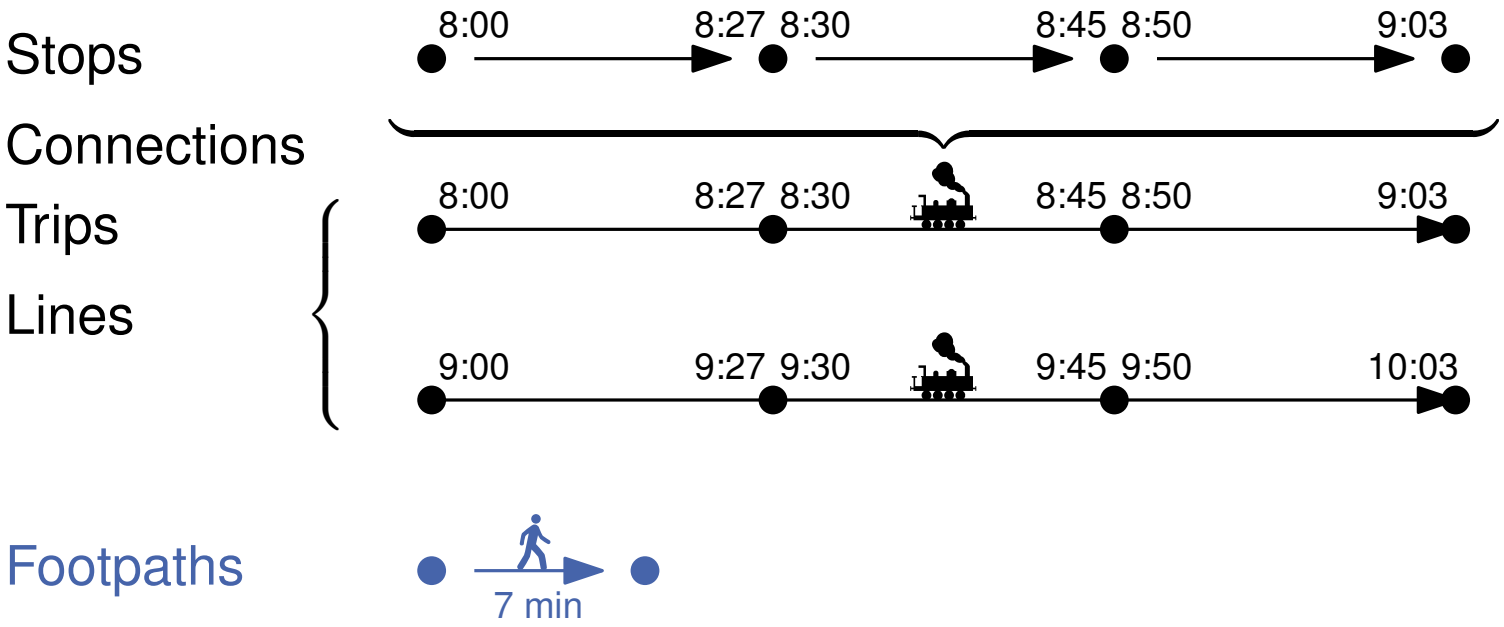
Public Transit Routing

Terminology



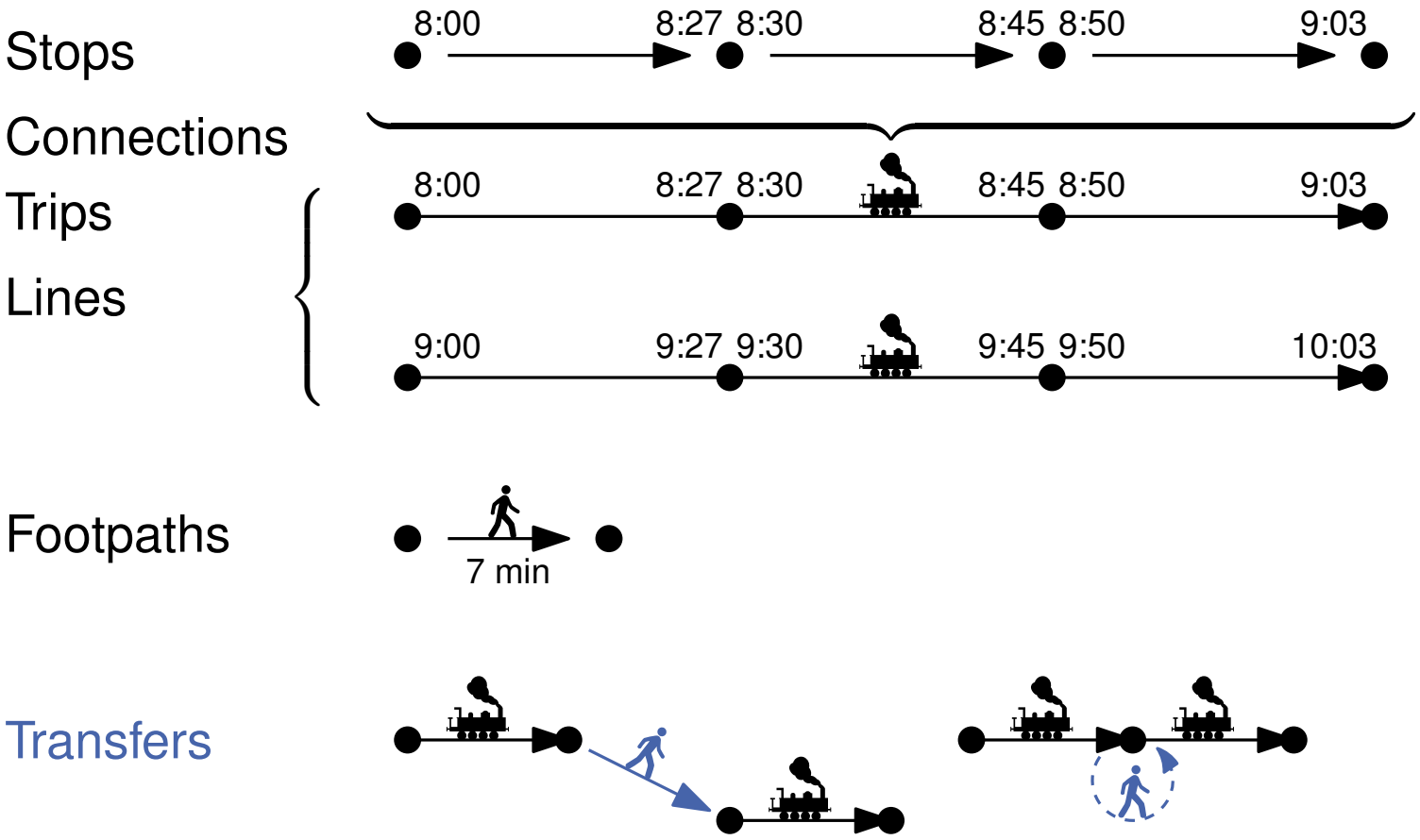
Public Transit Routing

Terminology



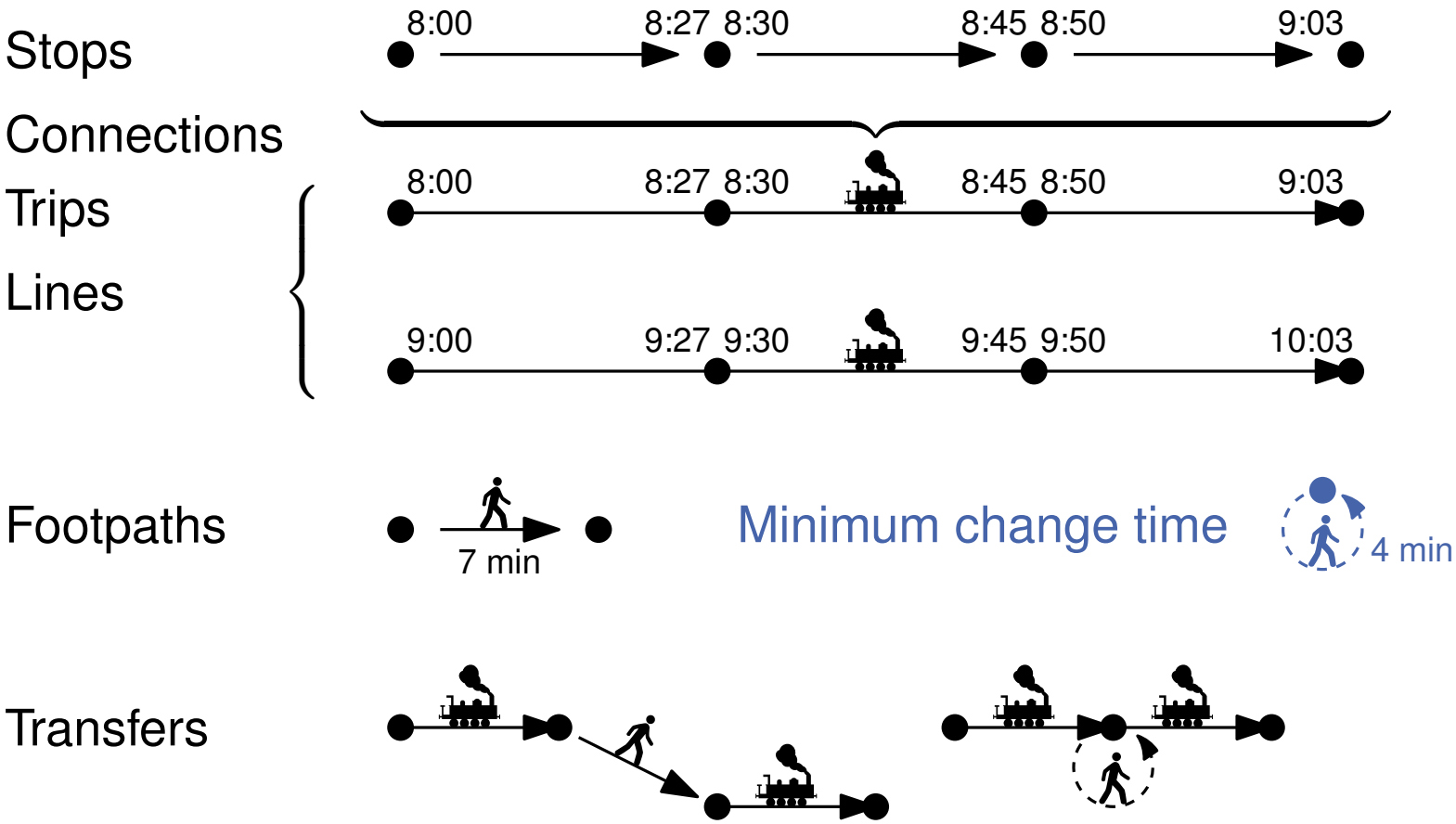
Public Transit Routing

Terminology



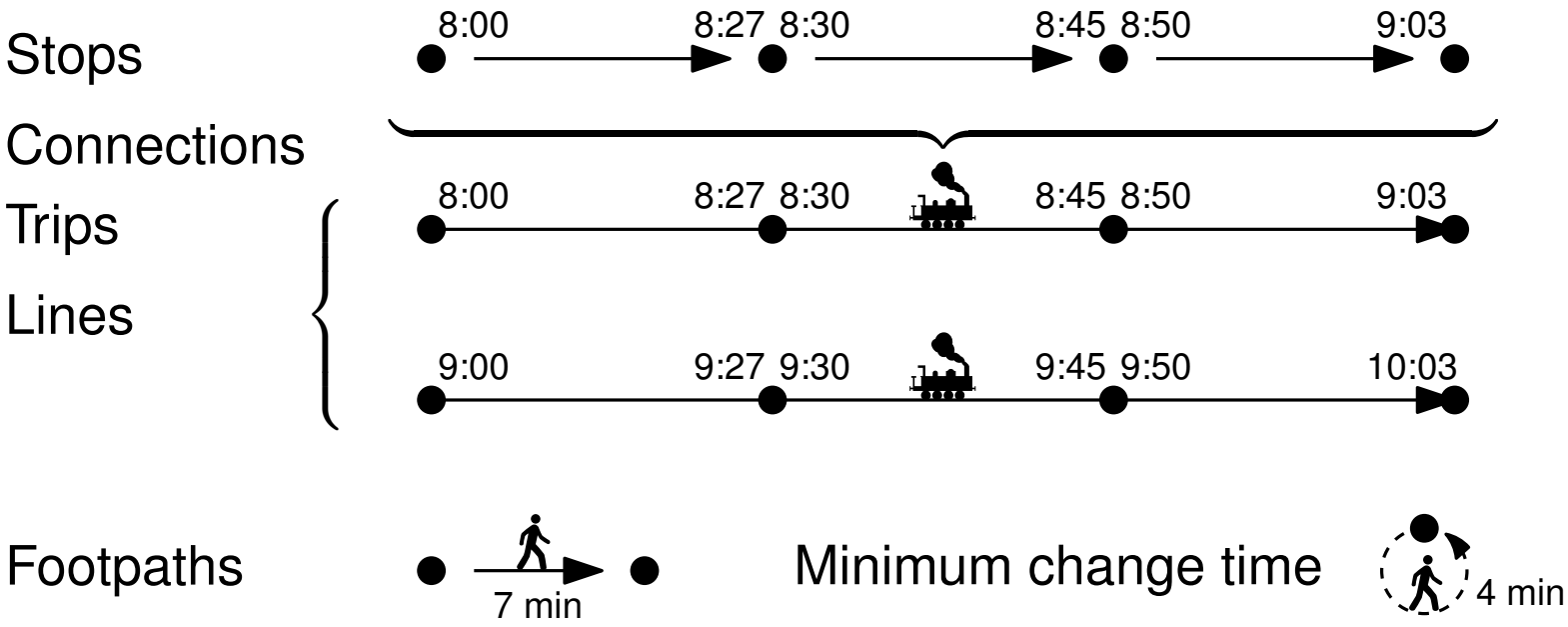
Public Transit Routing

Terminology



Public Transit Routing

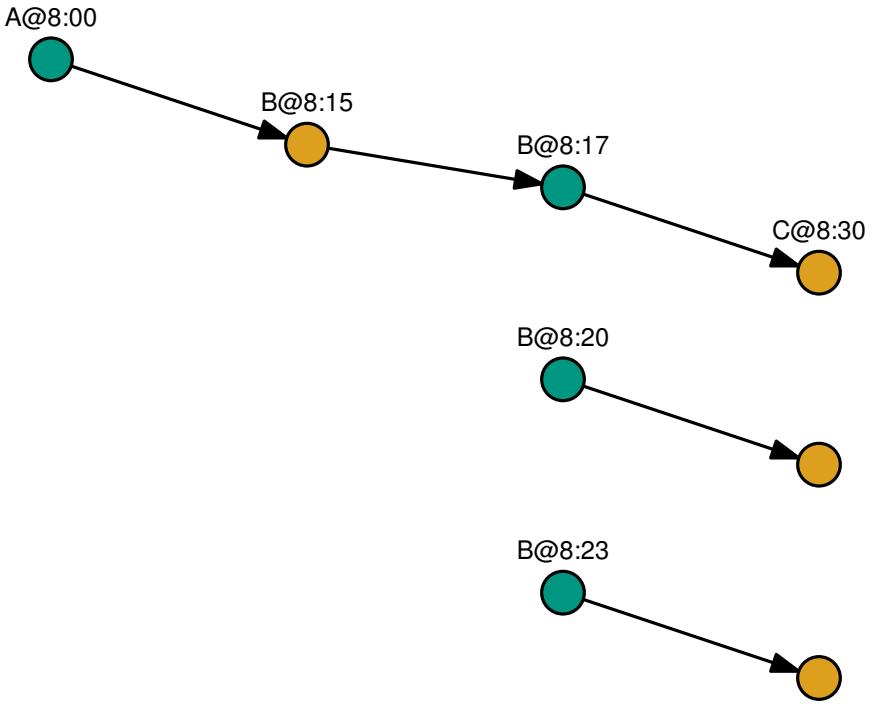
Terminology



Public Transit Routing

Time-Expanded Model

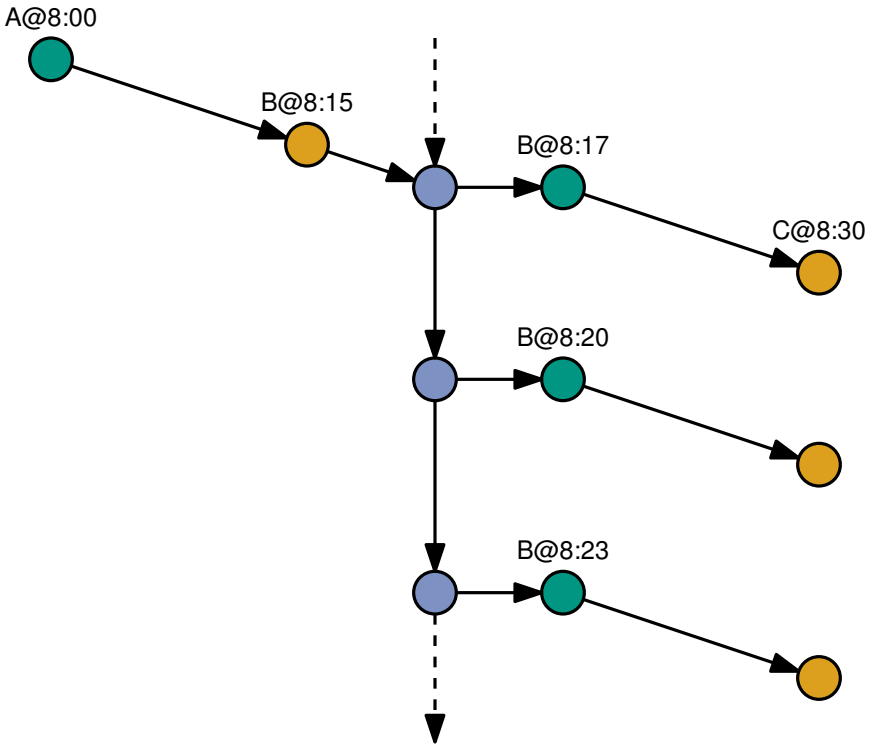
- One vertex per event



Public Transit Routing

Time-Expanded Model

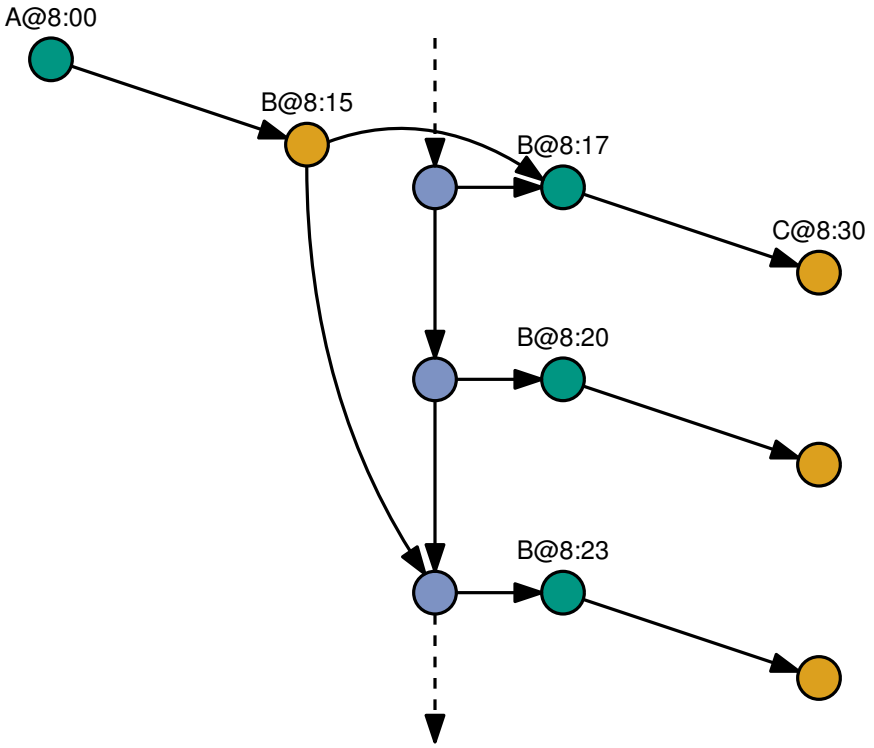
- One vertex per event
- Plus transfer vertices



Public Transit Routing

Time-Expanded Model

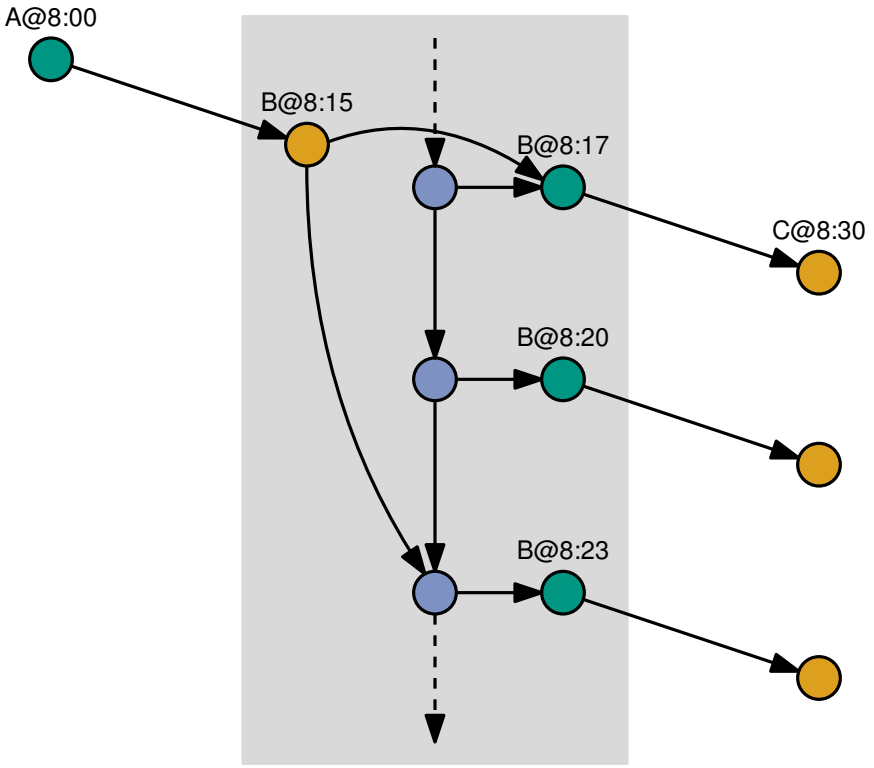
- One vertex per event
- Plus transfer vertices
- Transfer arcs obey minimum change times
- Extra arcs allow staying in vehicle



Public Transit Routing

Time-Expanded Model

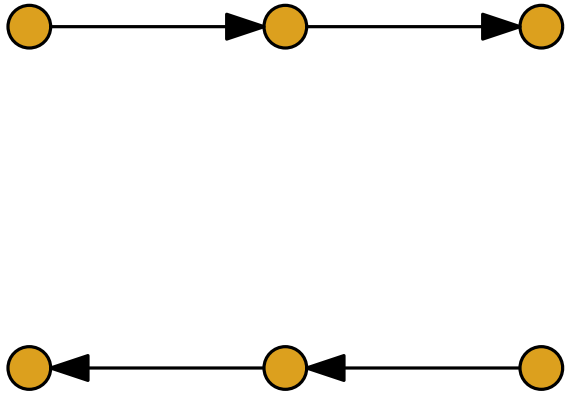
- One vertex per event
- Plus transfer vertices
- Transfer arcs obey minimum change times
- Extra arcs allow staying in vehicle
- Many vertices for each stop



Public Transit Routing

Time-Dependent Model

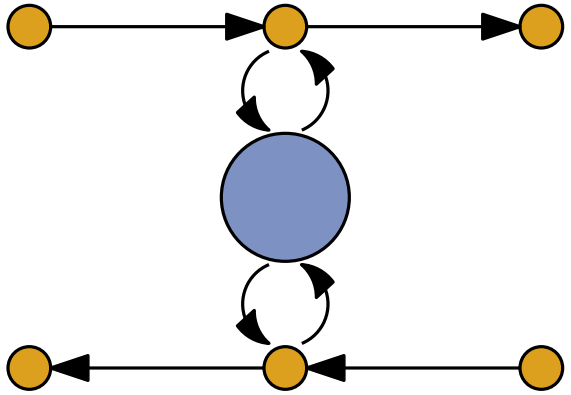
- One vertex per line at each stop



Public Transit Routing

Time-Dependent Model

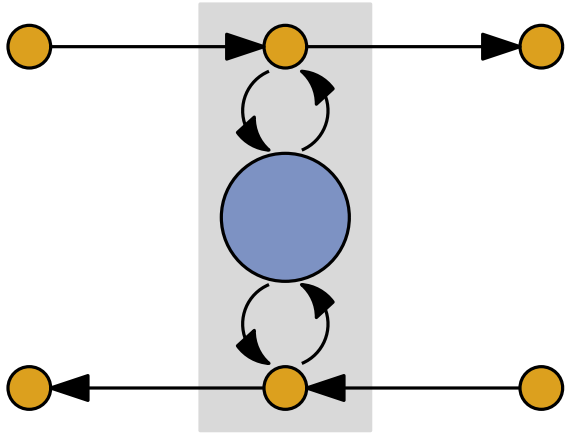
- One vertex per line at each stop
- Plus one transfer vertex per stop



Public Transit Routing

Time-Dependent Model

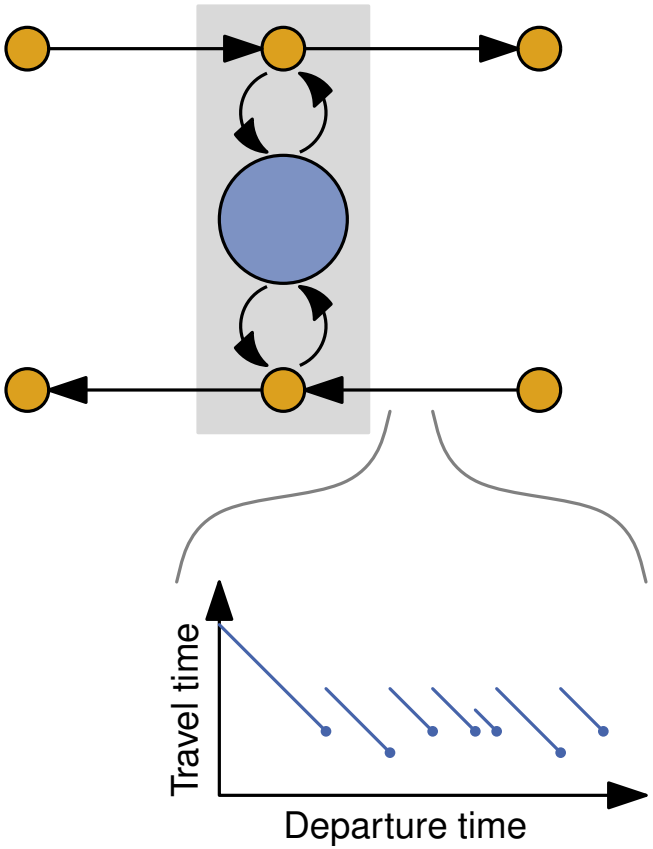
- One vertex per line at each stop
- Plus one transfer vertex per stop
- Fewer vertices and edges



Public Transit Routing

Time-Dependent Model

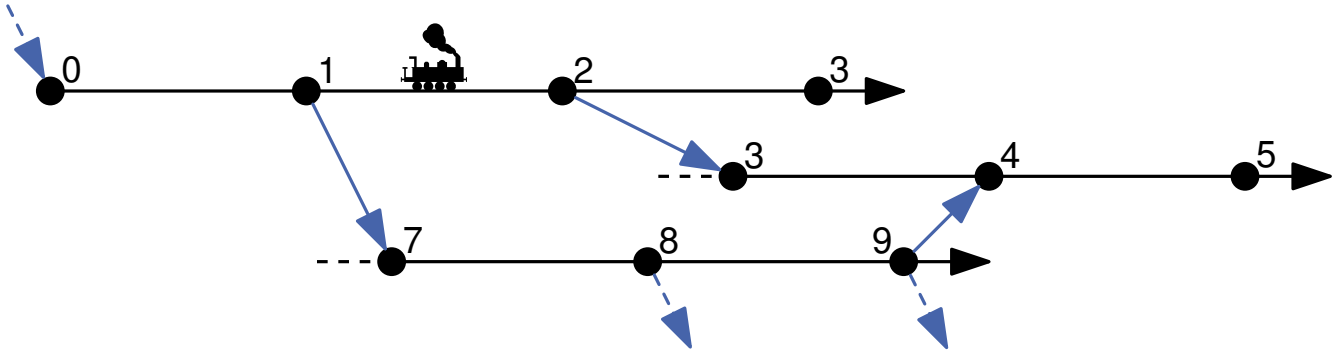
- One vertex per line at each stop
- Plus one transfer vertex per stop
- Fewer vertices and edges
- Edge weights replaced by functions



Trip-Based Routing

Motivation

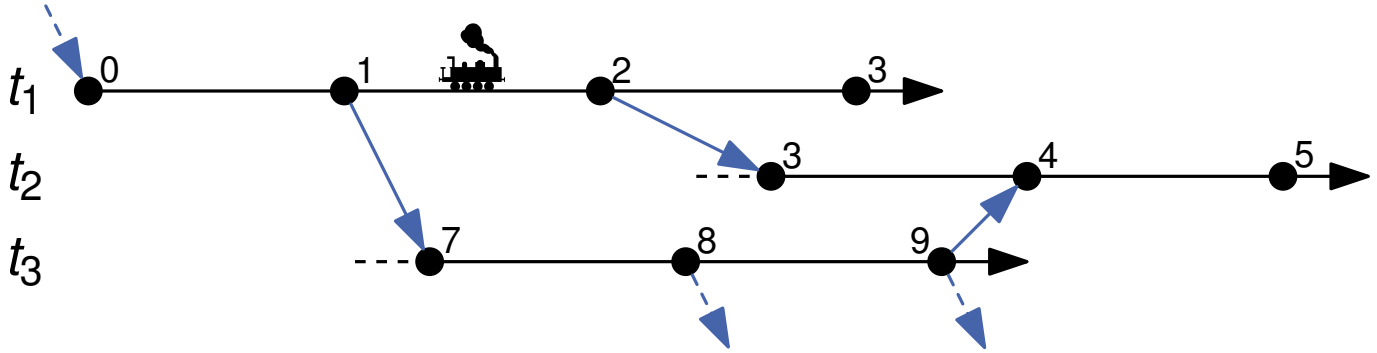
- Move emphasis from stops to trips
- Model transfers between trips explicitly



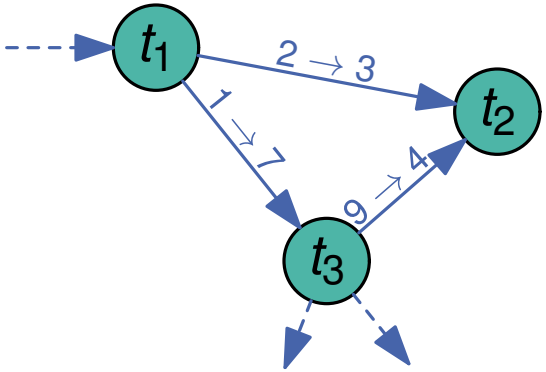
Trip-Based Routing

Motivation

- Move emphasis from stops to trips
- Model transfers between trips explicitly



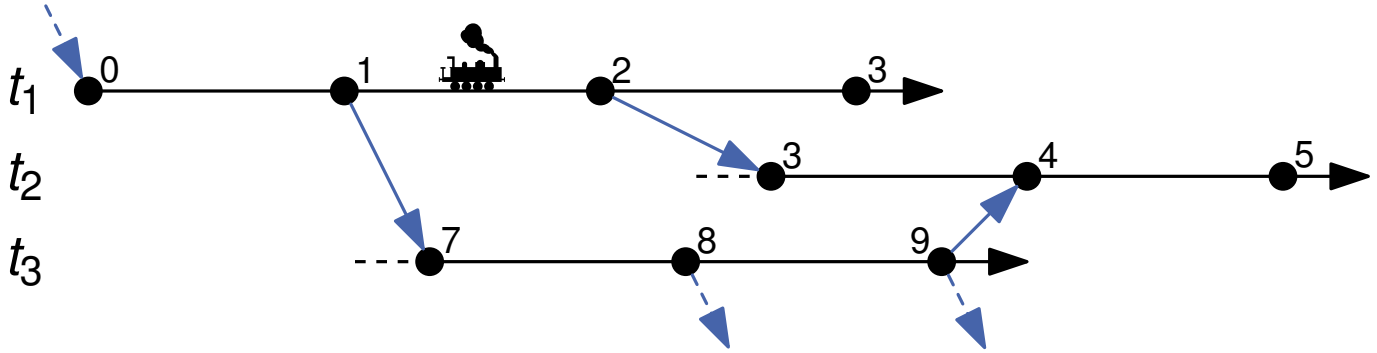
- Query is similar to breadth-first search
- Levels correspond to number of transfers



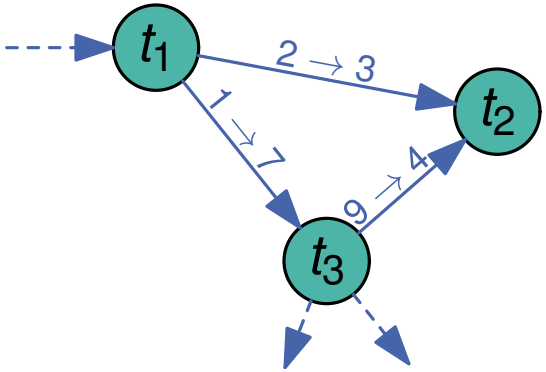
Trip-Based Routing

Motivation

- Move emphasis from stops to trips
- Model transfers between trips explicitly



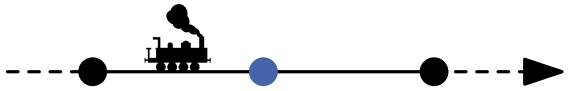
- Query is similar to breadth-first search
- Levels correspond to number of transfers
- Footpaths etc. are handled during preprocessing



Trip-Based Routing

Preprocessing

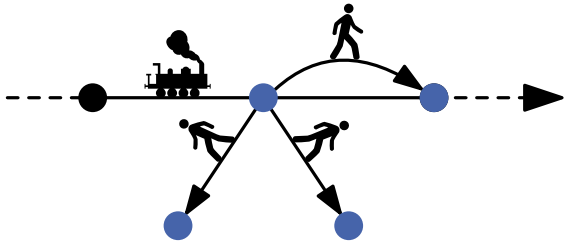
- Compute transfers between trips



Trip-Based Routing

Preprocessing

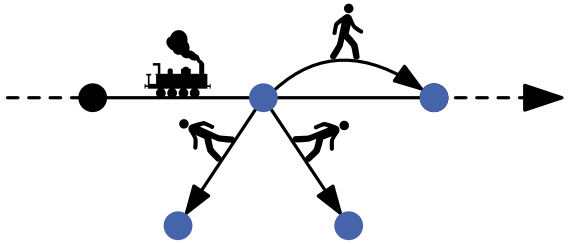
- Compute transfers between trips



Trip-Based Routing

Preprocessing

- Compute transfers between trips

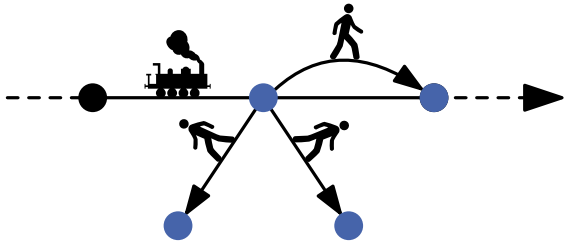


- For each line, find the first reachable trip
  $(\text{arrival time} + \text{footpath length} \leq \text{departure time})$

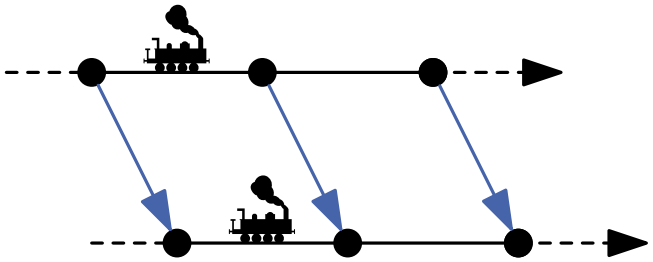
Trip-Based Routing

Preprocessing

- Compute transfers between trips



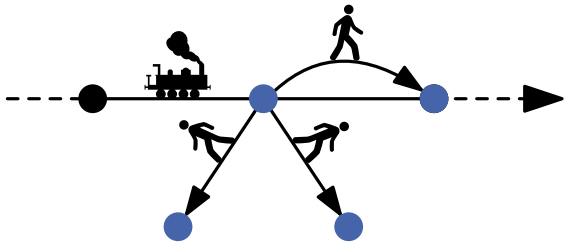
- For each line, find the first reachable trip
  $(\text{arrival time} + \text{footpath length} \leq \text{departure time})$
- Huge number of transfers, not all of which are useful



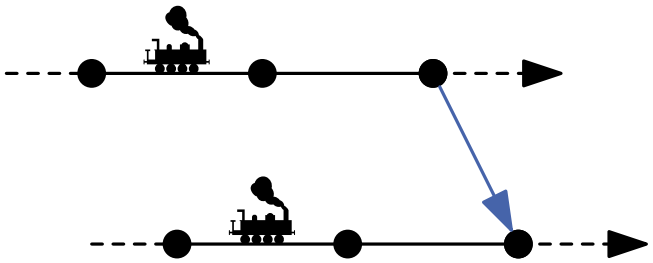
Parallel trips

Trip-Based Routing

Preprocessing



- Compute transfers between trips
- For each line, find the first reachable trip
  (arrival time + footpath length $\leq$ departure time)
- Huge number of transfers, not all of which are useful

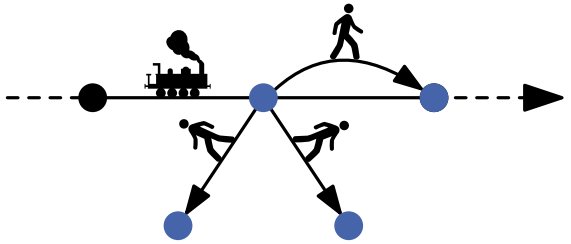


Parallel trips

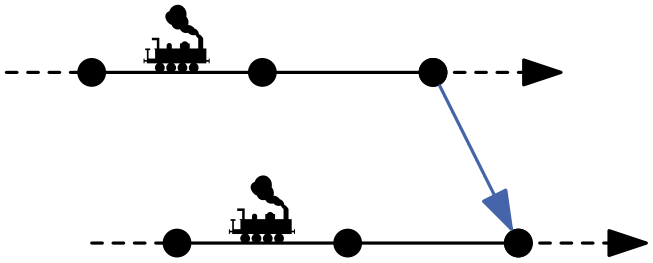
Trip-Based Routing

Preprocessing

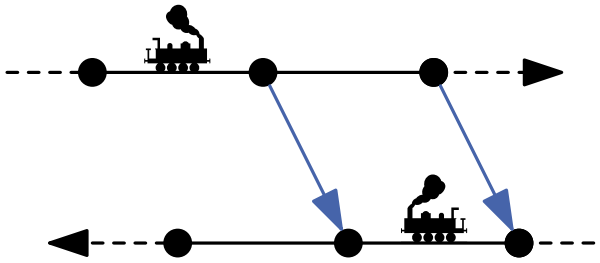
- Compute transfers between trips



- For each line, find the first reachable trip
  (arrival time + footpath length $\leq$ departure time)
- Huge number of transfers, not all of which are useful



Parallel trips

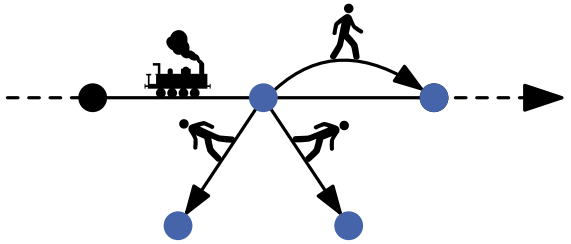


U-turns

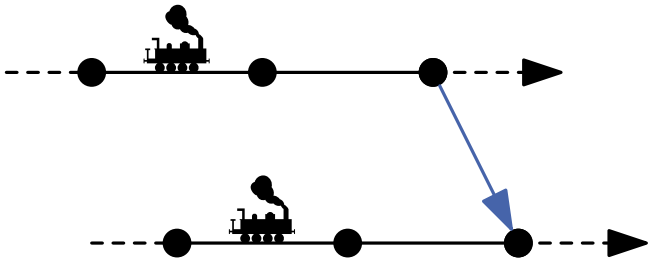
Trip-Based Routing

Preprocessing

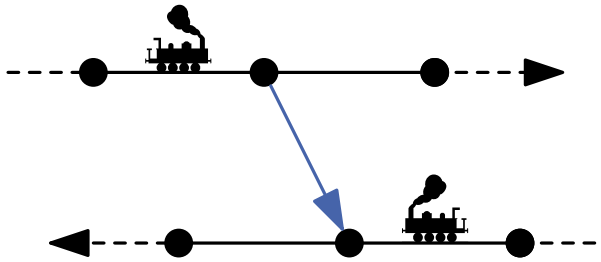
- Compute transfers between trips



- For each line, find the first reachable trip
  (arrival time + footpath length $\leq$ departure time)
- Huge number of transfers, not all of which are useful



Parallel trips



U-turns

Trip-Based Routing

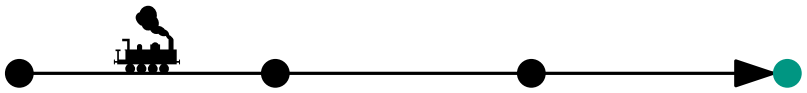
Preprocessing

- Reduce number of transfers by eliminating redundant ones

Trip-Based Routing

Preprocessing

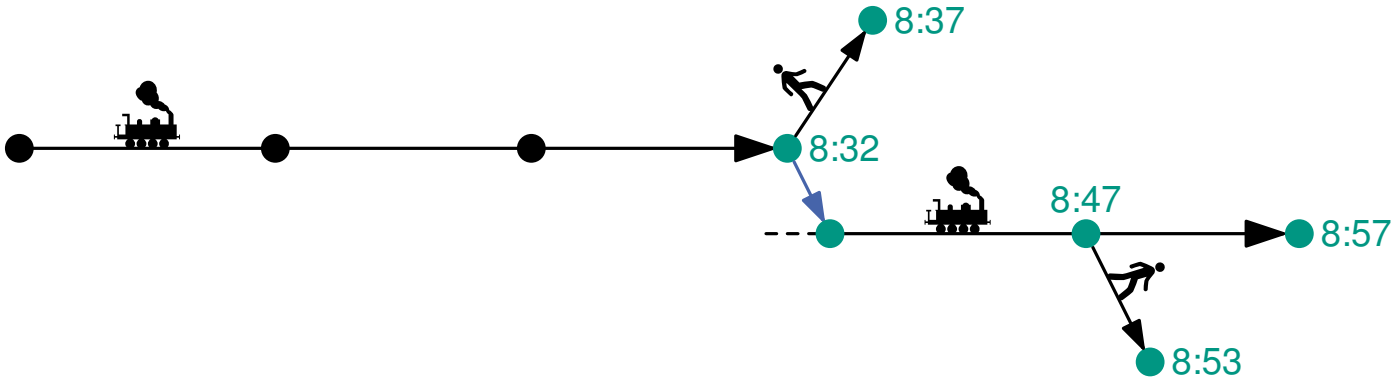
- Reduce number of transfers by eliminating redundant ones
- Process trips backwards



Trip-Based Routing

Preprocessing

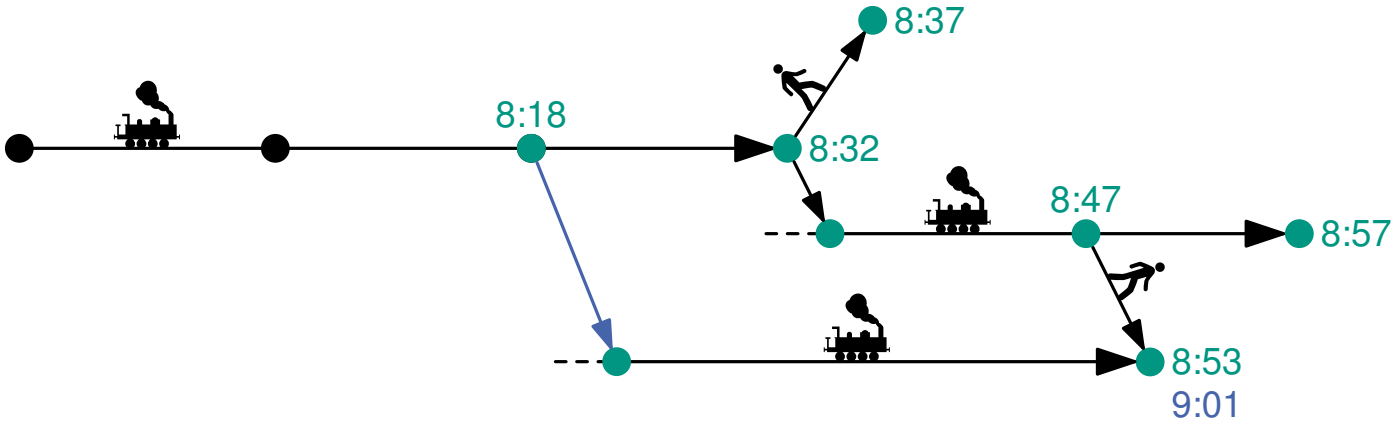
- Reduce number of transfers by eliminating redundant ones
- Process trips backwards
- Keep track of which stops can be reached at what time



Trip-Based Routing

Preprocessing

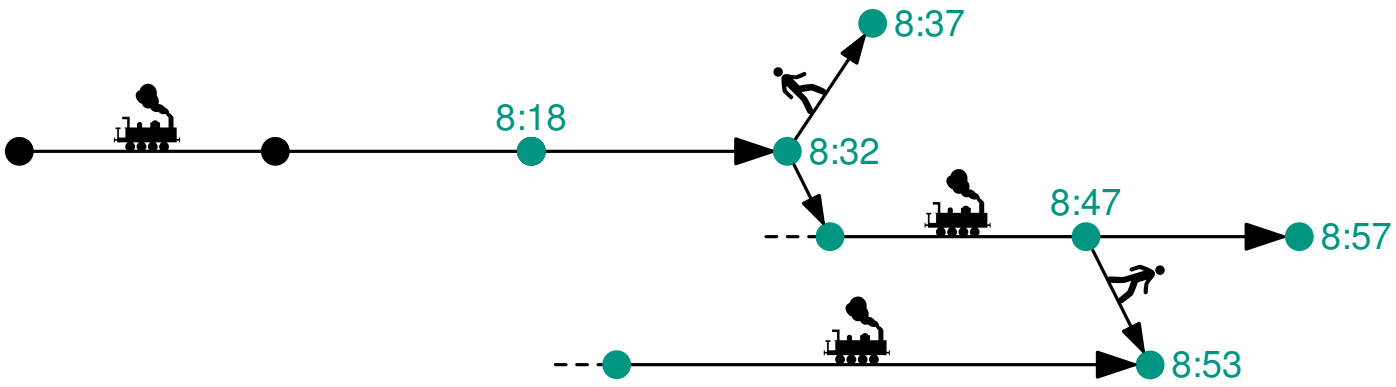
- Reduce number of transfers by eliminating redundant ones
- Process trips backwards
- Keep track of which stops can be reached at what time
- Evaluate transfers by checking if they improve arrival times



Trip-Based Routing

Preprocessing

- Reduce number of transfers by eliminating redundant ones
- Process trips backwards
- Keep track of which stops can be reached at what time
- Evaluate transfers by checking if they improve arrival times
- Removes up to 90% of original transfers



Trip-Based Routing

Query

- Input: Source stop, target stop, departure time

Trip-Based Routing

Query

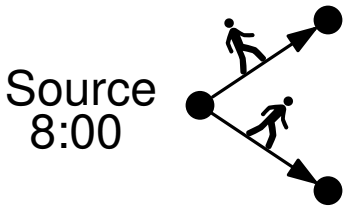
- Input: Source stop, target stop, departure time
- Identify trips reachable from the source

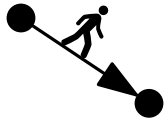
dep.	line	trip	index
8:00	2	15	8
8:03	4	56	0
8:07	11	456	31
9:00	110	3256	6

Trip-Based Routing

Query

- Input: Source stop, target stop, departure time
- Identify trips reachable from the source
- Identify lines reaching the target





dep.	line	trip	index
8:00	2	15	8
8:03	4	56	0
8:07	11	456	31
9:00	110	3256	6

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

Trip-Based Routing

Query

- Queue trips and mark as reached

Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
456	31
457	31
...	
3256	6
...	

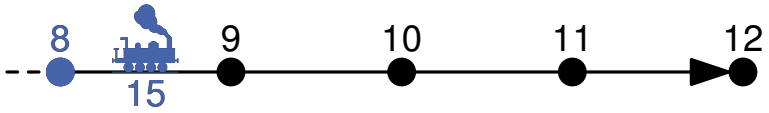
dep.	line	trip	index
8:00	2	15	8
8:03	4	56	0
8:07	11	456	31
9:00	110	3256	6

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue



Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
456	31
457	31
...	
3256	6
...	

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

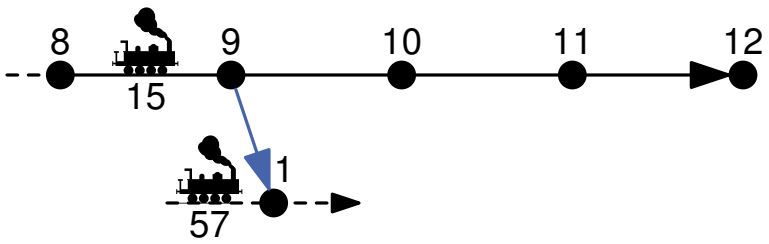
Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers

Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
456	31
457	31
...	
3256	6
...	



line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

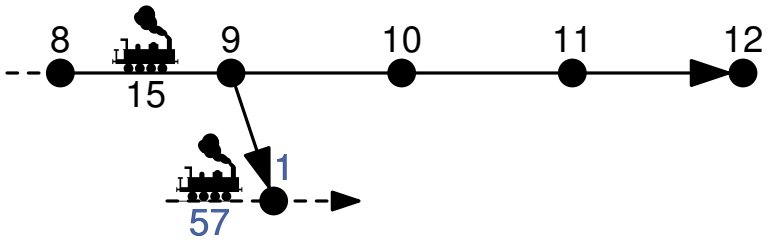
Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers
  - Compare against label

Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
456	31
457	31
...	
3256	6
...	



line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

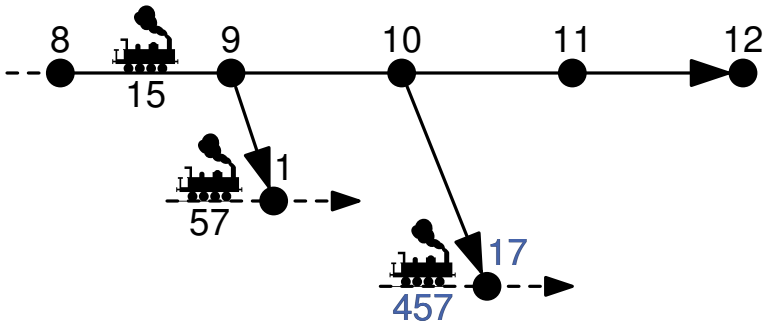
Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers
  - Compare against label

Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
456	31
457	31
...	
3256	6
...	

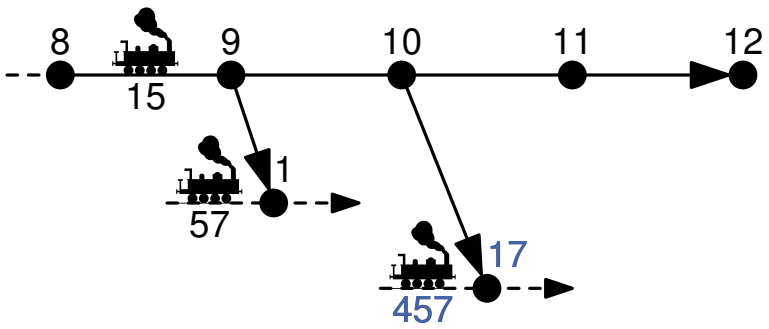


line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers
  - Compare against label
  - Queue newly reached trips



Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45
1	457	17–31

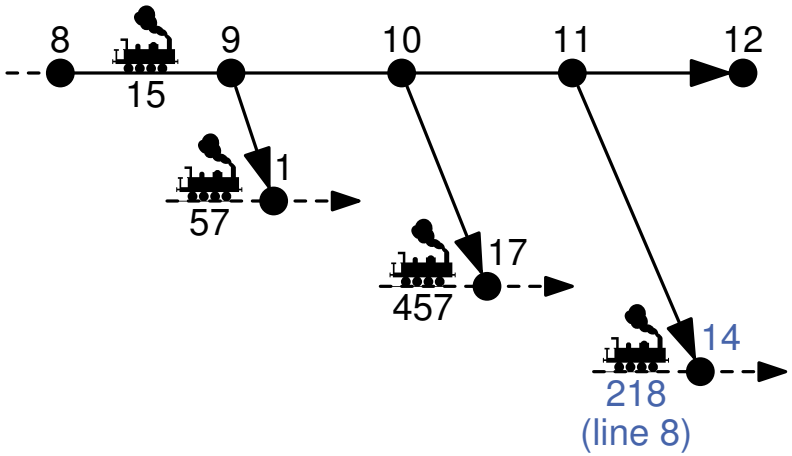
Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
456	31
457	17
458	17
...	
3256	6
...	

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers
  - Compare against label
  - Queue newly reached trips
  - Output a journey if target is reached



Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45
1	457	17–31
1	218	14–23

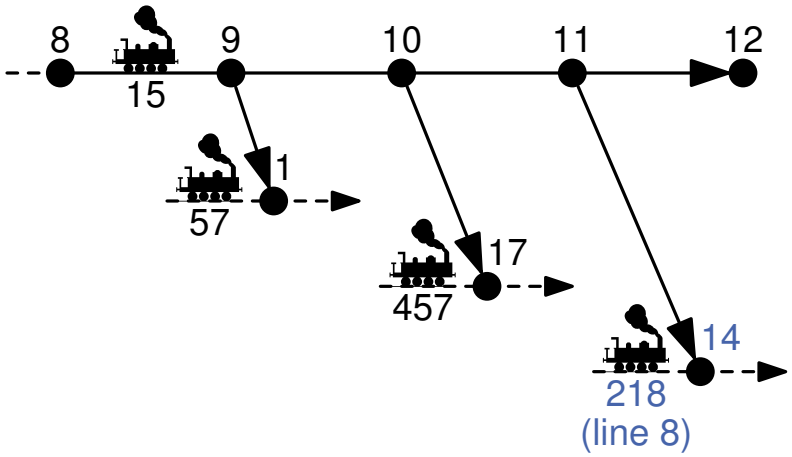
Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
218	14
...	
456	31
457	17
458	17
...	
3256	6
...	

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers
  - Compare against label
  - Queue newly reached trips
  - Output a journey if target is reached



Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45
1	457	17–31
1	218	14–23

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
218	14
...	
456	31
457	17
458	17
...	
3256	6
...	

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

$$\text{arrival\_time}(218, 17) = 9{:}24 \implies \text{Arrival at 9:28 after 1 transfer}$$

Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers
  - Compare against label
  - Queue newly reached trips
  - Output a journey if target is reached
- Continue until queue is empty

Queue		
# tr.	trip	range
0	15	8–12
0	56	0–14
0	456	31–78
0	3256	6–45
1	457	17–31
1	218	14–23

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
218	14
...	
456	31
457	17
458	17
...	
3256	6
...	

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

Arrival at 9:28 after 1 transfer

Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers
  - Compare against label
  - Queue newly reached trips
  - Output a journey if target is reached
- Continue until queue is empty
- Skip trips that cannot improve the currently best arrival time

$\text{departure\_time}(1302, 8) = 9{:}32 > 9{:}28$

Queue		
# tr.	trip	range
1	1302	8–45
1	2871	3–11
2	512	0–19
2	1523	19–88
2	43	13–15
2	44	4–53

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
218	14
...	
456	31
457	17
458	17
...	
3256	6
...	

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

Arrival at 9:28 after 1 transfer

Trip-Based Routing

Query

- Queue trips and mark as reached
- Process queue
- Examine transfers
  - Compare against label
  - Queue newly reached trips
  - Output a journey if target is reached
- Continue until queue is empty
- Skip trips that cannot improve the currently best arrival time

Queue		
# tr.	trip	range
1	2871	3–11
2	512	0–19
2	1523	19–88
2	43	13–15
2	44	4–53

Reached	
...	
15	8
16	8
17	8
...	
56	0
57	0
...	
218	14
...	
456	31
457	17
458	17
...	
3256	6
...	

line	index	footpath
3	8	—
8	17	4 min
27	3	4 min

Arrival at 9:28 after 1 transfer

Experiments

Instances

	London	Germany
Stops	20 764	249 724
Trips	129 263	2 389 253
Connections	4 991 130	46 116 453
Footpaths	45 624	100 470
Lines (Routes)	2 161	232 644
Transfers (full)	121 339 213	1 826 424 894
Transfers (reduced)	19 502 791	186 296 771
Space consumption	115.5 MiB	1 140.9 MiB

Experiments

Instances

	London	Germany
Stops	20 764	249 724
Trips	129 263	2 389 253
Connections	4 991 130	46 116 453
Footpaths	45 624	100 470
Lines (Routes)	2 161	232 644
Transfers (full)	121 339 213	1 826 424 894
Transfers (reduced)	19 502 791	186 296 771
Space consumption	115.5 MiB	1 140.9 MiB

Transfer	Preprocessing Times*	
Computation	3 s	37 s
Reduction	27 s	183 s
Total	30 s	220 s

* Dual 8-core Intel Xeon E5-2650 v2, 2.6 GHz, 128 GB DDR3-1600, 20MB L3, 16 threads

Experiments

Comparison

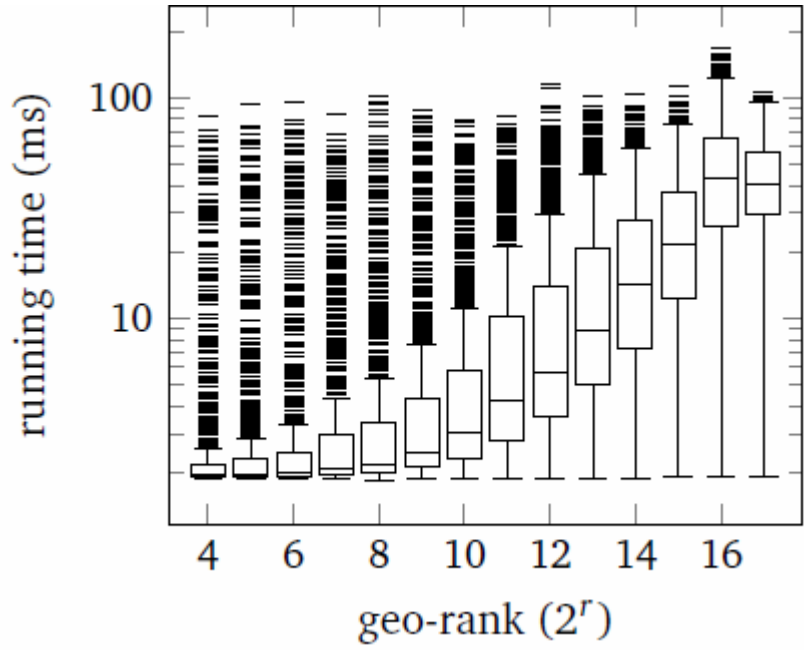
algorithm	instance	stops ($\cdot 10^3$)	conn. ($\cdot 10^6$)	transfers	profile	prep. (h)	query (ms)
Trip-Based	London	20.8	5.0	●	○	< 0.1	1.2
Transfer Patterns	Madrid	4.6	4.8	●	○	185.0	3.1
Public Transit Labeling	London	20.8	5.1	●	○	49.3	0.03
RAPTOR	London	20.8	5.1	●	○	—	5.4
Connection Scan	London	20.8	4.9	○	○	—	1.8
Contraction Hierarchies	Europe (LD)	30.5	1.7	○	○	< 0.1	0.3
Trip-Based	Germany	249.7	46.1	●	○	< 0.1	40.8
Transfer Patterns	Germany	248.4	13.9	●	○	372.0	0.3
Connection Scan	Germany	252.4	46.2	○	○	—	298.6
Accelerated Conn. Scan	Germany	252.4	46.2	○	○	0.2	8.7
Trip-Based	London	20.8	5.0	●	●	< 0.1	70.0
Transfer Patterns	Madrid	4.6	4.8	●	●	185.0	3.1
rRAPTOR	London	20.8	5.1	●	●	—	922.0
Connection Scan	London	20.8	4.9	●	●	—	466.0
Trip-Based	Germany	249.7	46.1	●	●	< 0.1	301.7
Transfer Patterns	Germany	248.4	13.9	●	●	372.0	5.0
Accelerated Conn. Scan	Germany	252.4	46.2	○	●	0.2	171.0

Dual 8-core Intel Xeon E5-2650 v2, 2.6 GHz, 128 GB DDR3-1600, 20MB L3

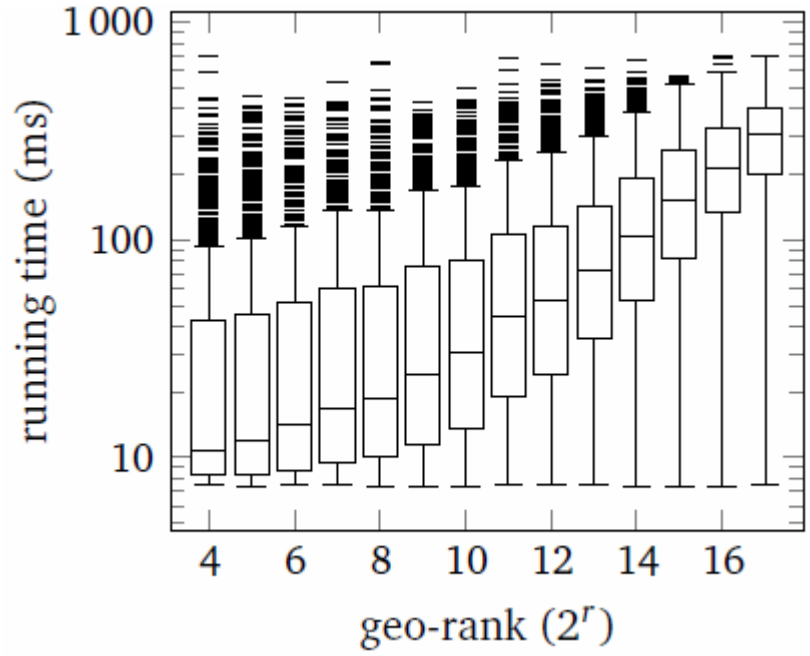
Experiments

Running times by distance

Earliest Arrival Query



Profile Query



Conclusion

- Novel algorithm for routing in public transit networks
- Focused on trips and transfers between them
- Fine-grained modeling
- Competitive performance

Conclusion

- Novel algorithm for routing in public transit networks
- Focused on trips and transfers between them
- Fine-grained modeling
- Competitive performance

- Explore trip hierarchies (and other speedup techniques)
- Enable partial recomputing of transfers for dynamic scenarios
- Exploit periodicity (preprocessing and storage)
- Allow more criteria (financial cost, walking distance, . . .)

Conclusion

- Novel algorithm for routing in public transit networks
- Focused on trips and transfers between them
- Fine-grained modeling
- Competitive performance

- Explore trip hierarchies (and other speedup techniques)
- Enable partial recomputing of transfers for dynamic scenarios
- Exploit periodicity (preprocessing and storage)
- Allow more criteria (financial cost, walking distance, . . .)

- Technical Report at http://arxiv.org/abs/1504.07149