Karlsruhe Institute of Technology

# Trip-Based Public Transit Routing Using Condensed Search Trees

Sascha Witt – *sascha.witt@kit.edu*

Institute of Theoretical Informatics – Algorithmics

# Agenda

- Introduction
- Trip-Based Public Transit Routing
- Prefix Trees
- Splitting Trees
- Experiments
- Conclusion

# Introduction

## Context

- Routing in large, real-world Public Transit Networks
- One-to-one queries
- Pareto-optimal results regarding arrival time and number of transfers
- Earliest arrival and profile/range queries

# Introduction
## Terminology

Stops


**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

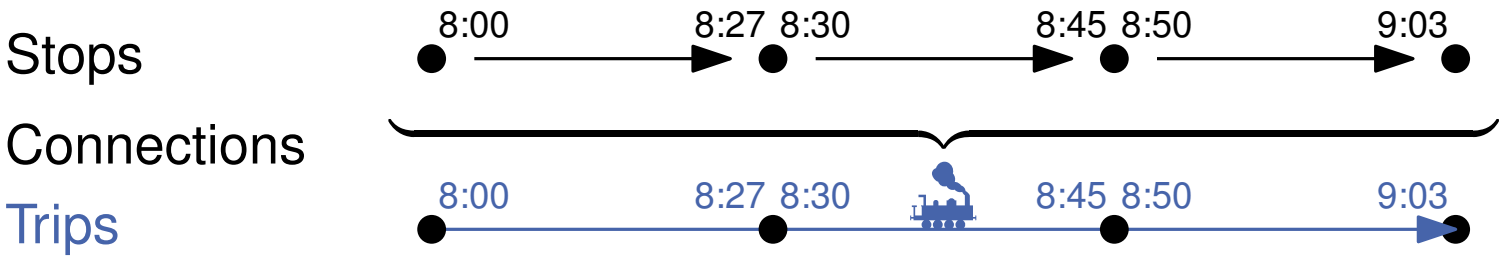
# Introduction
## Terminology

Stops

Connections

# Introduction

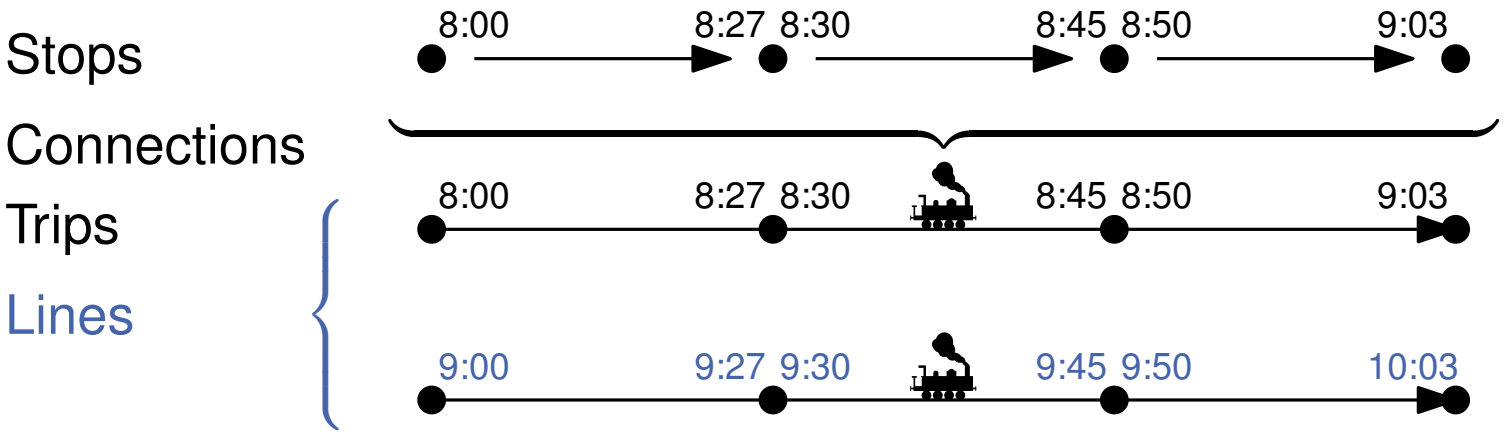
## Terminology



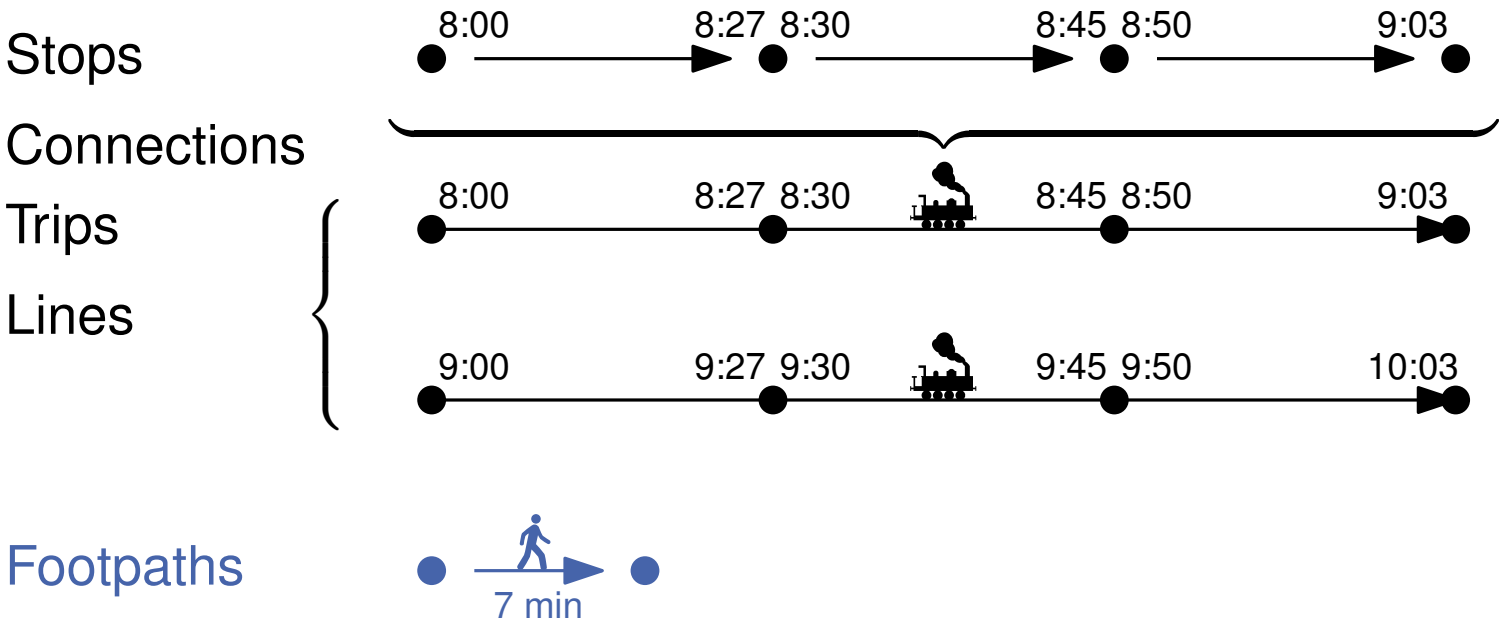


**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Introduction
## Terminology



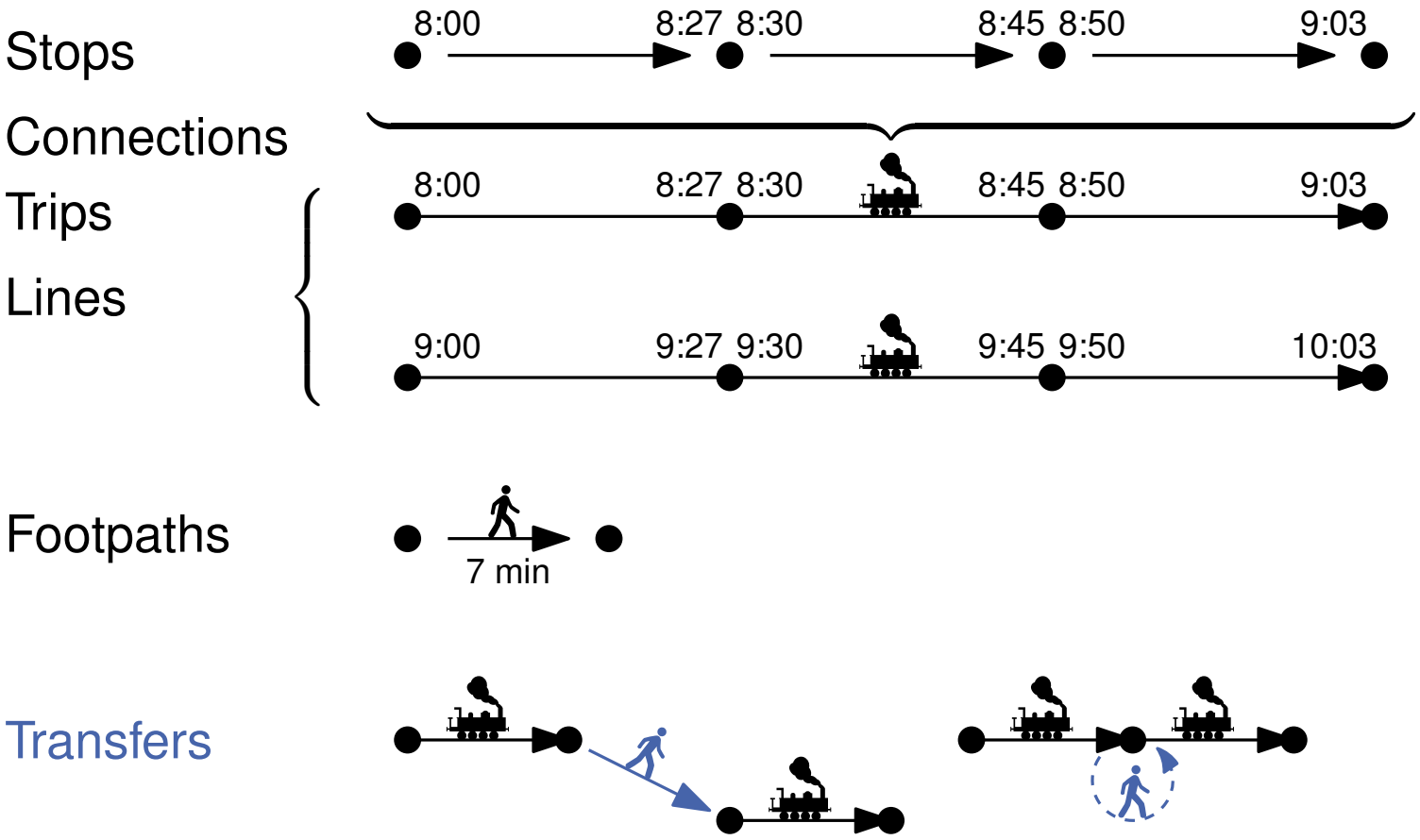
# Introduction
## Terminology



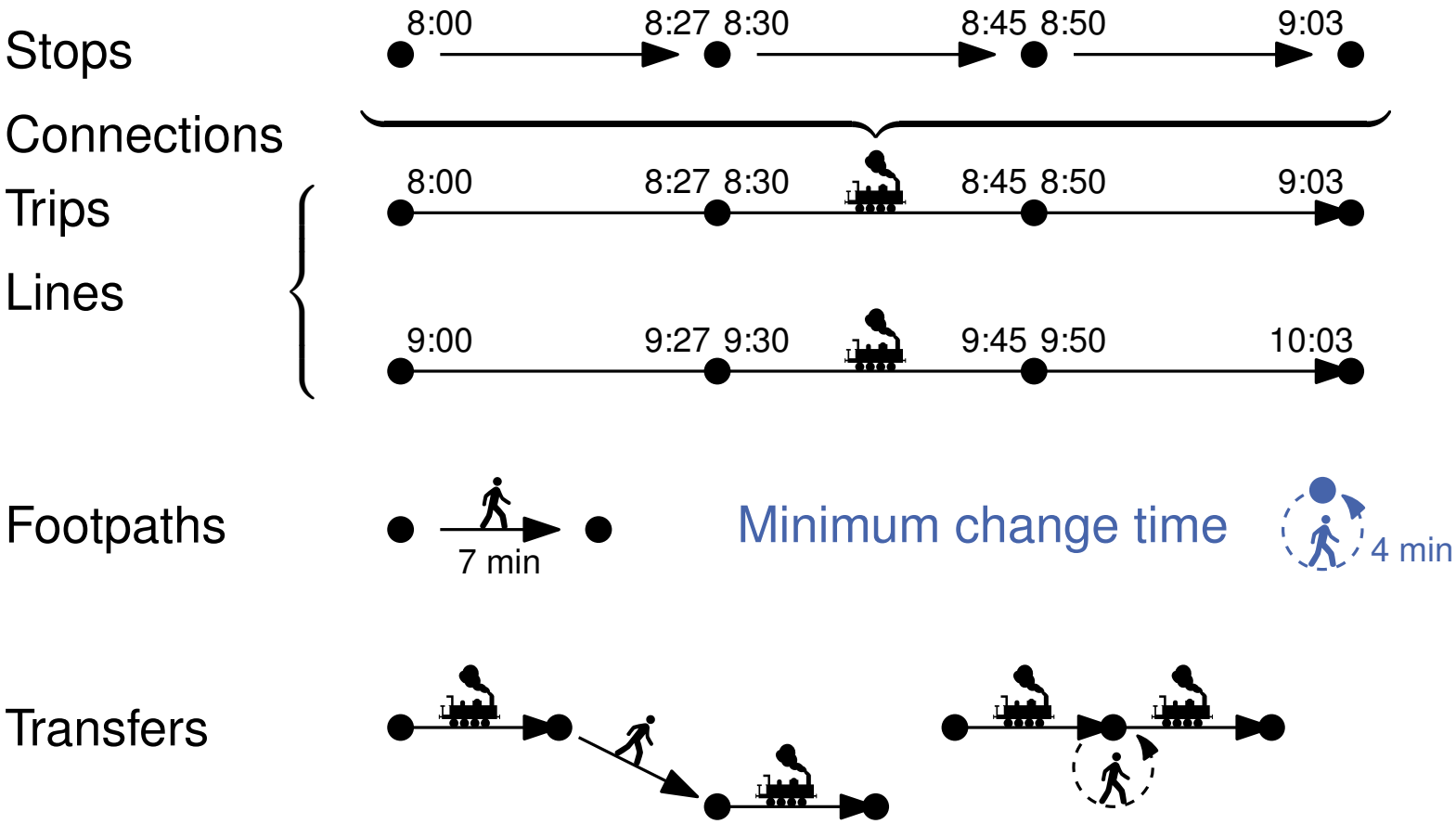


**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Introduction
## Terminology



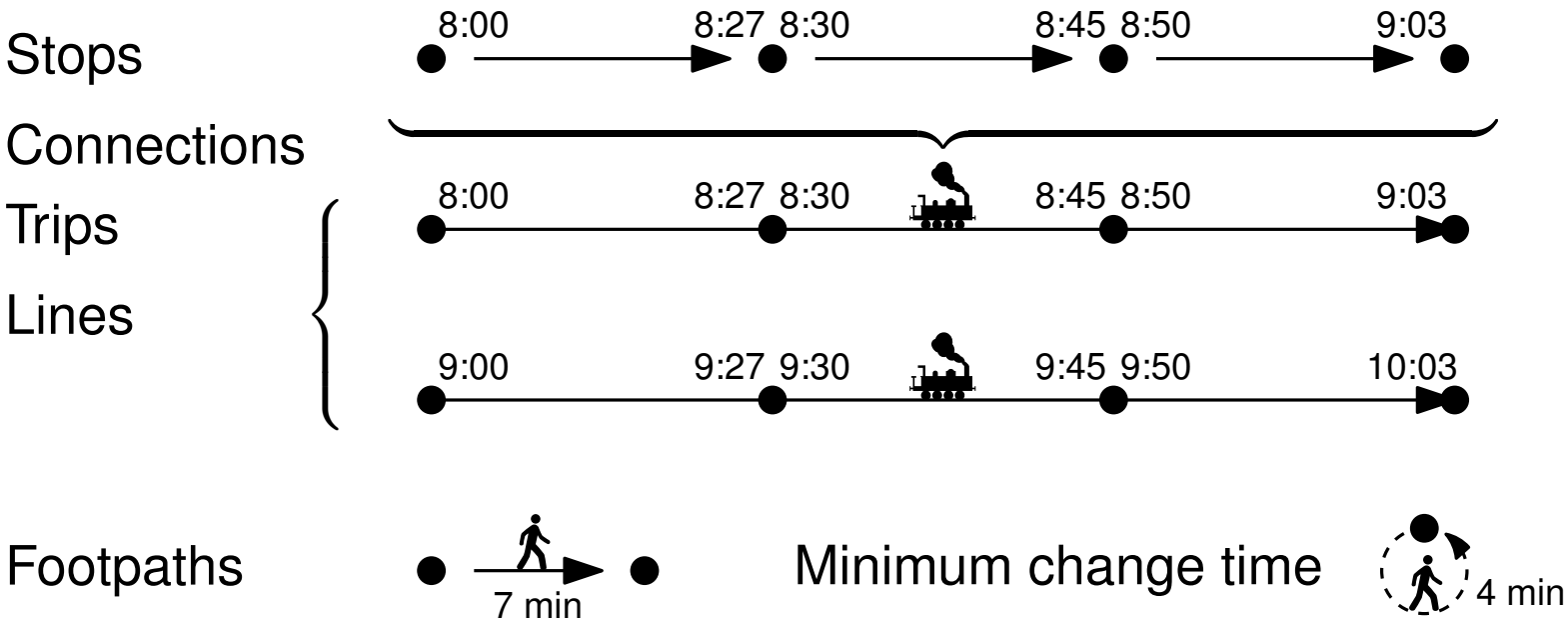
# Introduction
## Terminology





**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Introduction
## Terminology



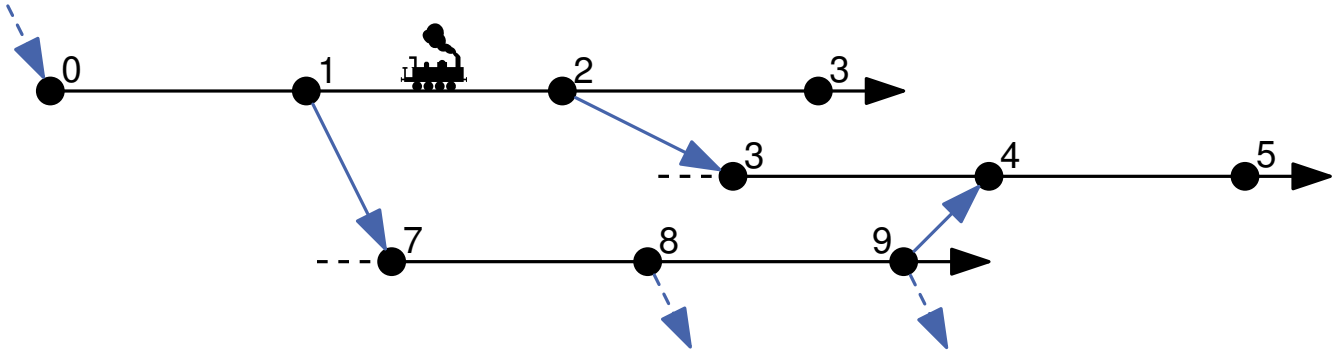


**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Trip-Based Routing

## Intuition

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



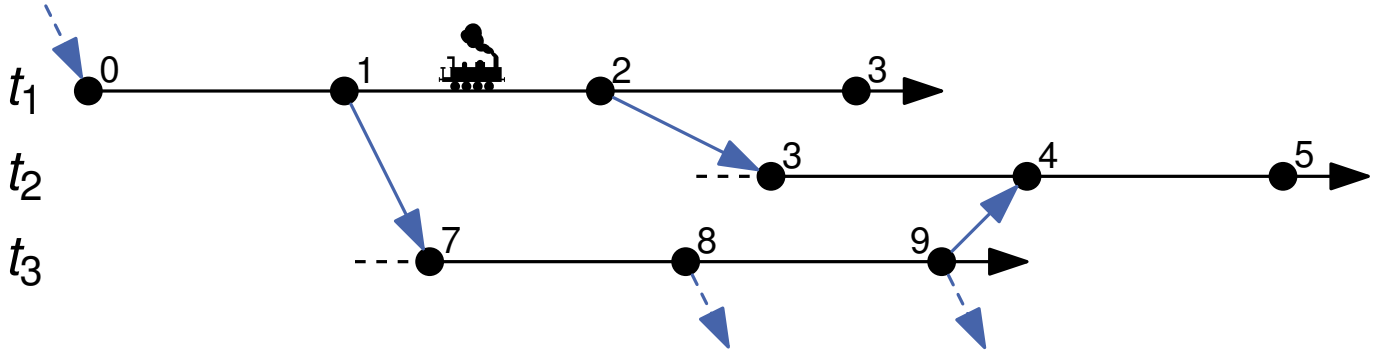


**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

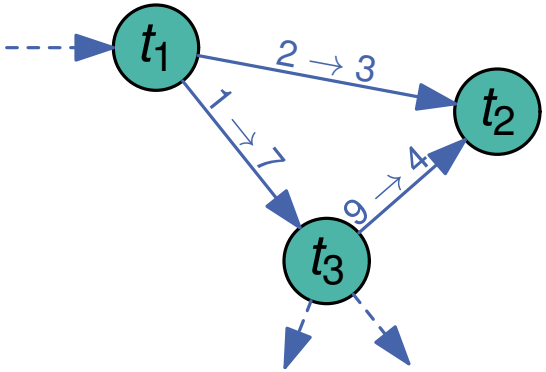
# Trip-Based Routing

## Intuition

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




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



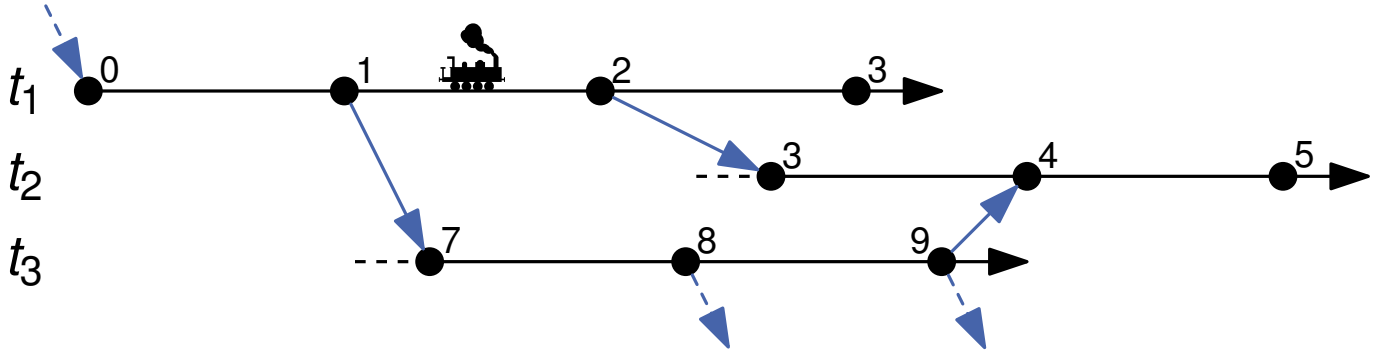


**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

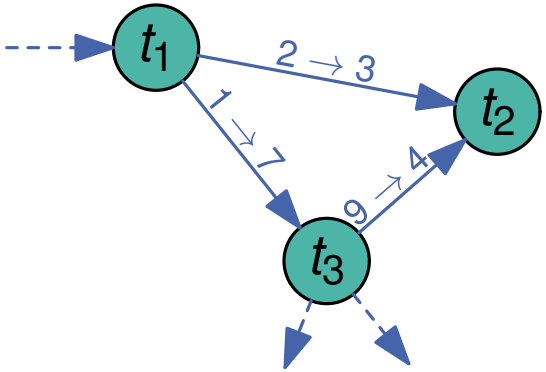
# Trip-Based Routing

## Intuition

- 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

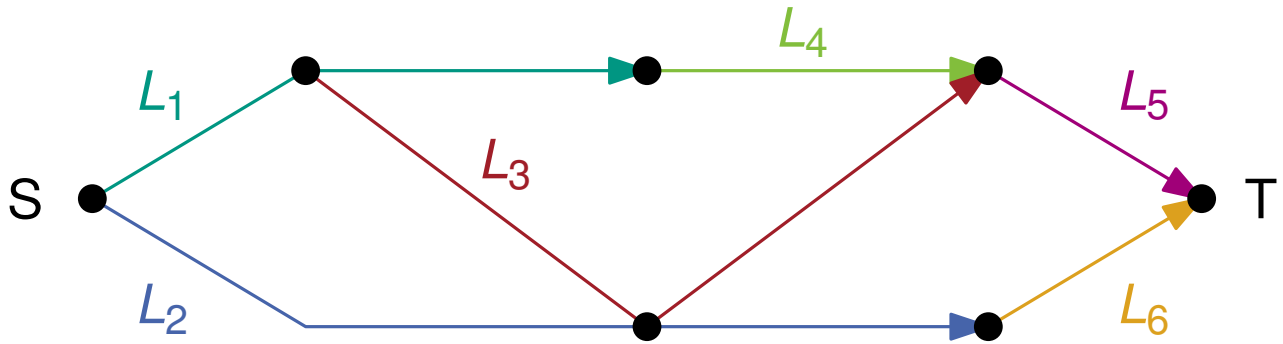




**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Trip-Based Routing

## Example




| $L_1$ | | |
|---|---|---|
| 8:00 | 8:30 | 9:00 |
| ⋮ | ⋮ | ⋮ |

| $L_3$ | | |
|---|---|---|
| 8:25 | 8:50 | 9:20 |
| 8:35 | 9:00 | 9:30 |
| ⋮ | ⋮ | ⋮ |

| $L_5$ | |
|---|---|
| 9:25 | 9:50 |
| 9:35 | 10:00 |
| ⋮ | ⋮ |

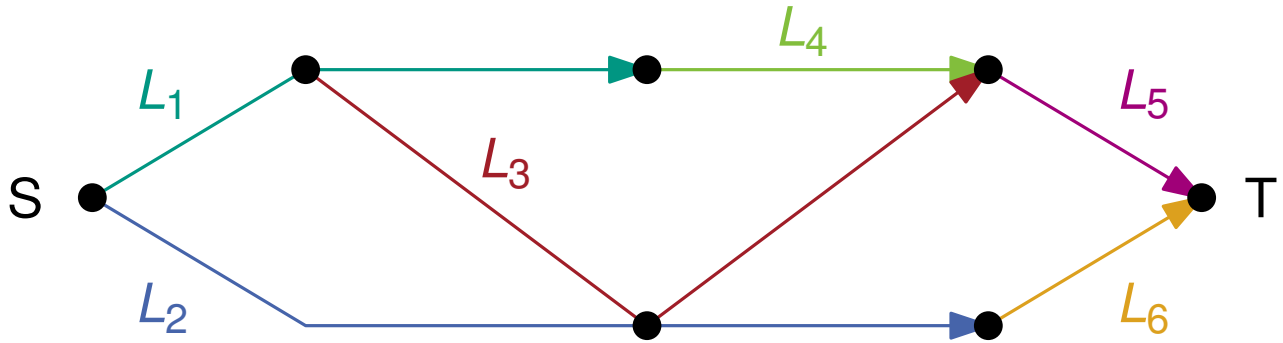
| $L_2$ | | |
|---|---|---|
| 7:30 | 8:45 | 9:30 |
| ⋮ | ⋮ | ⋮ |

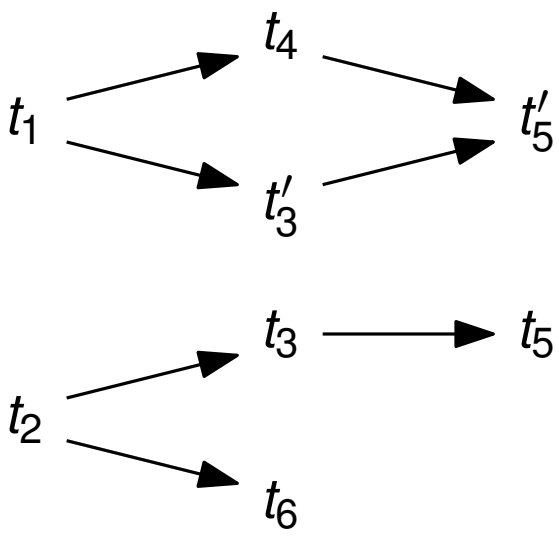
| $L_4$ | |
|---|---|
| 9:05 | 9:30 |
| ⋮ | ⋮ |

| $L_6$ | |
|---|---|
| 9:35 | 10:00 |
| ⋮ | ⋮ |

# Trip-Based Routing

## Example



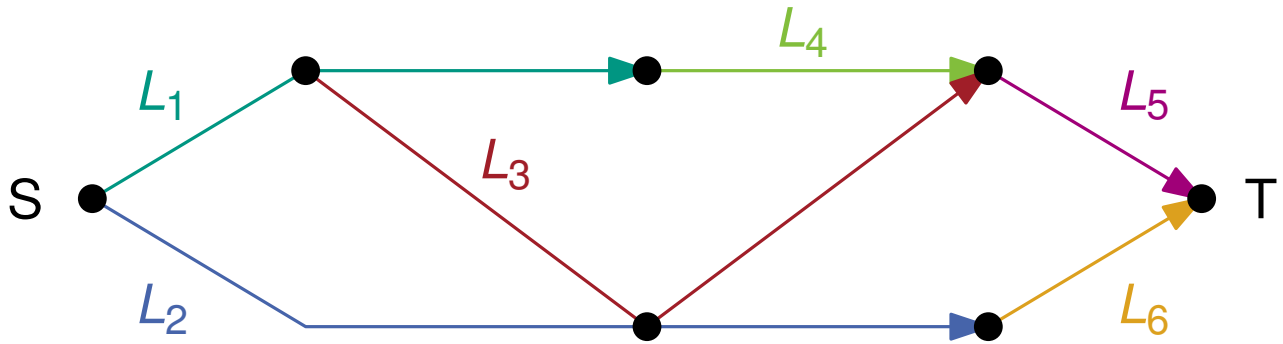



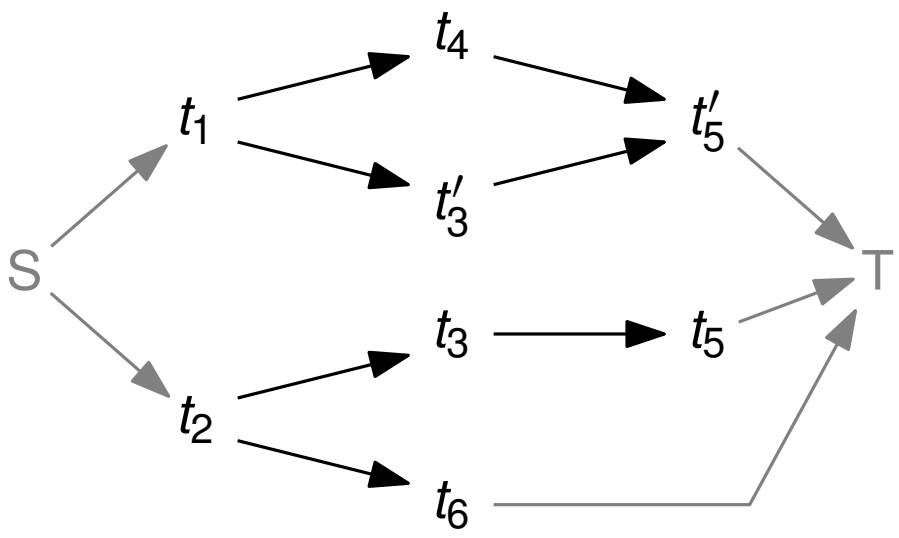


**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Trip-Based Routing

## Example



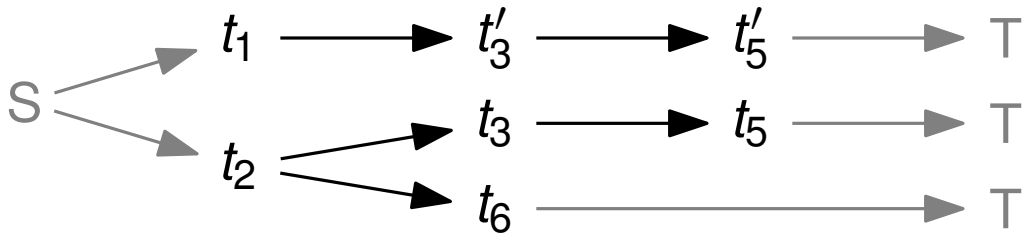





**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Trip-Based Routing

## Search Tree

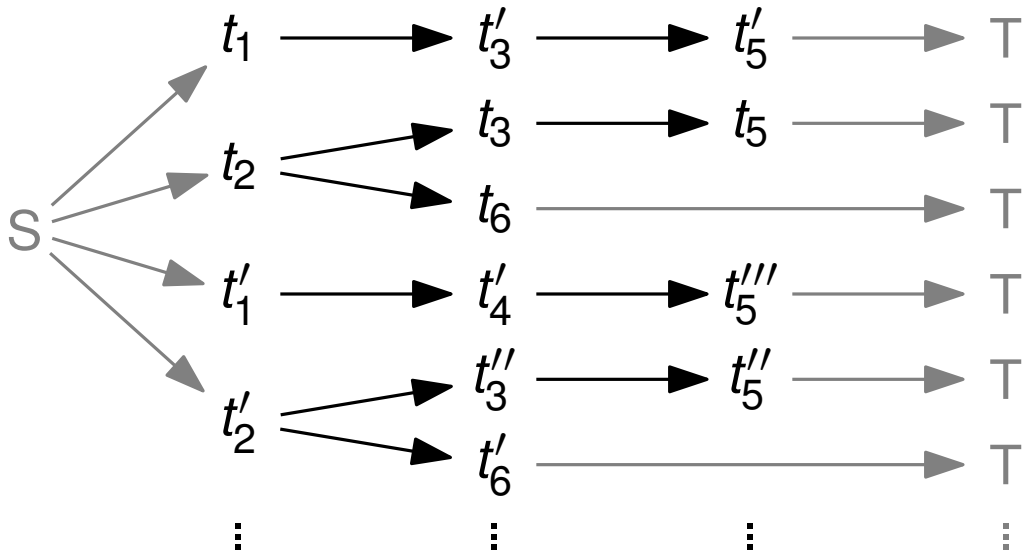




**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Trip-Based Routing

## Search Tree

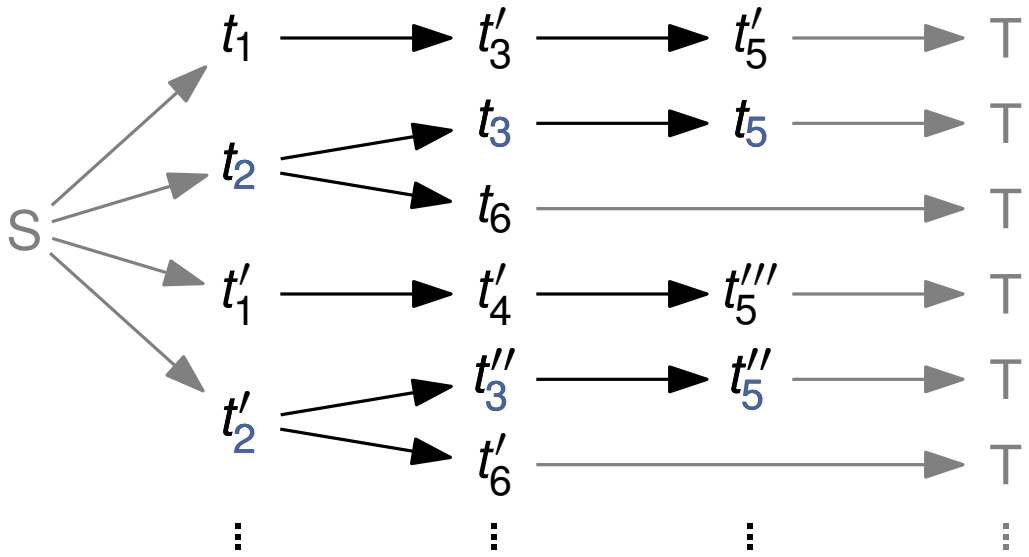




**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Trip-Based Routing

## Search Tree



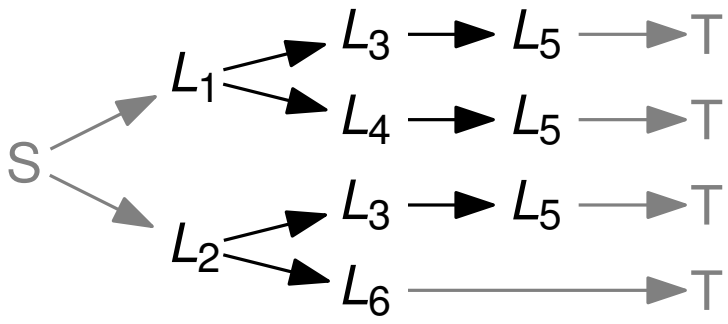

- Repeating patterns
- Can be used for goal-directed search


**Sascha Witt:**

**Institute of Theoretical Informatics**
Algorithmics

# Prefix Trees

## Computation

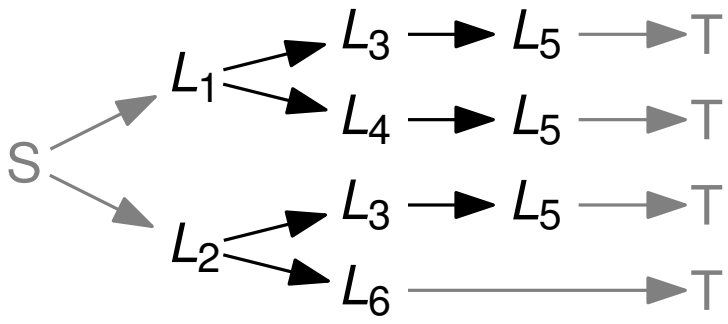




**Sascha Witt:**
Trip-Based Public Transit Routing Using Condensed Search Trees
**Institute of Theoretical Informatics**
Algorithmics

# Prefix Trees

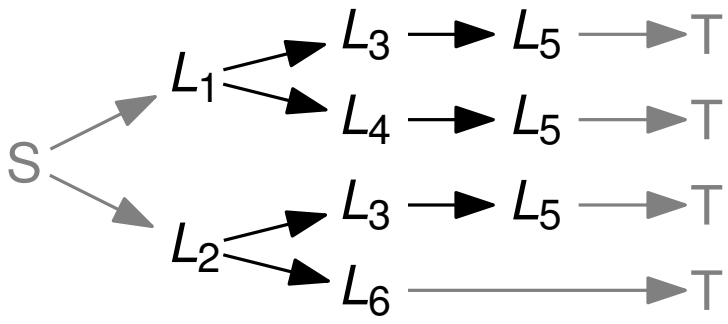
## Computation




- Root is source stop, leaves are destination stops
- Each path from S to T forms a sequence of lines that is optimal at some time
- All optimal journeys are covered
- Computed by performing one-to-all profile queries

# Prefix Trees

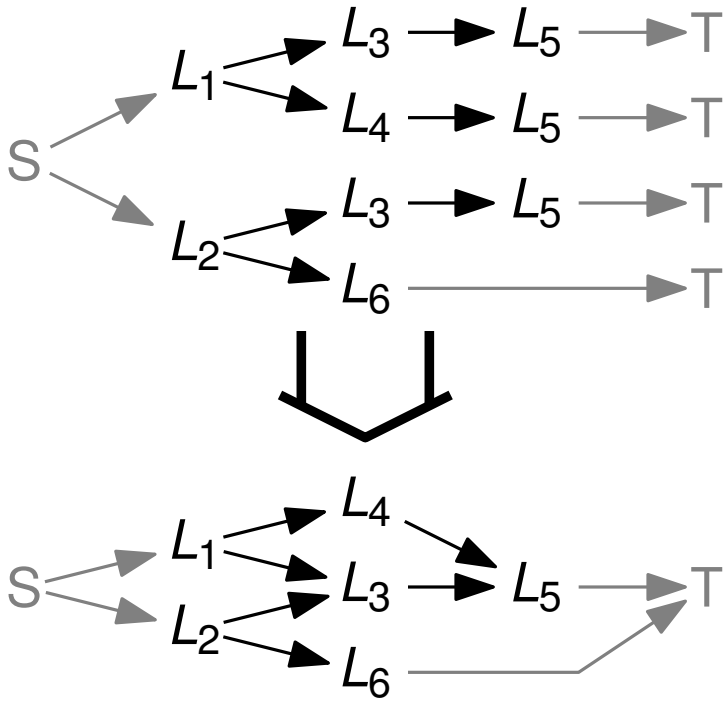
## Query




- Find all leaves labeled **T**

# Prefix Trees

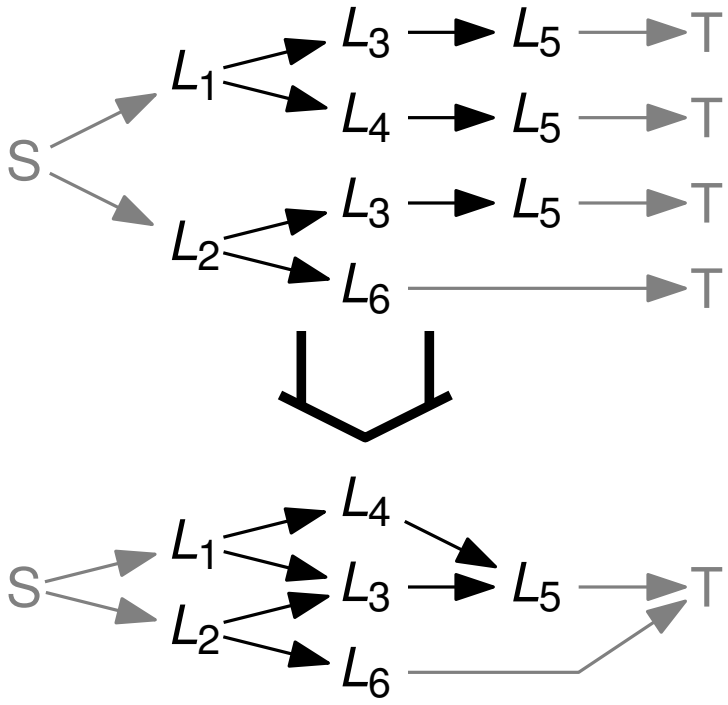
## Query




- Find all leaves labeled **T**
- Collapse paths into graph

# Prefix Trees

## Query




- Find all leaves labeled **T**
- Collapse paths into graph
- Perform query using this graph

# Prefix Trees

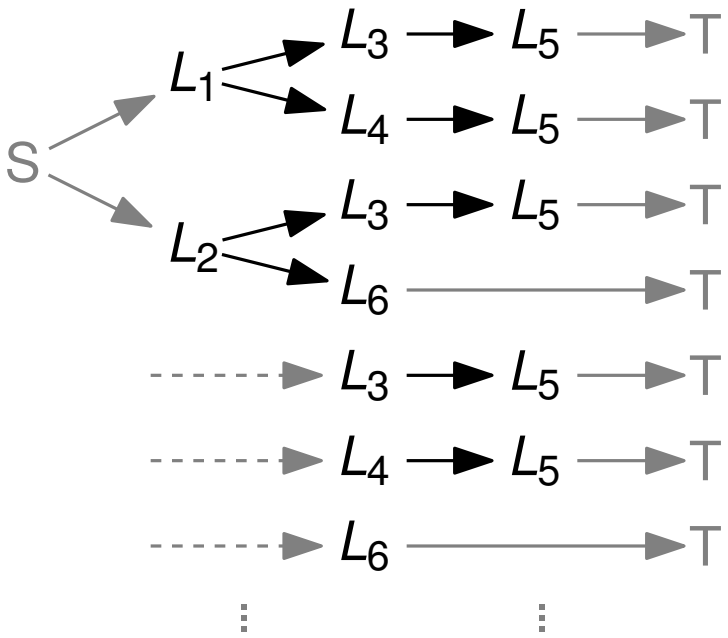
## Problems

- Large memory consumption
- Each tree covers the whole network

# Prefix Trees

## Problems

- Large memory consumption
- Each tree covers the whole network
- A lot of redundancy, especially near the leaves



# Splitting Trees

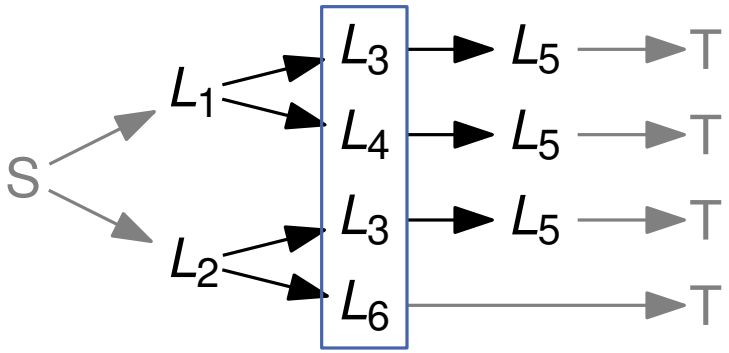
## Postfix Trees

- Solution: Postfix trees
- Can be constructed from prefix trees by cutting off branches

# Splitting Trees

## Postfix Trees

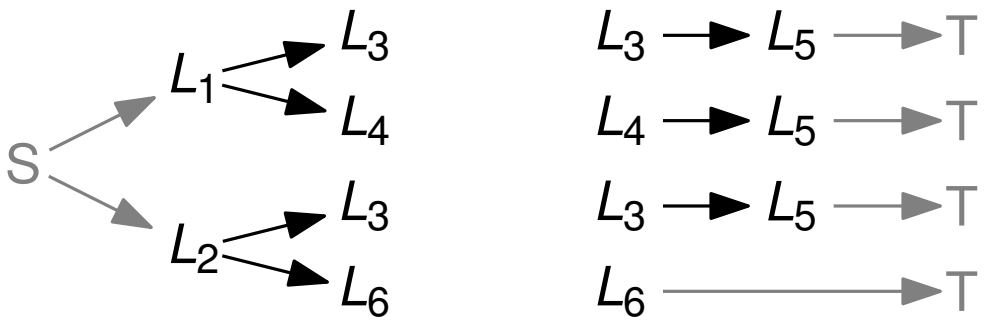
- Solution: Postfix trees
- Can be constructed from prefix trees by cutting off branches



# Splitting Trees

## Postfix Trees

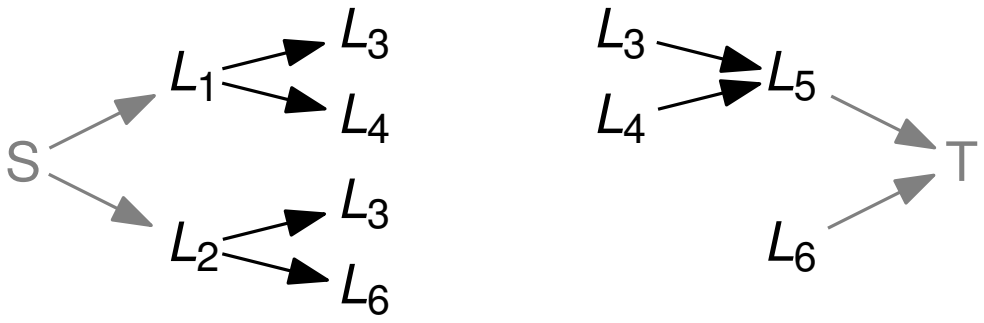
- Solution: Postfix trees
- Can be constructed from prefix trees by cutting off branches



# Splitting Trees

## Postfix Trees

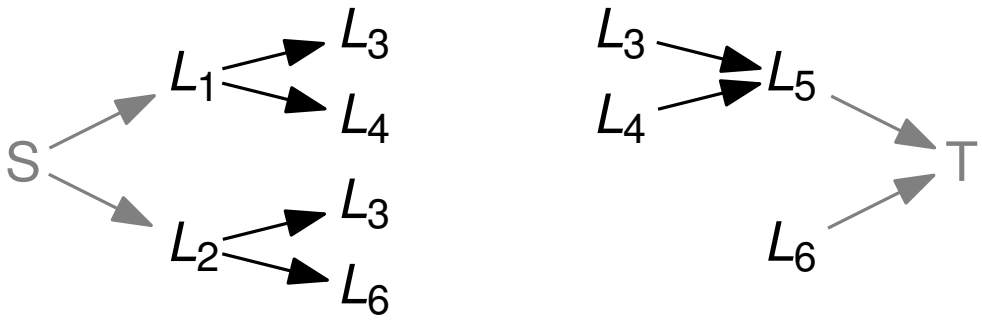
- Solution: Postfix trees
- Can be constructed from prefix trees by cutting off branches



# Splitting Trees

## Postfix Trees

- Solution: Postfix trees
- Can be constructed from prefix trees by cutting off branches

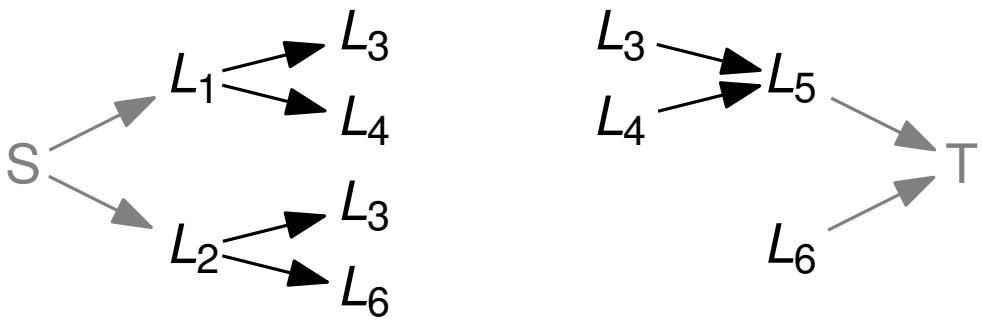



- Need to store two trees for each stop
- But these trees are much smaller

# Splitting Trees

## Query

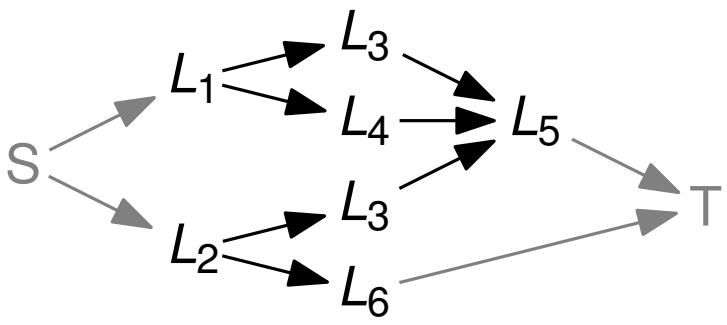
- Join trees at matching nodes



# Splitting Trees

## Query

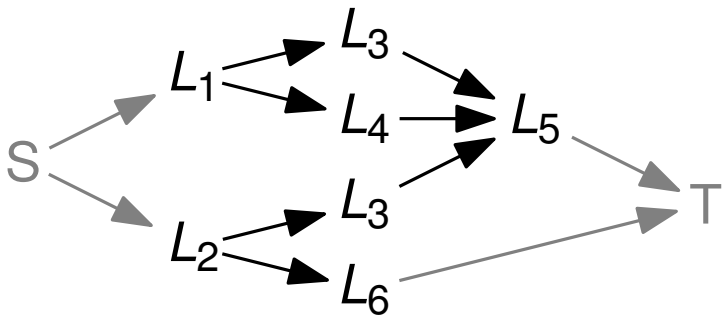
- Join trees at matching nodes



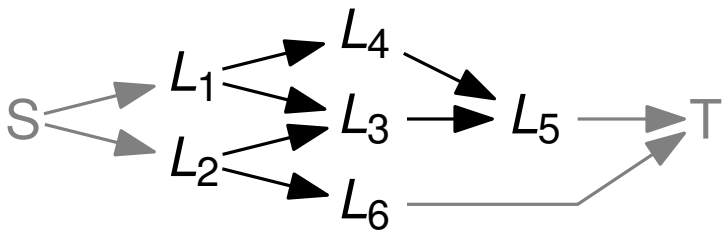
# Splitting Trees

## Query

- Join trees at matching nodes



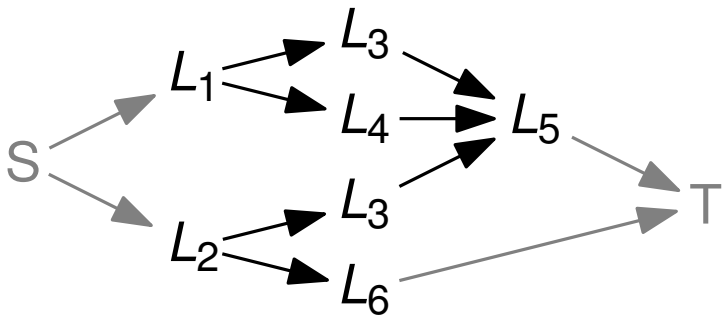

- Build query graph as before



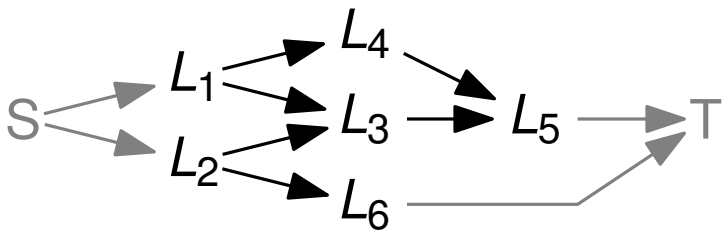
# Splitting Trees

## Query

- Join trees at matching nodes




- Build query graph as before




- Run query as before

# Splitting Trees

## Cut Selection

- Cut location is important
- Want to balance prefix and postfix trees, to minimize total size

# Splitting Trees

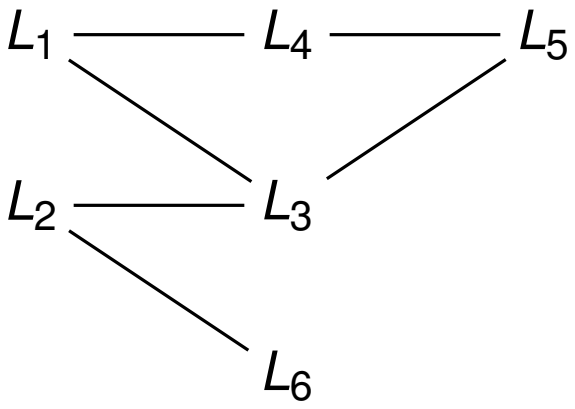
## Cut Selection

- Cut location is important
- Want to balance prefix and postfix trees, to minimize total size
- Our approach: Select ”important” lines
- Intuitively: ICE/TGV $>$ other train $>$ bus

# Splitting Trees

## Cut Selection

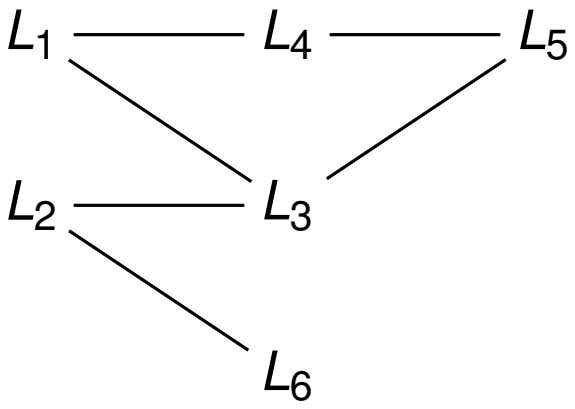
- Cut location is important
- Want to balance prefix and postfix trees, to minimize total size
- Our approach: Select ”important” lines
- Intuitively: ICE/TGV $>$ other train $>$ bus
- Use betweenness centrality on line graph as a measure of importance



# Splitting Trees

## Cut Selection

- Cut location is important
- Want to balance prefix and postfix trees, to minimize total size
- Our approach: Select ”important” lines
- Intuitively: ICE/TGV $>$ other train $>$ bus
- Use betweenness centrality on line graph as a measure of importance

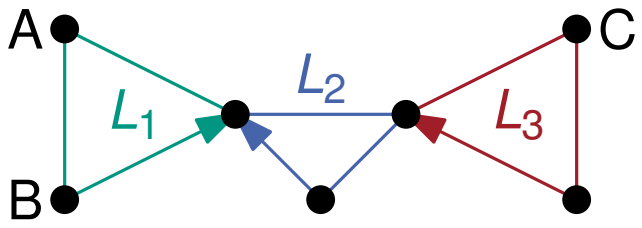



- As a result, trees reach out to long-distance lines

# Splitting Trees

## There And Back Again

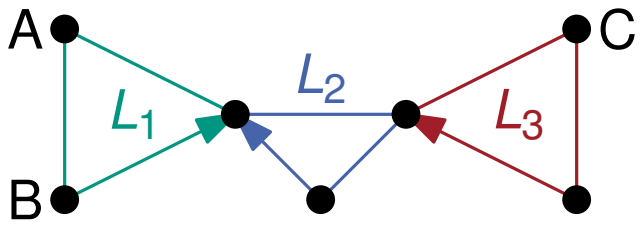
- Consider the following graph:



# Splitting Trees

## There And Back Again

- Consider the following graph:




- $L_2$ is the most central line
- Prefix and postfix trees for A and B look like this:

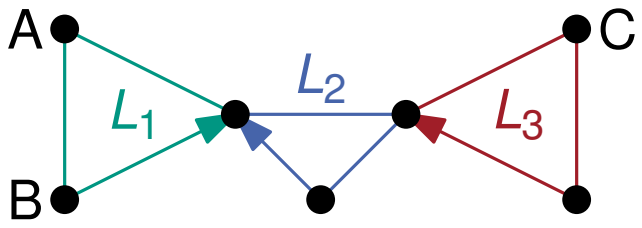
A → $L_1$ → $L_2$

$L_2$ → $L_1$ → B

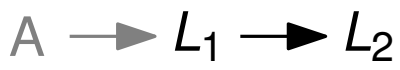
# Splitting Trees

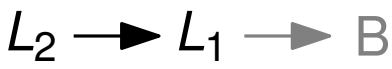
## There And Back Again

- Consider the following graph:

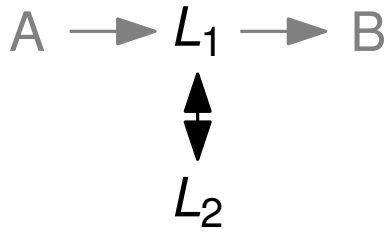



- $L_2$ is the most central line
- Prefix and postfix trees for A and B look like this:



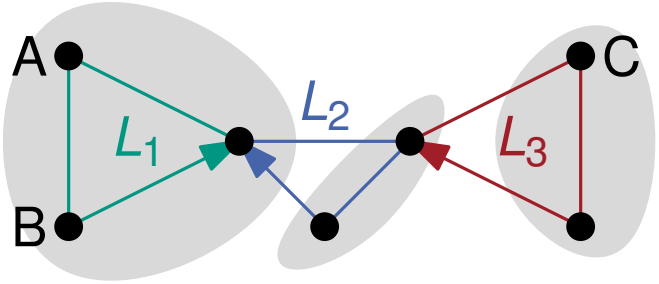

- Joining them results in this query graph:



# Splitting Trees

## There And Back Again

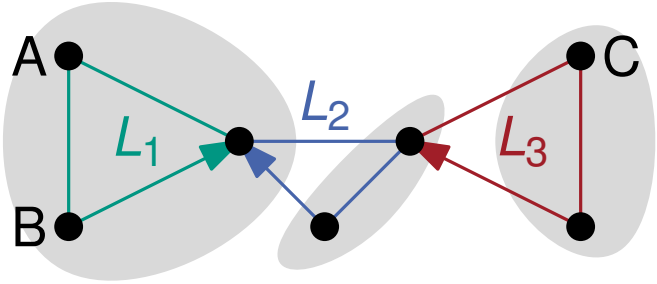
- Solution: Partition stops



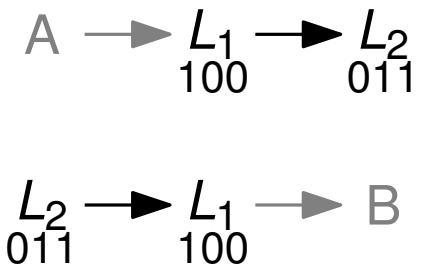
# Splitting Trees

## There And Back Again

- Solution: Partition stops



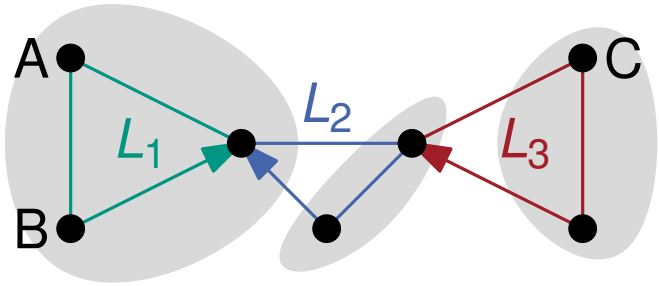

- Use bitmasks to indicate which nodes should be considered for the query graph



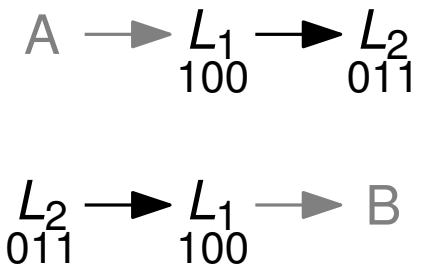
# Splitting Trees

## There And Back Again

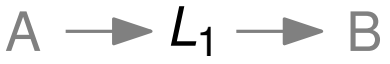
- Solution: Partition stops




- Use bitmasks to indicate which nodes should be considered for the query graph




- Result: Fewer false positives



# Experiments

## Instances

| Instance | Stops | Conn. | Trips | Lines | Footp. | Transfers |
|---|---|---|---|---|---|---|
| Germany | 296.6 k | 27,062 k | 1,432 k | 192.9 k | 102.8 k | 84,953 k |
| Sweden | 50.7 k | 6,054 k | 261 k | 17.6 k | 0.8 k | 16,455 k |
| Switzerland | 27.8 k | 4,650 k | 611 k | 14.4 k | 34.3 k | 12,626 k |
| London | 20.8 k | 4,991 k | 129 k | 2.2 k | 27.6 k | 15,883 k |
| Madrid | 4.6 k | 5,280 k | 190 k | 1.4 k | 1.4 k | 9,256 k |

# Experiments

## Instances

| Instance | Stops | Conn. | Trips | Lines | Footp. | Transfers |
|---|---|---|---|---|---|---|
| Germany | 296.6 k | 27,062 k | 1,432 k | 192.9 k | 102.8 k | 84,953 k |
| Sweden | 50.7 k | 6,054 k | 261 k | 17.6 k | 0.8 k | 16,455 k |
| Switzerland | 27.8 k | 4,650 k | 611 k | 14.4 k | 34.3 k | 12,626 k |
| London | 20.8 k | 4,991 k | 129 k | 2.2 k | 27.6 k | 15,883 k |
| Madrid | 4.6 k | 5,280 k | 190 k | 1.4 k | 1.4 k | 9,256 k |

| Instance | p. prefix tree [ms] | seq. [h:m] | par. [h:m] | speed up | avg. # of nodes | mem. [GB] |
|---|---|---|---|---|---|---|
| Germany | 2,143.6 | (231:16) | 13:48 | (16.8 x) | 6,131 | 23.2 |
| Sweden | 166.7 | 4:33 | 0:18 | 15.2 x | 2,433 | 1.6 |
| Switzerland | 209.3 | 3:18 | 0:12 | 16.5 x | 4,315 | 1.6 |
| London | 1,368.1 | 15:19 | 0:42 | 21.9 x | 20,390 | 6.0 |
| Madrid | 497.3 | 1:22 | 0:04 | 17.0 x | 32,293 | 2.0 |

* Quad 8-core Intel Xeon E5-4640, 2.4 GHz, 512 GB DDR3-1600, 64 threads

# Experiments

## Queries

| Instance | Var. | Query graph size [N+E] | Query graph time [µs] | EA [µs] | profile [µs] |
|---|---|---|---|---|---|
| Germany | TB | — | — | 30,856 | 192,952 |
| Sweden | TB | — | — | 2,760 | 16,532 |
| Switzerland | TB | — | — | 1,780 | 18,104 |
| London | TB | — | — | 1,374 | 96,114 |
| Madrid | TB | — | — | 711 | 54,118 |
| Germany | PT | $41 + 58$ | 994.4 | 63.3 | 155.0 |
| Sweden | PT | $23 + 32$ | 24.6 | 40.4 | 88.6 |
| Switzerland | PT | $38 + 59$ | 34.0 | 45.8 | 155.9 |
| London | PT | $91 + 196$ | 138.2 | 101.1 | 2,786.6 |
| Madrid | PT | $150 + 407$ | 306.9 | 81.7 | 6,913.8 |
| Germany | ST | $124 + 232$ | 81.1 | 75.0 | 430.5 |
| Sweden | ST | $66 + 122$ | 32.5 | 27.2 | 207.1 |
| Switzerland | ST | $118 + 233$ | 76.1 | 32.7 | 327.6 |
| London | ST | $331 + 1{,}242$ | 1,583.3 | 141.4 | 14,545.4 |
| Madrid | ST | $456 + 2{,}073$ | 11,822.9 | 165.8 | 28,919.0 |

# Experiments

## Comparison

| algorithm | instance | stops [$10^3$] | conn. [$10^6$] | transfers | profile | mem. [GB] | pre. [h] | query [µs] |
|---|---|---|---|---|---|---|---|---|
| CSA | Germany | 252.4 | 46.2 | ○ | ○ | — | — | 298.6 k |
| ACSA | Germany | 252.4 | 46.2 | ○ | ○ | n/a | 0.2 | 8.7 k |
| TP | Germany | 248.4 | 13.9 | ● | ○ | 140.0 | 372.0 | 300.0 |
| Sc-TP | Germany | 250.0 | 15.0 | ● | ○ | 1.2 | 16.5 | 32.0 k |
| TB | Germany | 296.6 | 27.1 | ● | ○ | 23.2 | 231.3 | 156.1 |
| TTL | Sweden | 51.4 | n/a | ○ | ○ | ≈ 0.5 | 0.2 | ≈ 10.0 |
| PTL | Sweden | 51.1 | 12.7 | ● | ○ | 12.3 | 36.2 | 27.6 |
| TB | Sweden | 50.7 | 6.1 | ● | ○ | 1.6 | 3.8 | 59.7 |
| PTL | Switzerland | 27.1 | 23.7 | ● | ○ | 12.7 | 61.6 | 21.7 |
| TB | Switzerland | 27.8 | 4.7 | ● | ○ | 1.6 | 2.7 | 108.8 |
| CSA | London | 20.8 | 4.9 | ○ | ○ | — | — | 1.8 k |
| PTL | London | 20.8 | 5.1 | ● | ○ | 26.2 | 49.3 | 30.0 |
| TB | London | 20.8 | 5.0 | ● | ○ | 6.0 | 11.6 | 1.7 k |
| TTL | Madrid | 4.6 | n/a | ○ | ○ | ≈ 0.4 | 0.1 | ≈ 30.0 |
| PTL | Madrid | 4.7 | 4.5 | ● | ○ | 9.9 | 10.9 | 64.3 |
| TP | Madrid | 4.6 | 4.8 | ● | ○ | n/a | 185.0 | 3.1 k |
| TB | Madrid | 4.6 | 5.3 | ● | ○ | 2.0 | 1.1 | 12.0 k |

# Experiments

## Comparison

| algorithm | instance | stops [$10^3$] | conn. [$10^6$] | transfers | profile | mem. [GB] | pre. [h] | query [µs] |
|---|---|---|---|---|---|---|---|---|
| ACSA | Germany | 252.4 | 46.2 | ○ | ● | n/a | 0.2 | 171.0 k |
| TP | Germany | 248.4 | 13.9 | ● | ● | 140.0 | 372.0 | 5.0 k |
| TB | Germany | 296.6 | 27.1 | ● | ● | 23.2 | 231.3 | 511.6 |
| PTL | Sweden | 51.1 | 12.7 | ○ | ● | 0.7 | 0.5 | 12.1 |
| TB | Sweden | 50.7 | 6.1 | ● | ● | 1.6 | 3.8 | 239.6 |
| PTL | Switzerland | 27.1 | 23.7 | ○ | ● | 0.7 | 0.7 | 24.5 |
| TB | Switzerland | 27.8 | 4.7 | ● | ● | 1.6 | 2.7 | 403.7 |
| PTL | London | 20.8 | 5.1 | ○ | ● | 1.3 | 0.9 | 74.3 |
| CSA | London | 20.8 | 4.9 | ● | ● | — | — | 466.0 k |
| TB | London | 20.8 | 5.0 | ● | ● | 6.0 | 11.6 | 16.1 k |
| PTL | Madrid | 4.7 | 4.5 | ○ | ● | 0.4 | 0.4 | 111.9 |
| TB | Madrid | 4.6 | 5.3 | ● | ● | 2.0 | 1.1 | 40.7 k |

# Conclusion

- Speed-up technique for trip-based public transit routing
- Exploits regularities in timetables
- Fast preprocessing of country-sized networks
- Pareto-optimal bi-criteria 24 h profile queries on microsecond scale

# Conclusion

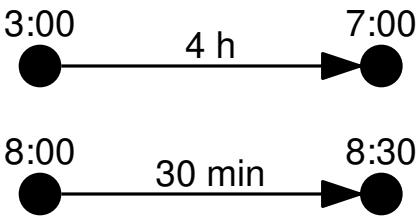
- Speed-up technique for trip-based public transit routing
- Exploits regularities in timetables
- Fast preprocessing of country-sized networks
- Pareto-optimal bi-criteria 24 h profile queries on microsecond scale

## Future Work

- Improve preprocessing time (and space) for very large networks
- Improve performance on metropolitan networks
- Better scaling to larger networks and longer timeframes

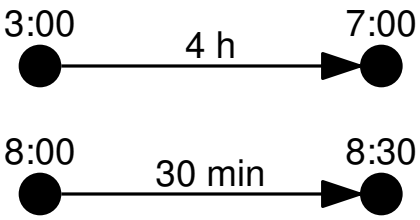
# Introduction
## Public Transit Routing

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

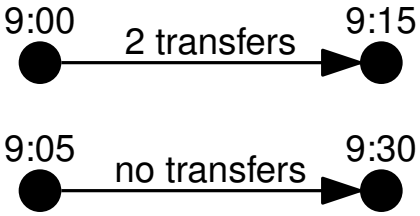


# Introduction
## Public Transit Routing

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



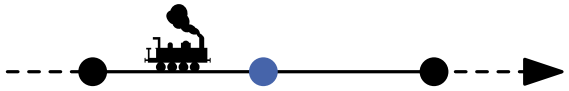

- Multiple natural problem variants
  - Earliest arrival queries
  - Profile (range) queries
  - Multi-criteria queries (e.g., number of transfers taken)



# Trip-Based Routing

## Preprocessing

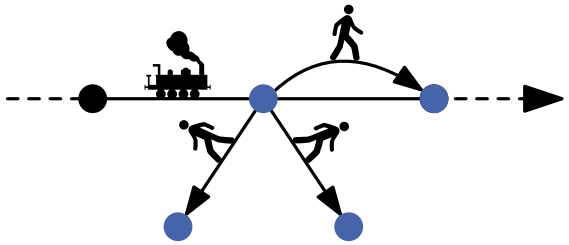
- Compute transfers between trips



# Trip-Based Routing

## Preprocessing

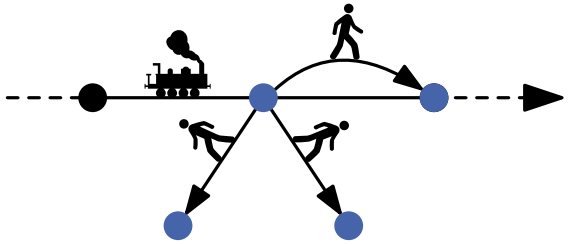
- Compute transfers between trips



# Trip-Based Routing

## Preprocessing

- Compute transfers between trips

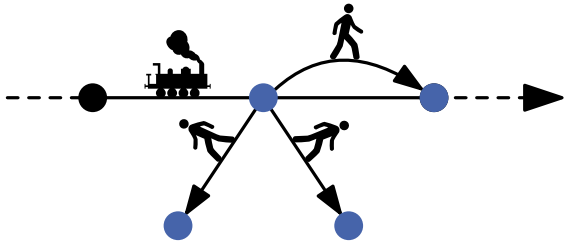


- For each line, find the first reachable trip
  $(\text{arrival time} + \text{footpath length} \leq \text{departure time})$

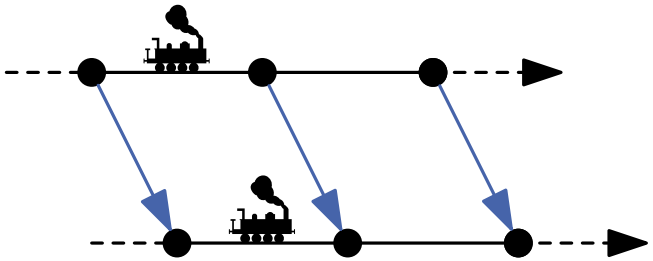
# Trip-Based Routing

## Preprocessing

- Compute transfers between trips



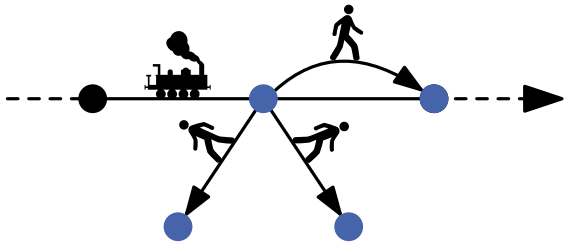
- For each line, find the first reachable trip
  (arrival time + footpath length $\leq$ departure time)
- Huge number of transfers, not all of which are useful



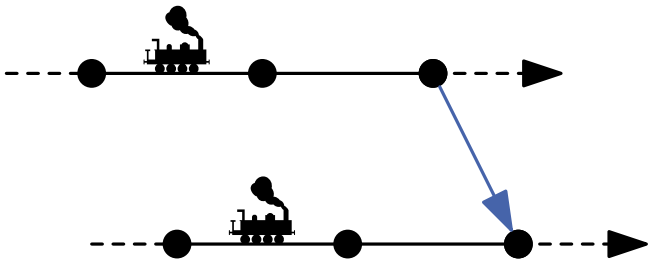
Parallel trips

# 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

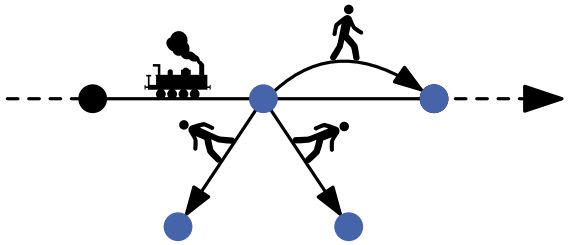


Parallel trips

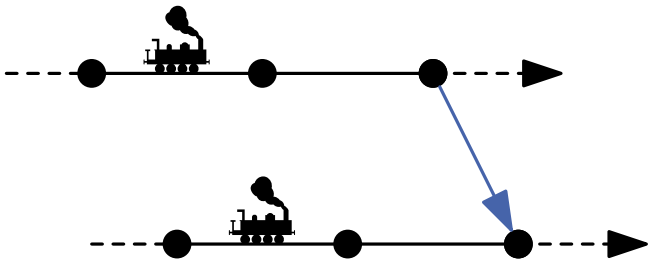
# Trip-Based Routing

## Preprocessing

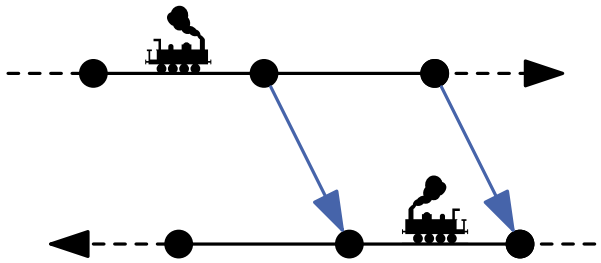
- Compute transfers between trips



- 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

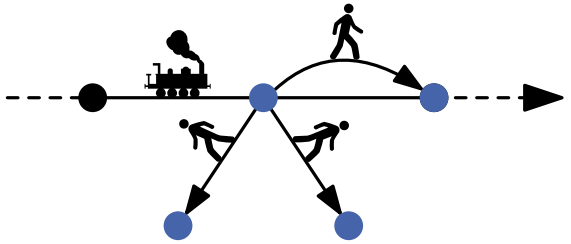


U-turns

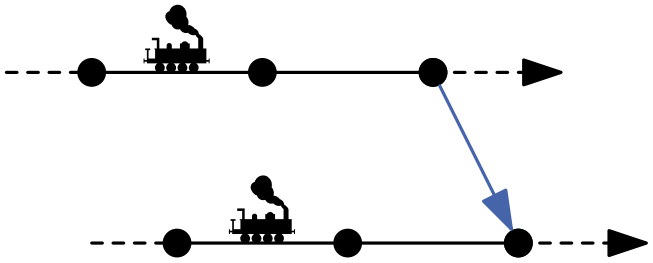
# Trip-Based Routing

## Preprocessing

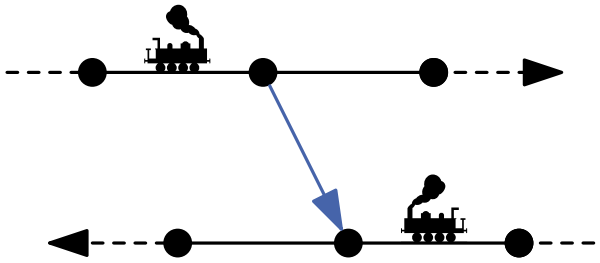
- Compute transfers between trips



- 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

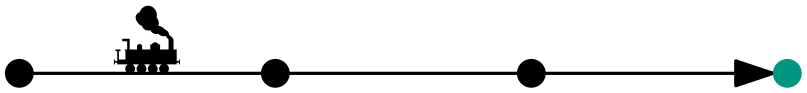
## Preprocessing

- Reduce number of transfers by eliminating redundant ones

# Trip-Based Routing

## Preprocessing

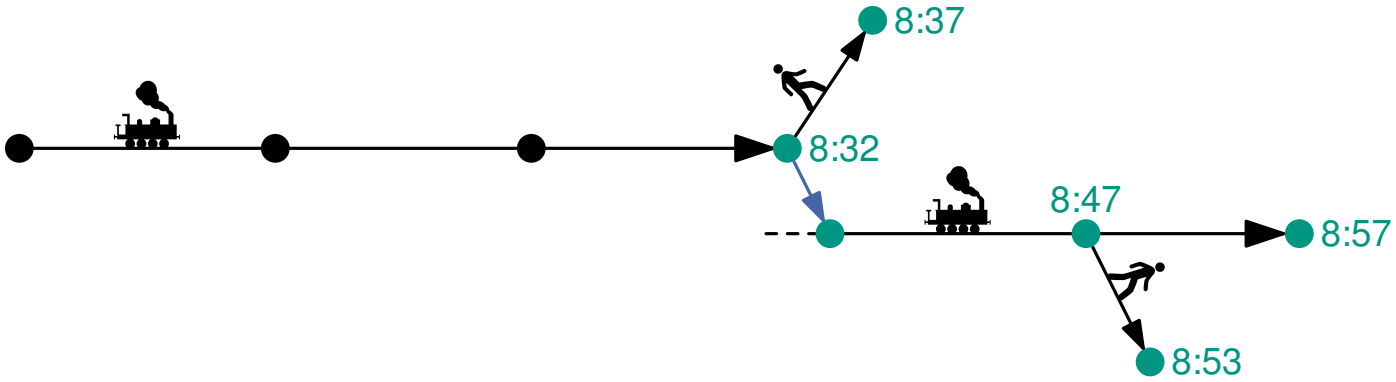
- Reduce number of transfers by eliminating redundant ones
- Process trips backwards



# Trip-Based Routing

## Preprocessing

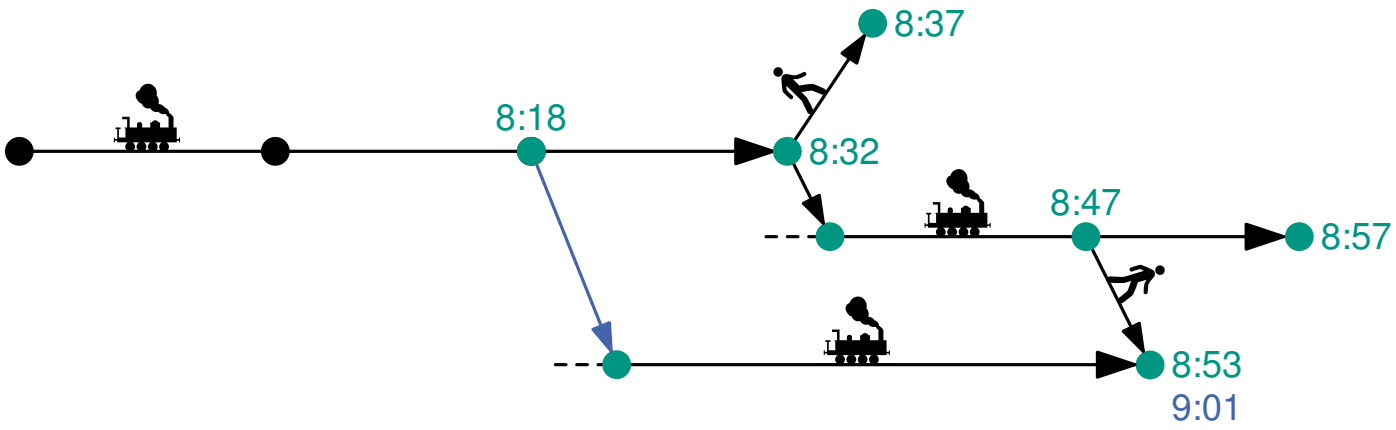
- Reduce number of transfers by eliminating redundant ones
- Process trips backwards
- Keep track of which stops can be reached at what time



# Trip-Based Routing

## Preprocessing

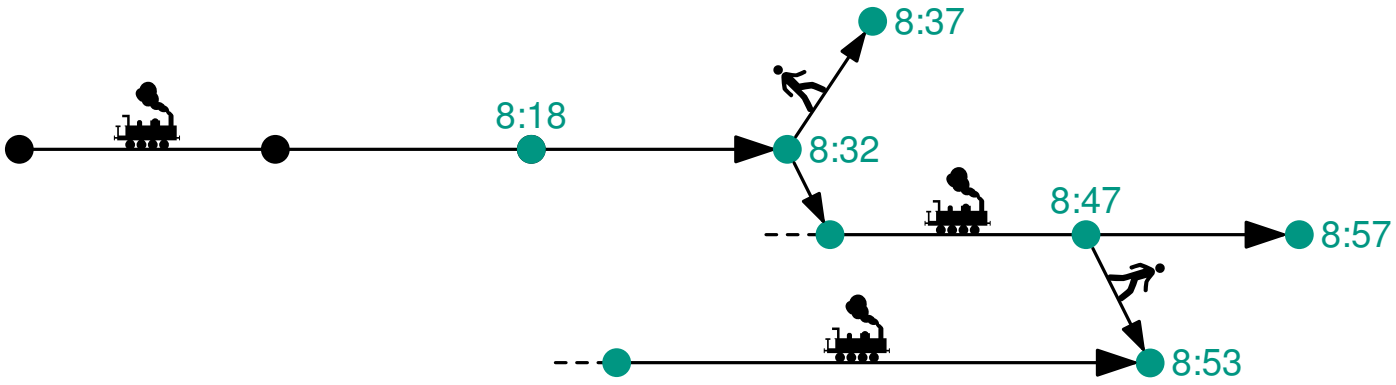
- Reduce number of transfers by eliminating redundant ones
- Process trips backwards
- Keep track of which stops can be reached at what time
- Evaluate transfers by checking if they improve arrival times



# 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

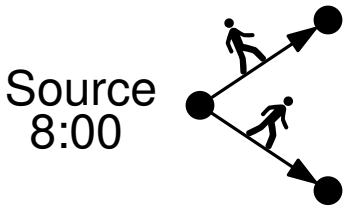
## Query

- Input: Source stop, target stop, departure time

# Trip-Based Routing

## Query

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



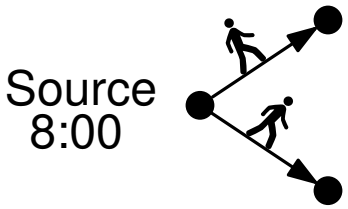


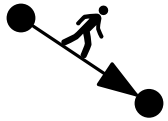
| dep. | line | trip | index |
|---|---|---|---|
| 8:00 | 2 | 15 | 8 |
| 8:03 | 4 | 56 | 0 |
| 8:07 | 11 | 456 | 31 |
| 9:00 | 110 | 3256 | 6 |

# 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 |
| ... | |

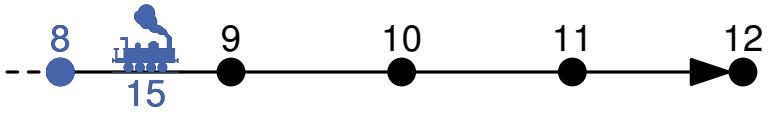
| dep. | line | trip | index |
|---|---|---|---|
| 8:00 | 2 | 15 | 8 |
| 8:03 | 4 | 56 | 0 |
| 8:07 | 11 | 456 | 31 |
| 9:00 | 110 | 3256 | 6 |

| 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 |

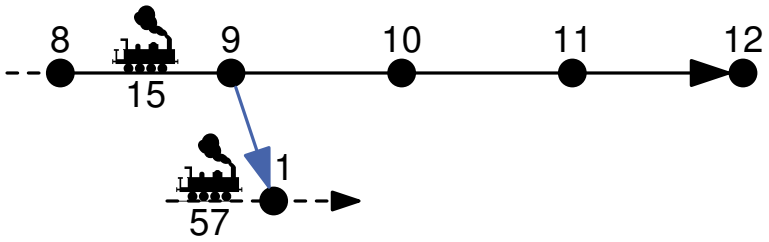
| Reached | |
|---|---|
| ... | |
| 15 | 8 |
| 16 | 8 |
| 17 | 8 |
| ... | |
| 56 | 0 |
| 57 | 0 |
| ... | |
| 456 | 31 |
| 457 | 31 |
| ... | |
| 3256 | 6 |
| ... | |

| 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 |

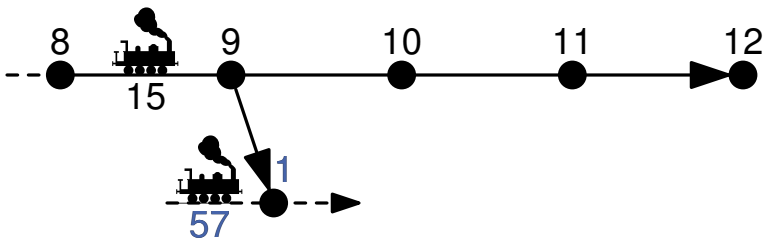
# Trip-Based Routing

## 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 |
| ... | |

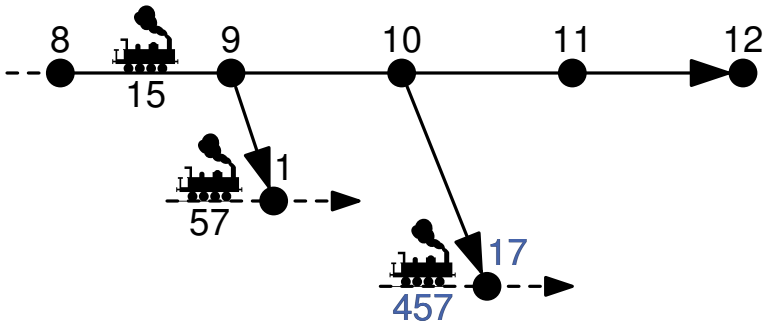



| line | index | footpath |
|---|---|---|
| 3 | 8 | — |
| 8 | 17 | 4 min |
| 27 | 3 | 4 min |

# 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 |

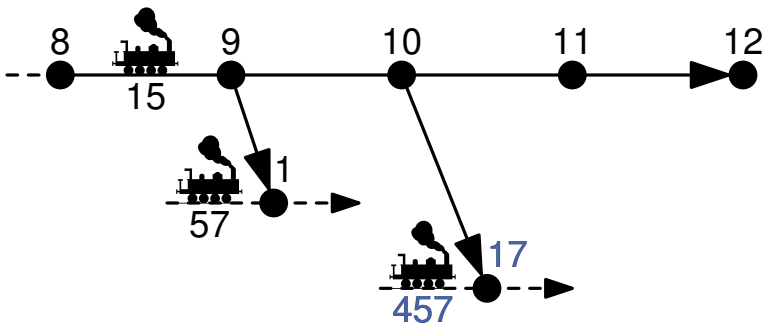
| Reached | |
|---|---|
| ... | |
| 15 | 8 |
| 16 | 8 |
| 17 | 8 |
| ... | |
| 56 | 0 |
| 57 | 0 |
| ... | |
| 456 | 31 |
| 457 | 31 |
| ... | |
| 3256 | 6 |
| ... | |

| 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 |

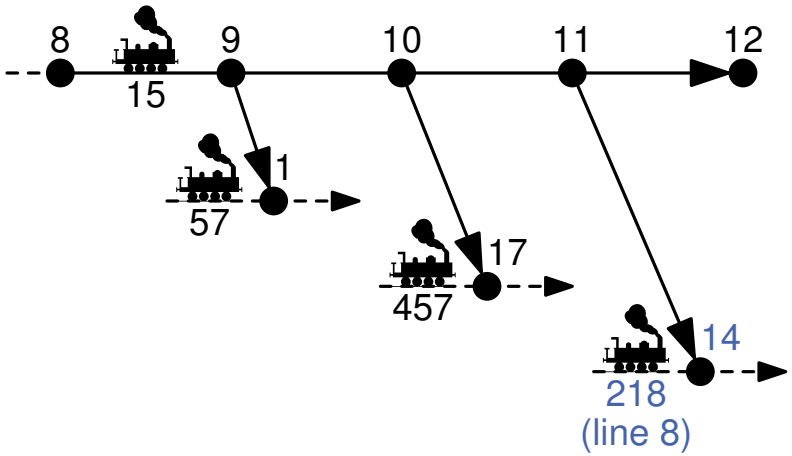
| Reached | |
|---|---|
| ... | |
| 15 | 8 |
| 16 | 8 |
| 17 | 8 |
| ... | |
| 56 | 0 |
| 57 | 0 |
| ... | |
| 456 | 31 |
| 457 | 17 |
| 458 | 17 |
| ... | |
| 3256 | 6 |
| ... | |

| 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 |

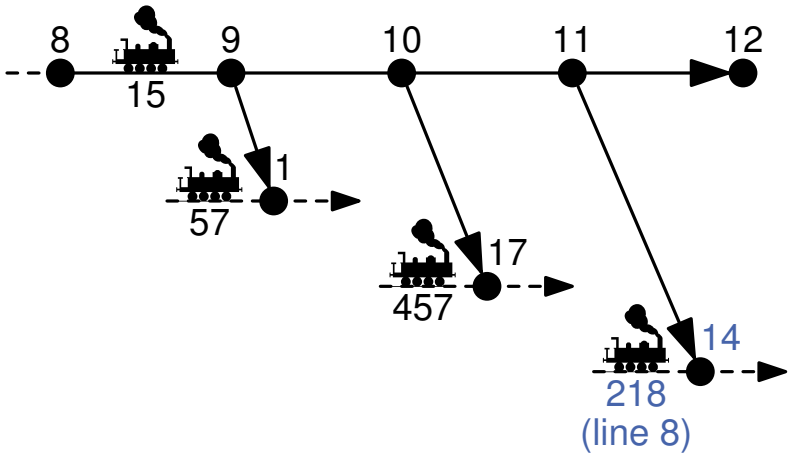
| Reached | |
|---|---|
| ... | |
| 15 | 8 |
| 16 | 8 |
| 17 | 8 |
| ... | |
| 56 | 0 |
| 57 | 0 |
| ... | |
| 218 | 14 |
| ... | |
| 456 | 31 |
| 457 | 17 |
| 458 | 17 |
| ... | |
| 3256 | 6 |
| ... | |

| 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 |

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

# 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