Mathematical Aspects of Public Transportation Networks

Niels Lindner

May 7, 2018
Chapter 2

Shortest Routes in Public Transportation Networks

§2.2 Graph Methods
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönaw @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönnow @ 07:03 → all stations, without minimum change times
Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
**Time-Dependent Dijkstra**

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
Example (shortest path tree)

Query: Hönnow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönnow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönnow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change time
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
§2.2 Graph Methods

Time-Dependent Dijkstra

Example (shortest path tree)

Query: Hönow @ 07:03 → all stations, without minimum change times
Chapter 2

Shortest Routes in Public Transportation Networks

§2.3 Advanced Methods
Topological orders

Let $G = (V, E)$ be a digraph.

**Definition**

A **topological order** of $G$ is a total order $\leq$ on $V$ such that for each edge $(v, w) \in E$ holds $v < w$.

**Lemma**

$G$ has a topological order if and only if $G$ contains no directed circuits.

**Proof.**

($\Rightarrow$) Let $\leq$ be a topological order. If $(v_1, \ldots, v_k, v_1)$ is a directed circuit in $G$, then $v_1 < \cdots < v_k < v_1$, contradicting the reflexivity of $\leq$.

($\Leftarrow$) Induction on the number of vertices: If $|V| = 1$ and $G$ has no loops, then take $\{v \leq v\}$ as order. Now suppose $|V| > 1$. Since $G$ has no directed circuit, there is a vertex $v$ with out-degree 0. Removing $v$ yields a graph having a topological order $\leq$ by the induction hypothesis. Append $v$ to $\leq$ as largest element, i.e., $v > w$ for all $w \in V \setminus \{v\}$.  

May 7, 2018
2.3 Advanced Methods

Shortest Paths in DAGs

Definition
A directed acyclic graph (DAG) is a directed graph without directed circuits.

Example
Our (aperiodic) time expansions are DAGs if every driving activity has positive length.

Theorem
Let $G$ be a directed acyclic graph on $n$ vertices and $m$ edges. The single-source shortest path problem in $G$ – i.e., finding all shortest paths from a fixed source to all target vertices – can be solved in $O(n + m)$ time.

Observation
Let $\leq$ be a topological order on a DAG. If there is a directed path from $v$ to $w$, then $v < w$. 
Let $G = (V, E)$ be a DAG with an arbitrary length function $\ell : E \to \mathbb{R}$, and let $s \in V$ be a source vertex.

**DAG Single-Source Shortest Path Algorithm**

1. For all $v \in V$:
   - $\text{path}(v) := [s]$
   - $\text{distance}(v) := \begin{cases} 0 & \text{if } v = s, \\ \infty & \text{else} \end{cases}$

2. Compute a topological order $\leq$ on $V$.

3. For all $v \geq s$ sorted in ascending order w.r.t. $\leq$:
   - For all successors $w$ of $v$:
     - If $\text{distance}(v) + \ell(v, w) < \text{distance}(w)$:
       - $\text{distance}(w) := \text{distance}(v) + \ell(v, w)$
       - $\text{path}(w) := \text{path}(v) + [w]$.

4. Return $\text{path}$, $\text{distance}$. 
Shortest Paths in DAGs: Correctness

Claim
After \( v \) is scanned in Step 3, \( \text{distance}(v) = \text{length of a shortest } s-v \text{-path.} \)

Proof.

- This is clear for \( v = s \).
- Suppose \( v > s \). Let \( p = (s, u_1, \ldots, u_k, v) \) be a shortest \( s-v \)-path.
- Since \( \text{distance}(v) \) is clearly the length of the path given by \( \text{path}(v) \), we have \( \text{distance}(v) \geq \ell(p) \).
- Moreover
  \[
  \ell(p) = \ell(s, u_1) + \sum_{i=1}^{k-1} \ell(u_i, u_{i+1}) + \ell(u_k, v) \\
  \geq \text{distance}(u_k) + \ell(u_k, v) \quad \text{induction hyp.}
  \]
  \[
  \geq \min_{(u, v) \in E} (\text{distance}(u) + \ell(u, v))
  \]
  \[
  = \text{distance}(v).
  \]
**Claim**
If $G$ has $n$ vertices and $m$ edges, then the algorithm runs in $O(n + m)$ time.

**Proof.**
Asymptotic complexity of the individual steps:

1. $O(n)$
2. $O(n + m)$ – compute a topological order by depth-first search (includes sorting)
3. $O(m)$ – inner for-loop visits each edge at most once

**Remark**
This yields a linear-time algorithm for shortest paths in time-expanded networks. Recall that building the graph is very expensive.

**Idea**
Use topological orders, but do not build the graph.

$\rightsquigarrow$ Connection Scan Algorithm (Dibbelt/Pajor/Strasser/Wagner, 2013)
2.3 Advanced Methods

Elementary connections

Consider a line network $\mathcal{N}$. Let $t_1, \ldots, t_k$ be the trips of a timetable for $\mathcal{N}$.

Definition

- An elementary connection on an edge $e = (v_{dep}, v_{arr}) \in E(\mathcal{N})$ is a 5-tuple $(v_{dep}, v_{arr}, \tau_{dep}(v_{dep}), \tau_{arr}(v_{arr}), i)$, where $t_i = (\tau_{dep}, \tau_{arr})$ is a trip of a line using $e$.
- If $c$ is an elementary connection, we will write $v_{dep}(c), v_{arr}(c), \tau_{dep}(c), \tau_{arr}(c), \text{trip}(c)$ for its entries.

Observation

There is a one-to-one correspondence between

- elementary connections,
- driving activities of the time expansion,
- departure events of the time expansion,
- arrival events of the time expansion.
Aside: Dominant elementary connections

Definition
Let $c, c'$ be elementary connections. Then $c$ dominates $c'$ if

1. $v_{\text{dep}}(c) = v_{\text{dep}}(c')$ and $v_{\text{arr}}(c) = v_{\text{arr}}(c')$,
2. $\tau_{\text{dep}}(c) \geq \tau_{\text{dep}}(c')$ and $\tau_{\text{arr}}(c) \leq \tau_{\text{arr}}(c')$.

Application to Time-Dependent Dijkstra

- In order to compute the time function, the time-dependent Dijkstra algorithm needs for every edge $(v, w)$ a list of elementary connections on $(v, w)$ sorted by $\tau_{\text{dep}}$.
- The FIFO property means that connections with the earliest departures have the earliest arrivals.
- In particular, finding the connection with the earliest arrival is a simple binary search on the sorted list of elementary connections.
- If FIFO does not hold, one needs to find the first dominant elementary connection.
§2.3 Advanced Methods

Connection Scan Algorithm (CSA)

Let \( s@\tau \rightarrow t \) be an earliest arrival query, \( s \neq t \).

Basic Connection Scan Algorithm (minimum transfer time \( \tau_{\text{min}} \))

**Preprocessing**

1. Sort all elementary connections by \( \tau_{\text{dep}} \) in ascending order.

**Query**

1. \(
\begin{align*}
\text{time}(v) & := \infty \text{ for all stops } v, \ \text{time}(s) := \tau \\
\text{path}(v) & := [] \text{ for all stops } v \\
\text{trip\_used}(i) & := \text{false} \text{ for all trips } i
\end{align*}
\)

2. For all elementary connections \( c \) increasing by \( \tau_{\text{dep}}(c) \):
   - If \( \text{trip\_used}(\text{trip}(c)) \) or \( \text{time}(v_{\text{dep}}(c)) \leq \tau_{\text{dep}}(c) \):
     - \( \text{trip\_used}(\text{trip}(c)) := \text{true} \)
     - If \( \tau_{\text{arr}}(c) + \tau_{\text{min}} < \text{time}(v_{\text{arr}}(c)) \):
       - \( \text{time}(v_{\text{arr}}(c)) := \tau_{\text{arr}}(c) + \tau_{\text{min}} \)
       - \( \text{path}(v_{\text{arr}}(c)) := \text{path}(v_{\text{dep}}(c)) + [c] \)

3. Return \( \text{path}(t), \ \text{time}(t) - \tau_{\text{min}} \).
Query: $A @ 7:00 \rightarrow D$, $\tau_{\text{min}} = 2$ minutes

$\nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

- $A, B, 7:00, 7:05, 1$
- $B, C, 7:05, 7:12, 1$
- $A, B, 7:10, 7:15, 2$
- $B, C, 7:15, 7:22, 2$
- $A, B, 7:05, 7:10, 3$
- $B, D, 7:10, 7:21, 3$
- $A, B, 7:15, 7:20, 4$
- $B, D, 7:20, 7:31, 4$
- $C, D, 7:04, 7:10, 5$
- $C, D, 7:14, 7:20, 6$
- $C, D, 7:24, 7:30, 7$

$C, B, 7:05, 7:12, 8$
$B, A, 7:12, 7:18, 8$
$C, B, 7:15, 7:22, 9$
$B, A, 7:22, 7:28, 9$
$D, B, 7:05, 7:16, 10$
$B, A, 7:16, 7:21, 10$
$D, B, 7:15, 7:26, 11$
$B, A, 7:26, 7:31, 11$
$D, C, 7:04, 7:10, 12$
$D, C, 7:14, 7:20, 13$
$D, C, 7:24, 7:30, 14$
CSA: Example

Query: A@7:00 → D, $\tau_{\text{min}} = 2$ minutes

$v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

- A, B, 7:00, 7:05, 1
- C, D, 7:04, 7:10, 5
- D, C, 7:04, 7:10, 12
- B, C, 7:05, 7:12, 1
- A, B, 7:05, 7:10, 3
- C, B, 7:05, 7:12, 8
- D, B, 7:05, 7:16, 10
- A, B, 7:10, 7:15, 2
- B, D, 7:10, 7:21, 3
- B, A, 7:12, 7:18, 8
- C, D, 7:14, 7:20, 6
- D, C, 7:14, 7:20, 13
- B, C, 7:15, 7:22, 2
- A, B, 7:15, 7:20, 4
- C, B, 7:15, 7:22, 9
- A, B, 7:15, 7:20, 13
- D, B, 7:15, 7:26, 11
- B, A, 7:16, 7:21, 10
- D, B, 7:20, 7:31, 4
- B, A, 7:22, 7:28, 9
- C, D, 7:24, 7:30, 7
- D, C, 7:24, 7:30, 14
- B, A, 7:26, 7:31, 11
§2.3 Advanced Methods

CSA: Example

Query: A@7:00 → D, \( \tau_{\text{min}} = 2 \) minutes

\[
\begin{align*}
&v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \\
A, B, 7:00, 7:05, 1 & D, C, 7:14, 7:20, 13 \\
C, D, 7:04, 7:10, 5 & B, C, 7:15, 7:22, 2 \\
D, C, 7:04, 7:10, 12 & A, B, 7:15, 7:20, 4 \\
B, C, 7:05, 7:12, 1 & C, B, 7:15, 7:22, 9 \\
A, B, 7:05, 7:10, 3 & D, B, 7:15, 7:26, 11 \\
C, B, 7:05, 7:12, 8 & B, A, 7:16, 7:21, 10 \\
D, B, 7:05, 7:16, 10 & B, D, 7:20, 7:31, 4 \\
A, B, 7:10, 7:15, 2 & B, A, 7:22, 7:28, 9 \\
B, D, 7:10, 7:21, 3 & C, D, 7:24, 7:30, 7 \\
B, A, 7:12, 7:18, 8 & D, C, 7:24, 7:30, 14 \\
C, D, 7:14, 7:20, 6 & B, A, 7:26, 7:31, 11
\end{align*}
\]
Query: A@7:00 → D, \( \tau_{\text{min}} = 2 \) minutes

\[\begin{align*}
\nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \\
A,B,7:00,7:05,1 & D,C,7:14,7:20,13 \\
C,D,7:04,7:10,5 & B,C,7:15,7:22,2 \\
D,C,7:04,7:10,12 & A,B,7:15,7:20,4 \\
B,C,7:05,7:12,1 & C,B,7:15,7:22,9 \\
A,B,7:05,7:10,3 & D,B,7:15,7:26,11 \\
C,B,7:05,7:12,8 & B,A,7:16,7:21,10 \\
D,B,7:05,7:16,10 & B,D,7:20,7:31,4 \\
A,B,7:10,7:15,2 & B,A,7:22,7:28,9 \\
B,D,7:10,7:21,3 & C,D,7:24,7:30,7 \\
B,A,7:12,7:18,8 & D,C,7:24,7:30,14 \\
C,D,7:14,7:20,6 & B,A,7:26,7:31,11
\end{align*}\]
2.3 Advanced Methods

CSA: Example

Query: A@7:00 → D, $\tau_{\text{min}} = 2$ minutes

<table>
<thead>
<tr>
<th>$v_{\text{dep}}$, $v_{\text{arr}}$, $\tau_{\text{dep}}$, $\tau_{\text{arr}}$, trip</th>
<th>$v_{\text{dep}}$, $v_{\text{arr}}$, $\tau_{\text{dep}}$, $\tau_{\text{arr}}$, trip</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,7:00,7:05,1</td>
<td>D,C,7:14,7:20,13</td>
</tr>
<tr>
<td>C,D,7:04,7:10,5</td>
<td>B,C,7:15,7:22,2</td>
</tr>
<tr>
<td>D,C,7:04,7:10,12</td>
<td>A,B,7:15,7:20,4</td>
</tr>
<tr>
<td>B,C,7:05,7:12,1</td>
<td>C,B,7:15,7:22,9</td>
</tr>
<tr>
<td>A,B,7:05,7:10,3</td>
<td>D,B,7:15,7:26,11</td>
</tr>
<tr>
<td>C,B,7:05,7:12,8</td>
<td>B,A,7:16,7:21,10</td>
</tr>
<tr>
<td>D,B,7:05,7:16,10</td>
<td>B,D,7:20,7:31,4</td>
</tr>
<tr>
<td>A,B,7:10,7:15,2</td>
<td>B,A,7:22,7:28,9</td>
</tr>
<tr>
<td>B,D,7:10,7:21,3</td>
<td>C,D,7:24,7:30,7</td>
</tr>
<tr>
<td>B,A,7:12,7:18,8</td>
<td>D,C,7:24,7:30,14</td>
</tr>
<tr>
<td>C,D,7:14,7:20,6</td>
<td>B,A,7:26,7:31,11</td>
</tr>
</tbody>
</table>

used trips:
1
§2.3 Advanced Methods

CSA: Example

Query: A@7:00 → D, $\tau_{min} = 2$ minutes

$v_{dep}, v_{arr}, \tau_{dep}, \tau_{arr}, trip$

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
<th>Start</th>
<th>End</th>
<th>Duration</th>
<th>Used Trips</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>7:00</td>
<td>7:05</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>7:04</td>
<td>7:10</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>7:04</td>
<td>7:10</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>7:05</td>
<td>7:12</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>7:05</td>
<td>7:10</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>7:05</td>
<td>7:12</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>D</td>
<td>B</td>
<td>7:05</td>
<td>7:16</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>7:10</td>
<td>7:21</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>7:12</td>
<td>7:18</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>7:14</td>
<td>7:20</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>7:24</td>
<td>7:30</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>7:24</td>
<td>7:30</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>7:26</td>
<td>7:31</td>
<td>11</td>
<td>1</td>
</tr>
</tbody>
</table>
2.3 Advanced Methods

CSA: Example

Query: $A@7:00 \rightarrow D$, $\tau_{min} = 2$ minutes

$v_{dep}, v_{arr}, \tau_{dep}, \tau_{arr}, \text{trip}$

$A,B,7:00,7:05,1$  $D,C,7:14,7:20,13$
$C,D,7:04,7:10,5$  $B,C,7:15,7:22,2$
$D,C,7:04,7:10,12$  $B,B,7:15,7:20,4$
$B,C,7:05,7:12,1$  $C,B,7:15,7:22,9$
$A,B,7:05,7:10,3$  $D,B,7:15,7:26,11$
$C,B,7:05,7:12,8$  $B,A,7:16,7:21,10$
$D,B,7:05,7:16,10$  $B,D,7:20,7:31,4$
$A,B,7:10,7:15,2$  $B,A,7:22,7:28,9$
$B,D,7:10,7:21,3$  $C,D,7:24,7:30,7$
$B,A,7:12,7:18,8$  $D,C,7:24,7:30,14$
$C,D,7:14,7:20,6$  $B,A,7:26,7:31,11$

used trips:

1
CSA: Example

Query: A@7:00 → D, $\tau_{\min} = 2$ minutes

$v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

- A,B,7:00,7:05,1
- C,D,7:04,7:10,5
- D,C,7:04,7:10,12
- B,C,7:05,7:12,1
- A,B,7:05,7:10,3
- C,B,7:05,7:12,8
- D,B,7:05,7:16,10
- A,B,7:10,7:15,2
- B,D,7:10,7:21,3
- B,A,7:12,7:18,8
- C,D,7:14,7:20,6
- D,C,7:14,7:20,13
- B,C,7:15,7:22,2
- A,B,7:15,7:20,4
- C,B,7:15,7:22,9
- D,B,7:15,7:26,11
- B,A,7:16,7:21,10
- B,D,7:20,7:31,4
- B,A,7:22,7:28,9
- C,D,7:24,7:30,7
- D,C,7:24,7:30,14
- B,A,7:26,7:31,11

used trips: 1, 3
Query: A@7:00 → D, $\tau_{\text{min}} = 2$ minutes

$\nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

- A, B, 7:00, 7:05, 1
- C, D, 7:04, 7:10, 5
- D, C, 7:04, 7:10, 12
- B, C, 7:05, 7:12, 1
- A, B, 7:05, 7:10, 3
- C, B, 7:05, 7:12, 8
- D, B, 7:05, 7:16, 10
- A, B, 7:10, 7:15, 2
- B, D, 7:10, 7:21, 3
- B, A, 7:12, 7:18, 8
- C, D, 7:14, 7:20, 6
- D, C, 7:14, 7:20, 13
- B, C, 7:15, 7:22, 2
- A, B, 7:15, 7:20, 4
- C, B, 7:15, 7:22, 9
- D, B, 7:15, 7:26, 11
- B, A, 7:16, 7:21, 10
- B, D, 7:20, 7:31, 4
- B, A, 7:22, 7:28, 9
- C, D, 7:24, 7:30, 7
- D, C, 7:24, 7:30, 14
- B, A, 7:26, 7:31, 11

used trips: 1, 3
**Query:** $A @ 7:00 \rightarrow D$, $\tau_{\text{min}} = 2$ minutes

<table>
<thead>
<tr>
<th>Trip</th>
<th>Dep</th>
<th>Arr</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A, B, 7:00, 7:05, 1$</td>
<td>$C, D, 7:04, 7:10, 5$</td>
<td>$D, C, 7:04, 7:10, 12$</td>
<td>$B, C, 7:05, 7:12, 1$</td>
</tr>
<tr>
<td>$A, B, 7:05, 7:10, 3$</td>
<td>$C, B, 7:05, 7:12, 8$</td>
<td>$D, B, 7:05, 7:16, 10$</td>
<td>$A, B, 7:15, 7:20, 4$</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Nodes:** A, B, C, D
- **Edges:** A to B, A to C, B to C, C to D, D to B
- **Time:** 7:00 to 7:14

**Used Trips:** 1, 3
Query: A@7:00 → D, $\tau_{\text{min}} = 2$ minutes

$v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

<table>
<thead>
<tr>
<th>Trip</th>
<th>Dep Time</th>
<th>Arr Time</th>
<th>Duration</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,7:00</td>
<td>7:00</td>
<td>7:05</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>C,D,7:04</td>
<td>7:04</td>
<td>7:10</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>D,C,7:04</td>
<td>7:04</td>
<td>7:10</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>B,C,7:05</td>
<td>7:05</td>
<td>7:12</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>A,B,7:05</td>
<td>7:05</td>
<td>7:10</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>C,B,7:05</td>
<td>7:05</td>
<td>7:12</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>D,B,7:05</td>
<td>7:05</td>
<td>7:16</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>A,B,7:10</td>
<td>7:10</td>
<td>7:15</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>B,D,7:10</td>
<td>7:10</td>
<td>7:21</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>B,A,7:12</td>
<td>7:12</td>
<td>7:18</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>C,D,7:14</td>
<td>7:14</td>
<td>7:20</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>B,A,7:26</td>
<td>7:26</td>
<td>7:31</td>
<td>5</td>
<td>11</td>
</tr>
</tbody>
</table>
CSA: Example

Query: A@7:00 → D, $\tau_{\text{min}} = 2$ minutes

$\nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

- A,B,7:00,7:05,1
- C,D,7:04,7:10,5
- D,C,7:04,7:10,12
- B,C,7:05,7:12,1
- A,B,7:05,7:10,3
- C,B,7:05,7:12,8
- D,B,7:05,7:16,10
- A,B,7:10,7:15,2
- B,D,7:10,7:21,3
- B,A,7:12,7:18,8
- C,D,7:14,7:20,6
- B,A,7:26,7:31,11

used trips: 1, 3, 2
Query: A@7:00 → D, \( \tau_{\text{min}} = 2 \) minutes

\( v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \)

\[
\begin{align*}
A,B,7:00,7:05,1 & \quad D,C,7:14,7:20,13 \\
C,D,7:04,7:10,5 & \quad B,C,7:15,7:22,2 \\
D,C,7:04,7:10,12 & \quad A,B,7:15,7:20,4 \\
B,C,7:05,7:12,1 & \quad C,B,7:15,7:22,9 \\
A,B,7:05,7:10,3 & \quad D,B,7:15,7:26,11 \\
C,B,7:05,7:12,8 & \quad B,A,7:16,7:21,10 \\
D,B,7:05,7:16,10 & \quad B,D,7:20,7:31,4 \\
A,B,7:10,7:15,2 & \quad B,A,7:22,7:28,9 \\
B,D,7:10,7:21,3 & \quad C,D,7:24,7:30,7 \\
B,A,7:12,7:18,8 & \quad D,C,7:24,7:30,14 \\
C,D,7:14,7:20,6 & \quad B,A,7:26,7:31,11
\end{align*}
\]

used trips: 1, 3, 2, 8
Query: A @ 7:00 → D, $\tau_{\text{min}} = 2$ minutes

$v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

A, B, 7:00, 7:05, 1
C, D, 7:04, 7:10, 5
D, C, 7:04, 7:10, 12
B, C, 7:05, 7:12, 1
A, B, 7:05, 7:10, 3
C, B, 7:05, 7:12, 8
D, B, 7:05, 7:16, 10
A, B, 7:10, 7:15, 2
B, D, 7:10, 7:21, 3
B, A, 7:12, 7:18, 8
C, D, 7:14, 7:20, 6
D, C, 7:14, 7:20, 13
B, C, 7:15, 7:22, 2
A, B, 7:15, 7:20, 4
C, B, 7:15, 7:22, 9
A, B, 7:15, 7:20, 4
B, A, 7:16, 7:21, 10
D, B, 7:15, 7:26, 11
B, A, 7:20, 7:31, 4
B, D, 7:20, 7:31, 4
B, A, 7:22, 7:28, 9
C, D, 7:24, 7:30, 7
D, C, 7:24, 7:30, 14
C, D, 7:24, 7:30, 14
B, A, 7:26, 7:31, 11

used trips:
1, 3, 2, 8, 6

May 7, 2018
Query: A@7:00 → D, $\tau_{\text{min}} = 2$ minutes

$v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$


\begin{align*}
A, B, 7:00, 7:05, &1 & D, C, 7:14, 7:20, &13 \\
C, D, 7:04, 7:10, &5 & B, C, 7:15, 7:22, &2 \\
D, C, 7:04, 7:10, &12 & A, B, 7:15, 7:20, &4 \\
B, C, 7:05, 7:12, &1 & C, B, 7:15, 7:22, &9 \\
A, B, 7:05, 7:10, &3 & D, B, 7:15, 7:26, &11 \\
C, B, 7:05, 7:12, &8 & B, A, 7:16, 7:21, &10 \\
D, B, 7:05, 7:16, &10 & B, D, 7:20, 7:31, &4 \\
A, B, 7:10, 7:15, &2 & B, A, 7:22, 7:28, &9 \\
B, D, 7:10, 7:21, &3 & C, D, 7:24, 7:30, &7 \\
B, A, 7:12, 7:18, &8 & D, C, 7:24, 7:30, &14 \\
C, D, 7:14, 7:20, &6 & B, A, 7:26, 7:31, &11 \\
\end{align*}

used trips: 1, 3, 2, 8, 6
### CSA: Example

**Query:** $A@7:00 \rightarrow D, \tau_{min} = 2$ minutes

<table>
<thead>
<tr>
<th>Trip</th>
<th>$v_{dep}$</th>
<th>$v_{arr}$</th>
<th>$\tau_{dep}$</th>
<th>$\tau_{arr}$</th>
<th>Trip ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A,B,7:00,7:05,1$</td>
<td>$D,C,7:14,7:20,13$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C,D,7:04,7:10,5$</td>
<td>$B,C,7:15,7:22,2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D,C,7:04,7:10,12$</td>
<td>$A,B,7:15,7:20,4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B,C,7:05,7:12,1$</td>
<td>$C,B,7:15,7:22,9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A,B,7:05,7:10,3$</td>
<td>$D,B,7:15,7:26,11$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C,B,7:05,7:12,8$</td>
<td>$B,A,7:16,7:21,10$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D,B,7:05,7:16,10$</td>
<td>$B,D,7:20,7:31,4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A,B,7:10,7:15,2$</td>
<td>$B,A,7:22,7:28,9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B,D,7:10,7:21,3$</td>
<td>$C,D,7:24,7:30,7$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B,A,7:12,7:18,8$</td>
<td>$D,C,7:24,7:30,14$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C,D,7:14,7:20,6$</td>
<td>$B,A,7:26,7:31,11$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Graph:**

- **Nodes:** A, B, C, D
- **Edges:** A-B, A-C, B-C, C-D

**Used trips:** 1, 3, 2, 8, 6
2.3 Advanced Methods

**CSA: Example**

*Query: A@7:00 → D, \( \tau_{\text{min}} = 2 \) minutes*

\( v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \)

<table>
<thead>
<tr>
<th>Trip</th>
<th>Dep</th>
<th>Arr</th>
<th>( \tau_{\text{dep}} )</th>
<th>( \tau_{\text{arr}} )</th>
<th>Trip</th>
<th>Dep</th>
<th>Arr</th>
<th>( \tau_{\text{dep}} )</th>
<th>( \tau_{\text{arr}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, 7:00</td>
<td>7:05</td>
<td>7:05</td>
<td>1</td>
<td></td>
<td>D, C, 7:14</td>
<td>7:20</td>
<td>7:20</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>C, D, 7:04</td>
<td>7:10</td>
<td>7:10</td>
<td>5</td>
<td></td>
<td>B, C, 7:15</td>
<td>7:22</td>
<td>7:22</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>D, C, 7:04</td>
<td>7:10</td>
<td>7:10</td>
<td>12</td>
<td></td>
<td>A, B, 7:15</td>
<td>7:20</td>
<td>7:20</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>B, C, 7:05</td>
<td>7:12</td>
<td>7:12</td>
<td>1</td>
<td></td>
<td>C, B, 7:15</td>
<td>7:22</td>
<td>7:22</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>A, B, 7:05</td>
<td>7:10</td>
<td>7:10</td>
<td>3</td>
<td></td>
<td>D, B, 7:15</td>
<td>7:26</td>
<td>7:26</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>C, B, 7:05</td>
<td>7:12</td>
<td>7:12</td>
<td>8</td>
<td></td>
<td>B, A, 7:16</td>
<td>7:21</td>
<td>7:21</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>D, B, 7:05</td>
<td>7:16</td>
<td>7:16</td>
<td>10</td>
<td></td>
<td>B, D, 7:20</td>
<td>7:31</td>
<td>7:31</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>B, D, 7:10</td>
<td>7:21</td>
<td>7:21</td>
<td>3</td>
<td></td>
<td>C, D, 7:24</td>
<td>7:30</td>
<td>7:30</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>B, A, 7:12</td>
<td>7:18</td>
<td>7:18</td>
<td>8</td>
<td></td>
<td>D, C, 7:24</td>
<td>7:30</td>
<td>7:30</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>C, D, 7:14</td>
<td>7:20</td>
<td>7:20</td>
<td>6</td>
<td></td>
<td>B, A, 7:26</td>
<td>7:31</td>
<td>7:31</td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

*used trips: 1, 3, 2, 8, 6, 4*
Query: A@7:00 → D, $\tau_{\text{min}} = 2$ minutes

\begin{align*}
\nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \\
A,B,7:00,7:05,1 & \quad D,C,7:14,7:20,13 \\
C,D,7:04,7:10,5 & \quad B,C,7:15,7:22,2 \\
D,C,7:04,7:10,12 & \quad A,B,7:15,7:20,4 \\
B,C,7:05,7:12,1 & \quad C,B,7:15,7:22,9 \\
A,B,7:05,7:10,3 & \quad D,B,7:15,7:26,11 \\
C,B,7:05,7:12,8 & \quad B,A,7:16,7:21,10 \\
D,B,7:05,7:16,10 & \quad B,D,7:20,7:31,4 \\
A,B,7:10,7:15,2 & \quad B,A,7:22,7:28,9 \\
B,D,7:10,7:21,3 & \quad C,D,7:24,7:30,7 \\
B,A,7:12,7:18,8 & \quad D,C,7:24,7:30,14 \\
C,D,7:14,7:20,6 & \quad B,A,7:26,7:31,11
\end{align*}

used trips:

1, 3, 2, 8, 6, 4, 9


**CSA: Example**

**Query:** \( A \text{@7:00} \rightarrow D, \ \tau_{\text{min}} = 2 \text{ minutes} \)

- \( v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \)

<table>
<thead>
<tr>
<th>Trip</th>
<th>Dep</th>
<th>Arr</th>
<th>Dep</th>
<th>Arr</th>
<th>Trip</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,7:00,7:05,1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,D,7:04,7:10,5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D,C,7:04,7:10,12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,C,7:05,7:12,1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A,B,7:05,7:10,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,B,7:05,7:12,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D,B,7:05,7:16,10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A,B,7:10,7:15,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,D,7:10,7:21,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,A,7:12,7:18,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,D,7:14,7:20,6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D,C,7:14,7:20,13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,C,7:15,7:22,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A,B,7:15,7:20,4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,B,7:15,7:22,9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D,B,7:15,7:26,11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,A,7:16,7:21,10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,D,7:20,7:31,4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,A,7:22,7:28,9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,D,7:24,7:30,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D,C,7:24,7:30,14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,A,7:26,7:31,11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**used trips:**

1, 3, 2, 8, 6, 4, 9
Query: $A@7:00 \rightarrow D$, $\tau_{\min} = 2$ minutes

$\nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

- $A, B, 7:00, 7:05, 1$
- $C, D, 7:04, 7:10, 5$
- $D, C, 7:04, 7:10, 12$
- $B, C, 7:05, 7:12, 1$
- $A, B, 7:05, 7:10, 3$
- $C, B, 7:05, 7:12, 8$
- $D, B, 7:05, 7:16, 10$
- $A, B, 7:10, 7:15, 2$
- $B, D, 7:10, 7:21, 3$
- $B, A, 7:12, 7:18, 8$
- $C, D, 7:14, 7:20, 6$
- $D, C, 7:14, 7:20, 13$
- $B, C, 7:15, 7:22, 2$
- $A, B, 7:15, 7:20, 4$
- $C, B, 7:15, 7:22, 9$
- $A, B, 7:15, 7:20, 4$
- $D, B, 7:15, 7:26, 11$
- $B, A, 7:16, 7:21, 10$
- $B, D, 7:20, 7:31, 4$
- $B, A, 7:22, 7:28, 9$
- $C, D, 7:24, 7:30, 7$
- $D, C, 7:24, 7:30, 14$
- $B, A, 7:26, 7:31, 11$

used trips:
1, 3, 2, 8, 6, 4, 9, 10
2.3 Advanced Methods

CSA: Example

Query: A@7:00 → D, $\tau_{\text{min}} = 2$ minutes

$v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

\[
\begin{align*}
A, B, 7:00, 7:05, 1 & \quad D, C, 7:14, 7:20, 13 \\
C, D, 7:04, 7:10, 5 & \quad B, C, 7:15, 7:22, 2 \\
D, C, 7:04, 7:10, 12 & \quad A, B, 7:15, 7:20, 4 \\
B, C, 7:05, 7:12, 1 & \quad C, B, 7:15, 7:22, 9 \\
A, B, 7:05, 7:10, 3 & \quad D, B, 7:15, 7:26, 11 \\
C, B, 7:05, 7:12, 8 & \quad B, A, 7:16, 7:21, 10 \\
D, B, 7:05, 7:16, 10 & \quad B, D, 7:20, 7:31, 4 \\
A, B, 7:10, 7:15, 2 & \quad B, A, 7:22, 7:28, 9 \\
B, D, 7:10, 7:21, 3 & \quad C, D, 7:24, 7:30, 7 \\
B, A, 7:12, 7:18, 8 & \quad D, C, 7:24, 7:30, 14 \\
C, D, 7:14, 7:20, 6 & \quad B, A, 7:26, 7:31, 11 \\
\end{align*}
\]

used trips:
1, 3, 2, 8, 6, 4, 9, 10
2.3 Advanced Methods

CSA: Example

Query: A@7:00 → D, \( \tau_{\text{min}} = 2 \) minutes

\[ \nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \]

- \( A, B, 7:00, 7:05, 1 \)
- \( C, D, 7:04, 7:10, 5 \)
- \( D, C, 7:04, 7:10, 12 \)
- \( B, C, 7:05, 7:12, 1 \)
- \( A, B, 7:05, 7:10, 3 \)
- \( C, B, 7:05, 7:12, 8 \)
- \( D, B, 7:05, 7:16, 10 \)
- \( A, B, 7:10, 7:15, 2 \)
- \( B, D, 7:10, 7:21, 3 \)
- \( B, A, 7:12, 7:18, 8 \)
- \( C, D, 7:14, 7:20, 6 \)
- \( D, C, 7:14, 7:20, 13 \)
- \( B, C, 7:15, 7:22, 2 \)
- \( A, B, 7:15, 7:20, 4 \)
- \( C, B, 7:15, 7:22, 9 \)
- \( A, B, 7:15, 7:20, 4 \)
- \( D, B, 7:15, 7:26, 11 \)
- \( C, B, 7:15, 7:22, 9 \)
- \( B, A, 7:16, 7:21, 10 \)
- \( B, D, 7:20, 7:31, 4 \)
- \( B, A, 7:22, 7:28, 9 \)
- \( C, D, 7:24, 7:30, 7 \)
- \( D, C, 7:24, 7:30, 14 \)
- \( B, A, 7:26, 7:31, 11 \)

used trips:
1, 3, 2, 8, 6, 4, 9, 10, 7, 14, 11
Query: \( A@7:00 \rightarrow D, \tau_{\text{min}} = 2 \text{ minutes} \)

\[
\begin{align*}
V_{\text{dep}}, V_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} & \\
A, B, 7:00, 7:05, 1 & \quad D, C, 7:14, 7:20, 13 \\
C, D, 7:04, 7:10, 5 & \quad B, C, 7:15, 7:22, 2 \\
D, C, 7:04, 7:10, 12 & \quad A, B, 7:15, 7:20, 4 \\
B, C, 7:05, 7:12, 1 & \quad C, B, 7:15, 7:22, 9 \\
A, B, 7:05, 7:10, 3 & \quad D, B, 7:15, 7:26, 11 \\
C, B, 7:05, 7:12, 8 & \quad B, A, 7:16, 7:21, 10 \\
D, B, 7:05, 7:16, 10 & \quad B, D, 7:20, 7:31, 4 \\
A, B, 7:10, 7:15, 2 & \quad B, A, 7:22, 7:28, 9 \\
B, D, 7:10, 7:21, 3 & \quad C, D, 7:24, 7:30, 7 \\
B, A, 7:12, 7:18, 8 & \quad D, C, 7:24, 7:30, 14 \\
C, D, 7:14, 7:20, 6 & \quad B, A, 7:26, 7:31, 11 \\
\end{align*}
\]

used trips:
1, 3, 2, 8, 6, 4, 9, 10, 7, 14, 11

\(~\Rightarrow~\text{earliest arrival time at D: 7:20}~\)
2.3 Advanced Methods

CSA: Interpretation

Interpretation

- Scanning the elementary connections in ascending order by departure time is equivalent to scanning departure events of the time-expanded graph w.r.t. the topological order given by departure time.

- The successor of a departure event is a single arrival event, whose successors are in turn several departure events. Hence it makes sense to contract the arrival event and to view the next departure events as direct successors.

- The CSA does the following: If a connection $c$ departs at $v_{\text{dep}}(c)$ at time $\tau_{\text{dep}}(c)$, then set the time at the next departure event at $v_{\text{arr}}(c)$ to $\tau_{\text{arr}}(c) + \tau_{\text{min}}$ if this improves the value in time.

- This is done if the $\tau_{\text{dep}}(c) \geq \text{time}(v_{\text{dep}}(c)) = \tau_{\text{arr}}(c') + \tau_{\text{min}}$ for some connection $c'$ – i.e., including a transfer – or if the same trip has been used before, disregarding the transfer time.

- The transfer time to the last departure event should be subtracted.
More Remarks

- $\text{path}(t)$ contains the elementary connections on the computed route.
- In fact, CSA solves the earliest arrival problem for every target vertex.

Optimizations

- Starting criterion: A connection $c$ needs only to be scanned if $\tau_{\text{dep}}(c) \geq \tau$. 
  $\rightsquigarrow$ Determine the first connection departing after $\tau$ by binary search.

- Stopping criterion: In an $s\oplus \tau \rightarrow t$ earliest arrival query, CSA can be stopped if it scans a connection $c$ with $\tau_{\text{dep}}(c) \geq \text{time}(t)$, because $\text{time}(t)$ then cannot be improved further.
CSA: Footpaths

Definition

A **footpath graph** for a line network $\mathcal{N}$ is a digraph $F$ with a length function $\ell : E(F) \to \mathbb{R}_{\geq 0}$ such that

1. $V(F) = V(\mathcal{N})$,
2. $F$ is transitively closed, i.e., $(u, v), (v, w) \in E(F) \Rightarrow (u, w) \in E(F)$,
3. $\ell$ satisfies the triangle inequality.

Application to transfers

- Transfers between stops of the line network are modeled by edges of the footpath graph, and their duration is given by $\ell$.
- Minimum transfer times at a stop correspond to loops in the footpath graph.
- In more detailed line networks, e.g., when each platform or bus stop location corresponds to a vertex, there might also be loops of length 0.
§2.3 Advanced Methods

CSA: Algorithm with footpaths

Connection Scan Algorithm (with footpath graph \((F, \ell)\))

Preprocessing

1. Sort all elementary connections by \(\tau_{dep}\) in ascending order.

Query

1. \(\text{time}(v) := \infty\) and \(\text{path}(v) := []\) for all stops \(v\),
   \(\text{time}(v) := \tau + \ell(s, v)\) and \(\text{path}(v) := [(s, v)]\) for all \((s, v) \in E(F)\),
   \(\text{trip\_used}(i) := \text{none}\) for all trips \(i\)

2. For all elementary connections \(c\) with \(\tau_{dep}(c) \geq \tau\), in asc. order:
   If \(\tau_{dep}(c) \geq \text{time}(t)\), go to Step 3.
   If \(\text{trip\_used}(\text{trip}(c)) \neq \text{none}\) or \(\text{time}(v_{dep}(c)) \leq \tau_{dep}(c)\):
     If \(\text{trip\_used}(\text{trip}(c)) = \text{none}\):
       \(\text{trip\_used}(\text{trip}(c)) := c\)
     If \(\tau_{arr}(c) < \text{time}(v_{arr}(c))\):
       For all \(f = (v_{arr}(c), w) \in E(F)\):
         If \(\tau_{arr}(c) + \ell(f) < \text{time}(w)\):
           \(\text{time}(w) := \tau_{arr}(c) + \ell(f)\)
           \(\text{path}(w) := \text{path}(v_{dep}(c)) + [(\text{trip\_used}(\text{trip}(c)), c), f]\)

3. Return \(\text{path}(t), \text{time}(t)\).
Every journey computed by CSA (path) is an alternating sequence $(f_1, l_1, f_2, l_2, \ldots, f_k, l_k, f_{k+1})$, where the $f_i$ are footpaths and $l_i$ are legs, i.e., subsequent elementary connections of the same trip.

The field path returns the legs by specifying the enter and exit connections of each trip.

The footpath graph needs loops at every station so that connections can be reached at all. Another variant would be to use a detailed model with parent stations connected to several platforms/bus stop locations/... In this model, there are footpaths between a parent station and its individual platforms. Queries run from parent station to parent station.

This CSA version includes the starting and stopping criterion.

Moreover, it uses the following limited walking strategy: Footpaths with $\tau_{arr}(c) \geq \text{time}(v_{arr}(c))$ are not considered.
2.3 Advanced Methods

CSA: Limited walking

Lemma

Suppose $\tau_{arr}(c) \geq \text{time}(\nu_{arr}(c))$. Then no footpath $(\nu_{arr}(c), w) \in E(F)$ improves $\text{time}(w)$.

Proof.

- Let $\nu := \nu_{arr}(c)$. Assume there is a footpath $(\nu, w) \in E(F)$ improving $\text{time}(w)$, i.e., $\text{time}(w) > \tau_{arr}(c) + \ell(\nu, w)$.
- Then by hypothesis, $\text{time}(w) > \text{time}(\nu) + \ell(\nu, w)$.
- If $\text{time}(\nu) = \infty$, then this is not improving.
- Otherwise $\text{time}(\nu) = \tau_{arr}(c') + \ell(u, \nu)$ for some connection $c'$ arriving at some stop $u$ and $(u, \nu) \in E(F)$.
- Thus $\text{time}(w) > \tau_{arr}(c') + \ell(u, \nu) + \ell(\nu, w)$. Since the footpath graph is transitive and $\ell$ satisfies the triangle inequality, there is an edge $(u, w) \in E(F)$ such that $\text{time}(w) > \tau_{arr}(c') + \ell(u, w)$.
- This is impossible, as CSA scanned $c'$ and $(u, w)$ before and would have set $\text{time}(w)$ accordingly.
Final remarks

- The main advantage of CSA is that it does not need to build the graph. Moreover, elementary connections are accessed sequentially, only the footpath graph needs random access.
- However, it scans every connection, also very distant and unlikely ones. Correcting this is one of the ideas of accelerated CSA.
- There is a profile version of CSA.
- There is a version finding the Pareto optimal solutions w.r.t. earliest arrival time and minimum number of transfers.
The next algorithm is called RAPTOR (Delling/Pajor/Werneck, 2012), which is short for *Round-based public transit routing*.

The RAPTOR algorithm is designed to solve the two-criteria problem w.r.t. earliest arrival time and minimum number of transfers in a Pareto sense: For \( k \in \mathbb{N}_0 \), it computes the earliest arrival time w.r.t. a journey with at most \( k \) transfers.

The algorithm works therefore with rounds, and each round increases the number of transfers by 1, so it is a *dynamic program*.

RAPTOR does not use graphs as underlying data structure.
Pareto optimization

Definition
Let \( X \) be a set and let \( f_1, \ldots, f_k : X \to \mathbb{R} \) be functions.

- The problem \( \min_{x \in X} (f_1(x), \ldots, f_k(x)) \) is called a \textit{multi-criteria minimization problem}.
- An element \( x \in X \) dominates \( y \in X \) if \( f_i(x) \leq f_i(y) \) for all \( i \).
- An element \( x \in X \) is \textit{Pareto-optimal} if \( x \) is not dominated by another element of \( X \).
- The \textit{Pareto set} or \textit{Pareto front} is the set of all Pareto-optimal elements of \( X \).

![Diagram showing earliest arrival time vs. number of transfers]
§2.3 Advanced Methods

RAPTOR: Pareto optimization

Application to RAPTOR

- A journey is a sequence of elementary connections and footpaths in the order of travel.
- A journey $J$ dominates a journey $J'$ if $J$ arrives no later than $J'$ and $J$ does not use more than transfers than $J'$.
- Given an earliest arrival problem $s@\tau \rightarrow t$, a Pareto set is a maximal set of pairwise non-dominating journeys from $s$ to $t$ not departing before $\tau$. 
RAPTOR: Model

Input data structures

RAPTOR works in principle on a directed line network with a timetable and a footpath graph. However, the information is usually organized in lists:

- **RouteStops**: for each line $L$ a sorted list of the stops served by $L$,
- **Trips**: for each line $L$ a sorted list of trips on $L$,
- **StopTimes**: for each trip a sorted list of departures and arrivals,
- **StopRoutes**: for each stop $v$ a list of the lines serving $v$,
- **Transfers**: for each stop $v$ a list of footpaths from $v$.

Comparison to the CSA model

The models for RAPTOR and CSA differ in the notion of journeys: There is not necessarily a footpath between trips. RAPTOR does hence not respect minimum transfer times if arrival and departure are at the same stop. This has to be modelled by introducing a stop for each platform/location.
2.3 Advanced Methods

**RAPTOR: Algorithm**

**Basic RAPTOR Algorithm (K rounds)**

1. \( \text{time}(k, v) := \infty \) for each stop \( v \) and \( k = 0, 1, \ldots, K \), \( \text{time}(0, s) := \tau \)

2. For \( k = 1, 2, \ldots, K \):
   \[ \text{time}(k, v) := \text{time}(k - 1, v) \] for all \( v \)
   For all lines \( L = (v_1, \ldots, v_r) \):
   \( \text{curr_trip} := \emptyset \)
   For \( j = 1, \ldots, r \):
     - If \( \text{curr_trip} \neq \emptyset \) and \( \tau_{\text{arr}}(v_j) < \text{time}(k, v_j) \):
       \[ (\tau_{\text{dep}}, \tau_{\text{arr}}) := \text{curr_trip} \]
       \( \text{time}(k, v_j) := \tau_{\text{arr}}(v_j) \)
       \( \text{path}(k, v_j) := \text{path}(k - 1, v_i) + [(v_i, v_j, \tau_{\text{dep}}(v_i), \tau_{\text{arr}}(v_j), \text{curr_trip})] \)
     - If \( \text{curr_trip} = \emptyset \) or \( \tau_{\text{dep}}(v_j) \geq \text{time}(k - 1, v_j) \):
       \( \text{curr_trip} := \text{trip}(k, v, L) \)
   For all footpaths \( (v, w) \):
     - If \( \text{time}(k, v) + \ell(v, w) < \text{time}(k, w) \):
       \( \text{time}(k, w) := \text{time}(k, v) + \ell(v, w) \)
       \( \text{path}(k, w) := \text{path}(k, v) + [(v, w)] \)

3. Return \( \text{path}(\cdot, t), \text{time}(\cdot, t) \).

\( \text{trip}(k, v, L) := \text{earliest trip } (\tau_{\text{dep}}, \tau_{\text{arr}}) \text{ on } L \text{ s.t. } \tau_{\text{dep}}(v) \geq \text{time}(k - 1, v) \text{ if exists, else } \emptyset \)
Query: A@7:00 → D, K = 2, no footpaths

 vomiting, ν_{arr}, τ_{dep}, τ_{arr}, trip

A, B, 7:00, 7:05, 1  C, B, 7:05, 7:12, 8
B, C, 7:05, 7:12, 1  B, A, 7:12, 7:18, 8
A, B, 7:10, 7:15, 2  C, B, 7:15, 7:22, 9
A, B, 7:05, 7:10, 3  D, B, 7:05, 7:16, 10
B, D, 7:10, 7:21, 3  B, A, 7:16, 7:21, 10
A, B, 7:15, 7:20, 4  D, B, 7:15, 7:26, 11
A, B, 7:15, 7:20, 4  D, B, 7:15, 7:26, 11

**§2.3 Advanced Methods**

**RAPTOR: Example**

**Query:** $A@7:00 \rightarrow D$, $K = 2$, no footpaths

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

$v_{dep}$, $v_{arr}$, $\tau_{dep}$, $\tau_{arr}$, trip

- $A, B, 7:00, 7:05, 1$
- $C, B, 7:05, 7:12, 8$
- $B, C, 7:05, 7:12, 1$
- $B, A, 7:12, 7:18, 8$
- $A, B, 7:10, 7:15, 2$
- $C, B, 7:15, 7:22, 9$
- $B, C, 7:15, 7:22, 2$
- $B, A, 7:22, 7:28, 9$
- $A, B, 7:05, 7:10, 3$
- $D, B, 7:05, 7:16, 10$
- $B, D, 7:10, 7:21, 3$
- $B, A, 7:16, 7:21, 10$
- $A, B, 7:15, 7:20, 4$
- $D, B, 7:15, 7:26, 11$
- $B, D, 7:20, 7:31, 4$
- $B, A, 7:26, 7:31, 11$
- $C, D, 7:04, 7:10, 5$
- $D, C, 7:04, 7:10, 12$
- $C, D, 7:14, 7:20, 6$
- $D, C, 7:14, 7:20, 13$
- $C, D, 7:24, 7:30, 7$
- $D, C, 7:24, 7:30, 14$
§2.3 Advanced Methods

RAPTOR: Example

Query: A@7:00 → D, K = 2, no footpaths

current trip: ∅

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>

\[ v_{dep}, v_{arr}, t_{dep}, t_{arr}, \text{trip} \]

\[
\begin{align*}
A, B, 7:00, 7:05, 1 & \quad C, B, 7:05, 7:12, 8 \\
B, C, 7:05, 7:12, 1 & \quad B, A, 7:12, 7:18, 8 \\
A, B, 7:10, 7:15, 2 & \quad C, B, 7:15, 7:22, 9 \\
B, C, 7:15, 7:22, 2 & \quad B, A, 7:22, 7:28, 9 \\
A, B, 7:05, 7:10, 3 & \quad D, B, 7:05, 7:16, 10 \\
B, D, 7:10, 7:21, 3 & \quad B, A, 7:16, 7:21, 10 \\
A, B, 7:15, 7:20, 4 & \quad D, B, 7:15, 7:26, 11 \\
B, D, 7:20, 7:31, 4 & \quad B, A, 7:26, 7:31, 11 \\
C, D, 7:04, 7:10, 5 & \quad D, C, 7:04, 7:10, 12 \\
C, D, 7:14, 7:20, 6 & \quad D, C, 7:14, 7:20, 13 \\
C, D, 7:24, 7:30, 7 & \quad D, C, 7:24, 7:30, 14 
\end{align*}
\]
§2.3 Advanced Methods

RAPTOR: Example

Query: A@7:00 → D, K = 2, no footpaths

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>

\[ v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \]

- A,B,7:00,7:05,1
- B,C,7:05,7:12,1
- A,B,7:10,7:15,2
- B,C,7:15,7:22,2
- A,B,7:00,7:10,3
- B,D,7:10,7:21,3
- A,B,7:15,7:20,4
- B,D,7:20,7:31,4
- C,D,7:04,7:10,5
- C,D,7:14,7:20,6
- C,D,7:24,7:30,7

- C,B,7:05,7:12,8
- B,A,7:12,7:18,8
- C,B,7:15,7:22,9
- B,A,7:22,7:28,9
- D,B,7:05,7:16,10
- B,A,7:16,7:21,10
- D,B,7:15,7:26,11
- B,A,7:26,7:31,11
- D,C,7:04,7:10,12
- D,C,7:14,7:20,13
- D,C,7:24,7:30,14
### RAPTOR: Example

**Query:** $A@7:00 \rightarrow D$, $K = 2$, no footpaths

![Graph](image)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>7:05</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>

- $v_{\text{dep}}$, $v_{\text{arr}}$, $\tau_{\text{dep}}$, $\tau_{\text{arr}}$, trip
  - $A,B,7:00,7:05,1$  $C,B,7:05,7:12,8$
  - $B,C,7:05,7:12,1$  $B,A,7:12,7:18,8$
  - $A,B,7:10,7:15,2$  $C,B,7:15,7:22,9$
  - $A,B,7:05,7:10,3$  $D,B,7:05,7:16,10$
  - $B,D,7:10,7:21,3$  $B,A,7:16,7:21,10$
  - $A,B,7:15,7:20,4$  $D,B,7:15,7:26,11$
  - $C,D,7:04,7:10,5$  $D,C,7:04,7:10,12$
  - $C,D,7:14,7:20,6$  $D,C,7:14,7:20,13$
  - $C,D,7:24,7:30,7$  $D,C,7:24,7:30,14$
### RAPTOR: Example

**Query:** A@7:00 $\rightarrow$ D, $K = 2$, no footpaths

![Graph diagram]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>$\infty$</td>
<td>7:05</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>7:12</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td></td>
</tr>
</tbody>
</table>

$v_{dep}$, $v_{arr}$, $\tau_{dep}$, $\tau_{arr}$, trip

- A,B,7:00,7:05,1
- B,C,7:05,7:12,1
- A,B,7:10,7:15,2
- B,C,7:15,7:22,2
- A,B,7:05,7:10,3
- B,D,7:10,7:21,3
- A,B,7:15,7:20,4
- B,D,7:20,7:31,4
- A,B,7:15,7:20,4
- B,A,7:22,7:28,9
- A,B,7:05,7:12,8
- B,A,7:12,7:18,8
- C,B,7:15,7:22,9
- B,A,7:22,7:28,9
- D,B,7:05,7:16,10
- B,A,7:16,7:21,10
- D,B,7:15,7:26,11
- B,A,7:26,7:31,11
- C,D,7:04,7:10,5
- D,C,7:04,7:10,12
- C,D,7:14,7:20,6
- D,C,7:14,7:20,13
- C,D,7:24,7:30,7
- D,C,7:24,7:30,14
Query: A@7:00 → D, K = 2, no footpaths

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>7:05</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>7:12</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>

v_{dep}, v_{arr}, \tau_{dep}, \tau_{arr}, \text{trip}

- A,B,7:00,7:05,1
- B,C,7:05,7:12,1
- A,B,7:10,7:15,2
- B,C,7:15,7:22,2
- B,D,7:10,7:21,3
- A,B,7:15,7:20,4
- B,D,7:20,7:31,4
- C,D,7:04,7:10,5
- C,D,7:14,7:20,6
- C,D,7:24,7:30,7

- C,B,7:05,7:12,8
- B,A,7:12,7:18,8
- C,B,7:15,7:22,9
- B,A,7:22,7:28,9
- D,B,7:05,7:16,10
- B,A,7:16,7:21,10
- D,B,7:15,7:26,11
- B,A,7:26,7:31,11
- D,C,7:04,7:10,12
- D,C,7:14,7:20,13
- D,C,7:24,7:30,14
§2.3 Advanced Methods

RAPTOR: Example

Query: A@7:00 → D, \( K = 2 \), no footpaths

\[
\begin{array}{|c|c|c|c|}
\hline
\text{current trip: 3} & 0 & 1 & 2 \\
\hline
A & 7:00 & 7:00 & \\
B & \infty & 7:05 & \\
C & \infty & 7:12 & \\
D & \infty & \infty & \\
\hline
\end{array}
\]

\[
\begin{array}{c}
\nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \\
A,B,7:00,7:05,1 & C,B,7:05,7:12,8 \\
B,C,7:05,7:12,1 & B,A,7:12,7:18,8 \\
A,B,7:10,7:15,2 & C,B,7:15,7:22,9 \\
A,B,7:05,7:10,3 & D,B,7:05,7:16,10 \\
B,D,7:10,7:21,3 & B,A,7:16,7:21,10 \\
A,B,7:15,7:20,4 & D,B,7:15,7:26,11 \\
C,D,7:04,7:10,5 & D,C,7:04,7:10,12 \\
C,D,7:14,7:20,6 & D,C,7:14,7:20,13 \\
C,D,7:24,7:30,7 & D,C,7:24,7:30,14 \\
\end{array}
\]
§2.3 Advanced Methods

RAPTOR: Example

Query: $A@7:00 \rightarrow D$, $K = 2$, no footpaths

$\nu_{\text{dep}}, \nu_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>current trip: 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>$\infty$</td>
<td>7:05</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>7:12</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>7:21</td>
<td></td>
</tr>
</tbody>
</table>

$A, B, 7:00, 7:05, 1$  $C, B, 7:05, 7:12, 8$
$B, C, 7:05, 7:12, 1$  $B, A, 7:12, 7:18, 8$
$A, B, 7:10, 7:15, 2$  $C, B, 7:15, 7:22, 9$
$A, B, 7:05, 7:10, 3$  $D, B, 7:05, 7:16, 10$
$B, D, 7:10, 7:21, 3$  $B, A, 7:16, 7:21, 10$
$A, B, 7:15, 7:20, 4$  $D, B, 7:15, 7:26, 11$
$C, D, 7:04, 7:20, 6$  $D, C, 7:04, 7:10, 12$
$C, D, 7:14, 7:20, 6$  $D, C, 7:14, 7:20, 13$
$C, D, 7:24, 7:30, 7$  $D, C, 7:24, 7:30, 14$
### §2.3 Advanced Methods

#### RAPTOR: Example

**Query:** A@7:00 → D, $K = 2$, no footpaths

![Graph with nodes A, B, C, D]

**v_{dep}, v_{arr}, \tau_{dep}, \tau_{arr}, trip**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>7:05</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>7:12</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>7:21</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B</td>
<td>7:00,7:05</td>
<td>7:10,7:15</td>
<td>7:05,7:10</td>
</tr>
<tr>
<td>B,C</td>
<td>7:05,7:12</td>
<td>7:15,7:22</td>
<td>7:05,7:16</td>
</tr>
<tr>
<td>A,B</td>
<td>7:10,7:15</td>
<td>7:20,7:30</td>
<td>7:04,7:14</td>
</tr>
<tr>
<td>B,D</td>
<td>7:10,7:21</td>
<td>7:20,7:31</td>
<td>7:04,7:10</td>
</tr>
<tr>
<td>A,B</td>
<td>7:15,7:20</td>
<td>7:24,7:30</td>
<td>7:14,7:20</td>
</tr>
</tbody>
</table>

May 7, 2018
**2.3 Advanced Methods**

### RAPTOR: Example

**Query:** $A @ 7:00 \rightarrow D$, $K = 2$, no footpaths

**Graph:**

```
  A -- B -- C -- D
  |     |     |
  B -- D
```

**Current trip:** $\emptyset$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>$\infty$</td>
<td>7:05</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>7:12</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>7:21</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$v_{dep}$, $v_{arr}$, $\tau_{dep}$, $\tau_{arr}$, trip</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A, B, 7:00, 7:05, 1$</td>
</tr>
<tr>
<td>$B, C, 7:05, 7:12, 1$</td>
</tr>
<tr>
<td>$A, B, 7:10, 7:15, 2$</td>
</tr>
<tr>
<td>$B, C, 7:15, 7:22, 2$</td>
</tr>
<tr>
<td>$A, B, 7:05, 7:10, 3$</td>
</tr>
<tr>
<td>$B, D, 7:10, 7:21, 3$</td>
</tr>
<tr>
<td>$A, B, 7:15, 7:20, 4$</td>
</tr>
<tr>
<td>$B, D, 7:20, 7:31, 4$</td>
</tr>
<tr>
<td>$A, B, 7:15, 7:20, 4$</td>
</tr>
<tr>
<td>$B, D, 7:20, 7:31, 4$</td>
</tr>
<tr>
<td>$C, D, 7:04, 7:10, 5$</td>
</tr>
<tr>
<td>$D, C, 7:04, 7:10, 5$</td>
</tr>
<tr>
<td>$C, D, 7:14, 7:20, 6$</td>
</tr>
<tr>
<td>$D, C, 7:14, 7:20, 6$</td>
</tr>
<tr>
<td>$C, D, 7:24, 7:30, 7$</td>
</tr>
<tr>
<td>$D, C, 7:24, 7:30, 7$</td>
</tr>
<tr>
<td>$C, D, 7:24, 7:30, 14$</td>
</tr>
<tr>
<td>$D, C, 7:24, 7:30, 14$</td>
</tr>
</tbody>
</table>
Query: $A@7:00 \rightarrow D$, $K = 2$, no footpaths

$v_{dep}$, $v_{arr}$, $\tau_{dep}$, $\tau_{arr}$, trip

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_{dep}$</td>
<td>$A,B,7:00,7:05,1$</td>
<td>$C,B,7:05,7:12,8$</td>
<td></td>
</tr>
<tr>
<td>$v_{arr}$</td>
<td>$B,C,7:05,7:12,1$</td>
<td>$B,A,7:12,7:18,8$</td>
<td></td>
</tr>
<tr>
<td>$\tau_{dep}$</td>
<td>$A,B,7:10,7:15,2$</td>
<td>$C,B,7:15,7:22,9$</td>
<td></td>
</tr>
<tr>
<td>$\tau_{arr}$</td>
<td>$B,C,7:15,7:22,2$</td>
<td>$B,A,7:22,7:28,9$</td>
<td></td>
</tr>
<tr>
<td>trip</td>
<td>$A,B,7:05,7:10,3$</td>
<td>$D,B,7:05,7:16,10$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$B,D,7:10,7:21,3$</td>
<td>$B,A,7:16,7:21,10$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A,B,7:15,7:20,4$</td>
<td>$D,B,7:15,7:26,11$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$C,D,7:04,7:10,5$</td>
<td>$D,C,7:04,7:10,12$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$C,D,7:14,7:20,6$</td>
<td>$D,C,7:14,7:20,13$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$C,D,7:24,7:30,7$</td>
<td>$D,C,7:24,7:30,14$</td>
<td></td>
</tr>
</tbody>
</table>
Query: A@7:00 → D, K = 2, no footpaths

\[ V_{dep}, V_{arr}, T_{dep}, T_{arr}, \text{trip} \]

\[
\begin{align*}
A,B,7:00,7:05,1 & \quad C,B,7:05,7:12,8 \\
B,C,7:05,7:12,1 & \quad B,A,7:12,7:18,8 \\
A,B,7:10,7:15,2 & \quad C,B,7:15,7:22,9 \\
B,C,7:15,7:22,2 & \quad B,A,7:22,7:28,9 \\
A,B,7:05,7:10,3 & \quad D,B,7:05,7:16,10 \\
B,D,7:10,7:21,3 & \quad B,A,7:16,7:21,10 \\
A,B,7:15,7:20,4 & \quad D,B,7:15,7:26,11 \\
B,D,7:20,7:31,4 & \quad B,A,7:26,7:31,11 \\
C,D,7:04,7:10,5 & \quad D,C,7:04,7:10,12 \\
C,D,7:14,7:20,6 & \quad D,C,7:14,7:20,13 \\
C,D,7:24,7:30,7 & \quad D,C,7:24,7:30,14 \\
\end{align*}
\]
### §2.3 Advanced Methods

**RAPTOR: Example**

**Query:** A@7:00 → D, \( K = 2 \), no footpaths

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>7:05</td>
<td>7:05</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>7:12</td>
<td>7:12</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>7:21</td>
<td>7:21</td>
</tr>
</tbody>
</table>

\( v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \)

- A,B,7:00,7:05,1
- B,C,7:05,7:12,1
- A,B,7:10,7:15,2
- B,C,7:15,7:22,2
- A,B,7:05,7:10,3
- B,D,7:10,7:21,3
- A,B,7:15,7:20,4
- B,D,7:20,7:31,4
- C,D,7:04,7:10,5
- C,D,7:14,7:20,6
- C,D,7:24,7:30,7
- C,D,7:04,7:12,12
- D,C,7:14,7:20,13
- D,C,7:24,7:30,14
RAPTOR: Example

Query: \( A \)@7:00 \( \rightarrow \) \( D \), \( K = 2 \), no footpaths

\[
\begin{array}{c|c|c|c}
\text{node} & \nu_{\text{dep}} & \nu_{\text{arr}} & \tau_{\text{dep}} & \tau_{\text{arr}} & \text{trip} \\
\hline
A & 7:00 & 7:05 & 1 & & A,B,7:00,7:05,1 \\
B & \infty & 7:05 & 1 & & C,B,7:05,7:12,8 \\
C & \infty & 7:12 & 1 & & B,A,7:12,7:18,8 \\
D & \infty & 7:21 & 1 & & B,A,7:22,7:28,9 \\
\end{array}
\]
2.3 Advanced Methods

**RAPTOR: Example**

**Query:** A@7:00 → D, K = 2, no footpaths

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>7:05</td>
<td>7:05</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>7:12</td>
<td>7:12</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>7:21</td>
<td>7:21</td>
</tr>
</tbody>
</table>

$v_{dep}, v_{arr}, \tau_{dep}, \tau_{arr}, \text{trip}$

- A,B,7:00,7:05,1
- B,C,7:05,7:12,1
- A,B,7:10,7:15,2
- B,C,7:15,7:22,2
- A,B,7:05,7:10,3
- B,D,7:10,7:21,3
- A,B,7:15,7:20,4
- B,D,7:20,7:31,4
- C,D,7:04,7:10,5
- C,D,7:14,7:20,6
- C,D,7:24,7:30,7
- D,C,7:04,7:10,12
- D,C,7:14,7:20,13
- D,C,7:24,7:30,14

Current trip: 3
Query: $A@7:00 \to D$, $K = 2$, no footpaths

$$
\begin{array}{c|c|c|c}
\text{current trip: 3} & 0 & 1 & 2 \\
\hline
A & 7:00 & 7:00 & 7:00 \\
B & \infty & 7:05 & 7:05 \\
C & \infty & 7:12 & 7:12 \\
D & \infty & 7:21 & 7:21 \\
\end{array}
$$

$$
\begin{align*}
&v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip} \\
&A,B,7:00,7:05,1 & C,B,7:05,7:12,8 \\
&B,C,7:05,7:12,1 & B,A,7:12,7:18,8 \\
&A,B,7:10,7:15,2 & C,B,7:15,7:22,9 \\
&A,B,7:05,7:10,3 & B,D,7:10,7:21,3 \\
&B,D,7:10,7:21,3 & B,A,7:16,7:21,10 \\
&A,B,7:15,7:20,4 & D,B,7:15,7:26,11 \\
&A,B,7:15,7:20,4 & D,B,7:15,7:26,11 \\
&C,D,7:04,7:10,5 & D,C,7:04,7:10,12 \\
&D,C,7:04,7:10,12 & D,C,7:14,7:20,13 \\
&C,D,7:14,7:20,6 & D,C,7:14,7:20,13 \\
&D,C,7:14,7:20,13 & D,C,7:24,7:30,14 \\
\end{align*}
$$
Query: A@7:00 → D, $K = 2$, no footpaths

$v_{dep}, v_{arr}, \tau_{dep}, \tau_{arr}, \text{trip}$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>7:05</td>
<td>7:05</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>7:12</td>
<td>7:12</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>7:21</td>
<td>7:21</td>
</tr>
</tbody>
</table>

A,B,7:00,7:05,1  C,B,7:05,7:12,8
B,C,7:05,7:12,1  B,A,7:12,7:18,8
A,B,7:10,7:15,2  C,B,7:15,7:22,9
A,B,7:05,7:10,3  D,B,7:05,7:16,10
B,D,7:10,7:21,3  B,A,7:16,7:21,10
A,B,7:15,7:20,4  D,B,7:15,7:26,11
C,D,7:04,7:10,5  D,C,7:04,7:10,12
C,D,7:14,7:20,6  D,C,7:14,7:20,13
C,D,7:24,7:30,7  D,C,7:24,7:30,14
Query: A@7:00 → D, $K = 2$, no footpaths

$v_{\text{dep}}, v_{\text{arr}}, \tau_{\text{dep}}, \tau_{\text{arr}}, \text{trip}$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>∞</td>
<td>7:05</td>
<td>7:05</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>∞</td>
<td>7:12</td>
<td>7:12</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>∞</td>
<td>7:21</td>
<td>7:21</td>
</tr>
<tr>
<td><strong>A,B,7:00,7:05,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B,C,7:05,7:12,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A,B,7:10,7:15,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B,C,7:15,7:22,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A,B,7:05,7:10,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B,D,7:10,7:21,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A,B,7:15,7:20,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B,D,7:20,7:31,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A,B,7:15,7:20,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B,D,7:20,7:31,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C,D,7:04,7:10,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D,C,7:04,7:10,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A,B,7:15,7:20,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B,A,7:26,7:31,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B,C,7:14,7:20,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D,C,7:14,7:20,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C,D,7:24,7:30,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D,C,7:24,7:30,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Query: A@7:00 → D, K = 2, no footpaths

v_{dep}, v_{arr}, \tau_{dep}, \tau_{arr}, \text{trip}

A,B,7:00,7:05,1  C,B,7:05,7:12,8
B,C,7:05,7:12,1  B,A,7:12,7:18,8
A,B,7:10,7:15,2  C,B,7:15,7:22,9
A,B,7:05,7:10,3  D,B,7:05,7:16,10
B,D,7:10,7:21,3  B,A,7:16,7:21,10
A,B,7:15,7:20,4  D,B,7:15,7:26,11
C,D,7:04,7:10,5  D,C,7:04,7:10,12
C,D,7:14,7:20,6  D,C,7:14,7:20,13
C,D,7:24,7:30,7  D,C,7:24,7:30,14

current trip: 6
§2.3 Advanced Methods

**RAPTOR: Example**

*Query*: A@7:00 → D, $K = 2$, no footpaths

![Diagram of a network with nodes A, B, C, D and connections]

<table>
<thead>
<tr>
<th>current trip:</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>7:05</td>
<td>7:05</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>7:12</td>
<td>7:12</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>7:21</td>
<td>7:20</td>
</tr>
</tbody>
</table>


$\nu_{dep}, \nu_{arr}, \tau_{dep}, \tau_{arr}, \text{ trip}$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,7:00,7:05,1</td>
<td>C,B,7:05,7:12,8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,C,7:05,7:12,1</td>
<td>B,A,7:12,7:18,8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A,B,7:10,7:15,2</td>
<td>C,B,7:15,7:22,9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A,B,7:05,7:10,3</td>
<td>D,B,7:05,7:16,10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B,D,7:10,7:21,3</td>
<td>B,A,7:16,7:21,10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A,B,7:15,7:20,4</td>
<td>D,B,7:15,7:26,11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,D,7:04,7:10,5</td>
<td>D,C,7:04,7:10,12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,D,7:14,7:20,6</td>
<td>D,C,7:14,7:20,13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C,D,7:24,7:30,7</td>
<td>D,C,7:24,7:30,14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

⇠ earliest arrival at D: 7:20

(Remark: omitted scanning the lines in the other direction)
RAPTOR: Discussion

- \( \text{time}(k, v) \) is the earliest arrival time at \( v \) using at most \( k \) trips.
- \( \text{path}(k, v) \) contains the corresponding journey as sequence of elementary connections and footpaths.
- In a round, each line \( L \) is scanned exactly once.
- When a line is scanned, the algorithm begins with the first stop \( v \) on the line where a trip departs after \( \text{time}(k - 1, v) \). The arrival times on the following stops are set following this trip (\( \text{curr\_trip} \)). If a stop with an earlier arrival is found, reset \( \text{curr\_trip} \) to the earlier trip and continue.
- Footpaths are considered at the end of each round.
- There is no preprocessing, except for the organization of timetable data in arrays.
The algorithm can be stopped after round $k$ if for all stops $v$ holds $\text{time}(k, v) = \text{time}(k - 1, v)$.

**Marking:** During round $k$, it suffices to consider routes containing a stop reached with *exactly* $k - 1$ trips. The arrival times at the stops of all other routes have not improved in round $k - 1$, hence they can only be improved by *another* route in round $k$.

This adds the following to RAPTOR: If $\text{time}(k, v)$ changed, then *mark* $v$. Instead of scanning all routes in round $k + 1$, consider pairs $(L, v)$, where $L$ is a route and $v$ is the earliest marked stop on $L$. Before updating the arrival times, every stop gets unmarked. Mark $s$ at the beginning.

**Stopping criterion:** In an $s@\tau \rightarrow t$ query, there is no need to consider arrival times $\tau_{\text{arr}}(v_j) > \text{time}(k, t)$. 
§2.3 Advanced Methods

**RAPTOR: Extensions**

**Range queries (rRAPTOR)**

- Suppose we want to find a set of Pareto-optimal journeys departing at \( s \) in a time range \( R \).

- **rRAPTOR Algorithm:**
  - Let \( T \) be a list of departure times of trips leaving \( s \) within \( R \), sorted in descending order.
  - Run RAPTOR for each \( \tau \in T \), but keep time and path for the next \( \tau \).

**Multi-criteria problems (McRAPTOR)**

- The *McRAPTOR* algorithm stores a set of pairwise non-dominating labels for each round \( k \) and stop \( v \).
- If fare zones are to be integrated, a potential label could be (arrival time, set of touched fare zones).
- When traversing a route, the labels are updated. Finally, dominated labels are discarded.
Transfer Patterns

Overview

- Transfer Patterns are a speedup technique for large public transportation networks (Bast et. al., 2010).
- The key observation is that, independent of time, many optimal journeys from s to t make the same transfers.
- In a huge preprocessing step, for each pair \((s, t)\) of stops some sequences of potentially optimal transfers is computed – these are the actual transfer patterns.
- At query time, these transfer patterns are merged into a small query graph, where Dijkstra’s algorithm is fast enough to solve the shortest path problem in reasonable time.
- Transfer Patterns are used e.g. by Google Transit.
2.3 Advanced Methods

Transfer Patterns: Definition

For a line network $\mathcal{N}$ with a timetable, consider its realistic time-expanded network $\mathcal{E}$ (i.e., with transfer events).

**Definition**

- The **transfer pattern** of a path in $\mathcal{E}$ is the sequence of the stations in $\mathcal{N}$ corresponding to the first event, each arrival event whose successor is a transfer event, and the last event.

- An **optimal set of transfer patterns** for a pair $(s, t) \in V(\mathcal{N})$ is a set $S$ of transfer patterns such that:
  - for all earliest arrival queries $s@\tau \rightarrow t$, there is an optimal set of $s$-$t$-journeys whose transfer patterns are contained in $S$,
  - every element in $S$ is the transfer pattern of an optimal $s$-$t$-journey for some earliest arrival query $s@\tau \rightarrow t$.

In this context, an $s$-$t$-journey for an earliest arrival query $s@\tau \rightarrow t$ is a path from the first transfer event at $s$ after $\tau$ to some arrival event of $\tau$. 
Transfer Patterns: Direct Connections

Consider the following data structures:

- For each line $L$, sort its trips by first departure and organize them in a table:

<table>
<thead>
<tr>
<th>Line L1</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trip 1</td>
<td>07:12</td>
<td>07:22</td>
<td>07:23</td>
<td>07:44</td>
</tr>
<tr>
<td>Trip 2</td>
<td>09:14</td>
<td>09:22</td>
<td>09:23</td>
<td>09:44</td>
</tr>
</tbody>
</table>

- For each station, compute the list of lines incident to it and its position on the line:

  - Stop $v_1$:
    - (L1, 1) (L2, 5) ... |
  - Stop $v_2$:
    - (L1, 2) (L3, 6) ... |

A direct connection query $s@\tau \rightarrow t$ (i.e., no transfers allowed), can be answered in the following way:

- Intersect the list of incident lines of $s$ and $t$.
- For each occurrence of $s$ before $t$ on a line, read off the cost of the earliest feasible trip (FIFO), and take the optimal cost among all lines.
Transfer Patterns: Preprocessing

Preprocessing

For every station $s$ of the line network, compute the transfer patterns $(s, t)$ for all stations $t$ reachable from $s$.

Details

- Run a shortest-path algorithm from all transfer events of $s$ to compute transfer patterns $(s, v_1, \ldots, v_k, t)$ for all reachable stations $t$.
- Store the transfer patterns in a DAG with reversed arrows:

Each directed path between rectangular vertices corresponds to a transfer pattern. A reachable station occurs as the label of a rectangular vertex and possibly in several circular vertices.

- There is also a multi-criteria version.
Transfer Patterns: Query graph

Query graph

For a query $s @ T \rightarrow t$, the query graph $Q_{s,t}$ is constructed as follows:

- Fetch the transfer pattern DAG for $s$.
- Let $L$ be the set of (distinct) labels of the successors of $t$ in the DAG. Add edges $(\ell, t)$ to $Q_{s,t}$ for each $\ell \in L$.
- Recursively perform Step 2 for each successor $\neq s$.

Remark

The DAG has reversed arrows because it is easier to look for successors than for predecessors.
Let $s@\tau \rightarrow t$ be an earliest arrival query.

**Basic Transfer Patterns Algorithm**

*Preprocessing*

1. Create the data structures for direct connection queries.
2. Compute the transfer patterns DAG for every station $s$ of the line network.

*Query*

1. Build the query graph $Q_{s,t}$.
2. Run the time-dependent Dijkstra algorithm on $Q_{s,t}$ using the direct connection data structures.
§2.3 Advanced Methods

Transfer Patterns: Example

Query: A@7:00 → D, no footpaths
**Transfer Patterns: Example**

*Query:* A@7:00 $\rightarrow$ D, no footpaths

**Direct connection tables (backward direction omitted):**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td>7:00</td>
<td>7:05</td>
<td>7:05</td>
<td>7:12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7:10</td>
<td>7:15</td>
<td>7:15</td>
<td>7:22</td>
<td></td>
</tr>
<tr>
<td>blue</td>
<td>3</td>
<td>7:05</td>
<td>7:10</td>
<td>7:10</td>
<td>7:21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7:15</td>
<td>7:20</td>
<td>7:20</td>
<td>7:31</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>green</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7:04</td>
<td>7:10</td>
</tr>
<tr>
<td>6</td>
<td>7:14</td>
<td>7:20</td>
</tr>
<tr>
<td>7</td>
<td>7:24</td>
<td>7:30</td>
</tr>
</tbody>
</table>

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>red, 1</td>
<td>blue, 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>red, 2</td>
<td>blue, 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>red, 3</td>
<td>green, 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>blue, 3</td>
<td>green, 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
§2.3 Advanced Methods

Transfer Patterns: Example

Query: A@7:00 → D, no footpaths

Direct connection tables (backward direction omitted):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>green</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7:00</td>
<td>7:05</td>
<td>7:12</td>
<td></td>
<td>7:04</td>
<td>7:10</td>
</tr>
<tr>
<td>2</td>
<td>7:10</td>
<td>7:15</td>
<td>7:22</td>
<td></td>
<td>7:14</td>
<td>7:20</td>
</tr>
<tr>
<td>3</td>
<td>7:05</td>
<td>7:10</td>
<td>7:21</td>
<td></td>
<td>7:24</td>
<td>7:30</td>
</tr>
<tr>
<td>4</td>
<td>7:15</td>
<td>7:20</td>
<td>7:31</td>
<td></td>
<td>7:24</td>
<td>7:30</td>
</tr>
</tbody>
</table>

Transfer Pattern DAG for A:
§2.3 Advanced Methods

Transfer Patterns: Example

Query: \( A \@ 7:00 \rightarrow D \), no footpaths

Direct connection tables (backward direction omitted):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td>7:00</td>
<td>7:05</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7:10</td>
<td>7:15</td>
</tr>
<tr>
<td>blue</td>
<td>3</td>
<td>7:05</td>
<td>7:10</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7:15</td>
<td>7:20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>green</td>
<td>5</td>
<td>7:04</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>7:14</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>7:24</td>
</tr>
</tbody>
</table>

Transfer Pattern DAG for A:

Query graph for (A, D):

May 7, 2018
**Transfer Patterns: Example**

**Query:** A@7:00 → D, no footpaths

Direct connection tables (backward direction omitted):

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>green</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>7:00</td>
<td>7:05</td>
</tr>
<tr>
<td>2</td>
<td>7:10</td>
<td>7:15</td>
</tr>
<tr>
<td>3</td>
<td>7:05</td>
<td>7:10</td>
</tr>
<tr>
<td>4</td>
<td>7:15</td>
<td>7:20</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transfer Pattern DAG for A:

Query graph for (A, D):

May 7, 2018
§2.3 Advanced Methods

Transfer Patterns: Example

Query: A@7:00 → D, no footpaths

Direct connection tables (backward direction omitted):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>7:00</td>
<td>7:05</td>
<td>7:12</td>
</tr>
<tr>
<td>1</td>
<td>7:00</td>
<td>7:05</td>
<td>7:12</td>
</tr>
<tr>
<td>2</td>
<td>7:10</td>
<td>7:15</td>
<td>7:22</td>
</tr>
<tr>
<td>blue</td>
<td>7:05</td>
<td>7:10</td>
<td>7:21</td>
</tr>
<tr>
<td>3</td>
<td>7:05</td>
<td>7:10</td>
<td>7:21</td>
</tr>
<tr>
<td>4</td>
<td>7:15</td>
<td>7:20</td>
<td>7:31</td>
</tr>
<tr>
<td>green</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7:04</td>
<td>7:10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7:14</td>
<td>7:20</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7:24</td>
<td>7:30</td>
<td></td>
</tr>
</tbody>
</table>

Transfer Pattern DAG for A:

Query graph for (A, D):

May 7, 2018
Transfer Patterns: Example

Query: A@7:00 → D, no footpaths

Direct connection tables (backward direction omitted):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td>7:00</td>
<td>7:05 7:05</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7:10</td>
<td>7:15 7:15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>green</td>
<td>5</td>
<td>7:04 7:10</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>7:14 7:20</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>7:24 7:30</td>
</tr>
</tbody>
</table>

Transfer Pattern DAG for A:

Query graph for (A, D):

A → B → D

B → C

C → D

A → B

C → D

May 7, 2018
Transfer Patterns: Example

Query: A@7:00 → D, no footpaths

Direct connection tables (backward direction omitted):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>7:00</td>
<td>7:05 7:05</td>
<td>7:12</td>
</tr>
<tr>
<td></td>
<td>7:10</td>
<td>7:15 7:15</td>
<td>7:22</td>
</tr>
<tr>
<td>blue</td>
<td>7:05</td>
<td>7:10 7:10</td>
<td>7:21</td>
</tr>
<tr>
<td></td>
<td>7:15</td>
<td>7:20 7:20</td>
<td>7:31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>green</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>7:04</td>
<td>7:10</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>7:14</td>
<td>7:20</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>7:24</td>
<td>7:30</td>
</tr>
</tbody>
</table>

Transfer Pattern DAG for A:

Query graph for (A, D):

May 7, 2018
2.3 Advanced Methods

Transfer Patterns: Example

Query: A@7:00 → D, no footpaths

Direct connection tables (backward direction omitted):

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>green</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7:00</td>
<td>7:05</td>
<td>7:05</td>
<td>7:12</td>
<td>5</td>
<td>7:04</td>
<td>7:10</td>
</tr>
<tr>
<td>2</td>
<td>7:10</td>
<td>7:15</td>
<td>7:15</td>
<td>7:22</td>
<td>6</td>
<td>7:14</td>
<td>7:20</td>
</tr>
<tr>
<td>3</td>
<td>7:05</td>
<td>7:10</td>
<td>7:10</td>
<td>7:21</td>
<td>7</td>
<td>7:24</td>
<td>7:30</td>
</tr>
</tbody>
</table>

Transfer Pattern DAG for A:

Query graph for (A, D):

⇝ earliest arrival at D: 7:20
2.3 Advanced Methods

Transfer Patterns: Optimization

- Remember that time-expanded networks are huge. This makes the preprocessing for Transfer Patterns a real challenge.
- Solution: Store only transfer patterns to hubs.
- Hubs are selected using a heuristic strategy, e.g., by random sampling of earliest arrival queries and taking the stations with the highest number of journeys passing through it.
- The query graph needs to evaluate the transfer patterns from, to and between all relevant hubs for $s$ and $t$.
- This strategy is still correct and reduces memory usage significantly.
- There are more optimizations, some of them are heuristic.
## §2.3 Advanced Methods

### Routing: Comparison

#### Earliest arrival query benchmarks (Bast et. al., 2015)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Instance</th>
<th>Elem. conn.</th>
<th>Query time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-Expanded Dijkstra</td>
<td>London</td>
<td>$5 \cdot 10^6$</td>
<td>44.8 ms</td>
</tr>
<tr>
<td>Time-Dependent Dijkstra</td>
<td>London</td>
<td>$5 \cdot 10^6$</td>
<td>11.0 ms</td>
</tr>
<tr>
<td>Connection Scan</td>
<td>London</td>
<td>$5 \cdot 10^6$</td>
<td>2.0 ms</td>
</tr>
<tr>
<td>RAPTOR*</td>
<td>London</td>
<td>$5 \cdot 10^6$</td>
<td>5.4 ms</td>
</tr>
<tr>
<td>Transfer Patterns**</td>
<td>Germany</td>
<td>$90 \cdot 10^6$</td>
<td>0.4 ms</td>
</tr>
</tbody>
</table>

*RAPTOR: Pareto optimization  **Transfer Patterns: 22 d 13 h preprocessing time

#### Range+Pareto query benchmarks (Bast et. al., 2015)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Instance</th>
<th>Elem. conn.</th>
<th>Query time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Scan</td>
<td>London</td>
<td>$5 \cdot 10^6$</td>
<td>466.0 ms</td>
</tr>
<tr>
<td>rRAPTOR</td>
<td>London</td>
<td>$5 \cdot 10^6$</td>
<td>922.0 ms</td>
</tr>
<tr>
<td>Transfer Patterns*</td>
<td>Germany</td>
<td>$90 \cdot 10^6$</td>
<td>39.6 ms</td>
</tr>
</tbody>
</table>

*Transfer Patterns: 23 d 14 h preprocessing time
Appendix: GTFS

Google’s *General Transit Feed Specification* is a de facto standard for exchanging timetable data of public transportation networks. It is a zip archive containing several plain text files. Each text file represents a table by comma-separated values (csv).

**Contents**

- `agency.txt` – information about transport companies
- `stops.txt` – list of stops with coordinates
- `routes.txt` – list of routes (lines)
- `trips.txt` – trips for each route
- `stop_times.txt` – departure and arrival times for each trip
- `calendar.txt` – days on which trips run
- several optional files (transfers, calendar exceptions, frequencies, . . . )
Appendix: GTFS stops

**stops.txt**

```
stop_id,stop_name,stop_lat,stop_lon,location_type,parent_station
900000051303,"U Dahlem-Dorf",52.457695,13.290011,1,
070101000015,"U Dahlem-Dorf",52.457695,13.290011,0,9000000051303
070101000067,"U Dahlem-Dorf",52.457695,13.290011,0,9000000051303
070101000679,"U Dahlem-Dorf",52.457695,13.290011,0,9000000051303
070101000843,"U Dahlem-Dorf",52.457695,13.290011,0,9000000051303
070201034001,"U Dahlem-Dorf",52.457695,13.290011,0,9000000051303
070201034002,"U Dahlem-Dorf",52.457695,13.290011,0,9000000051303
```

- Each stop location has a unique ID (`stop_id`), e.g., 070201034001 is the southbound track of subway line U3.
- All stop locations belong to a meta-stop (`parent_station`).
- Stops are equipped with WGS84 coordinates (`stop_lat`, `stop_lon`).
§2.3 Advanced Methods

Appendix: GTFS trips

**routes.txt**

route_id,agency_id,route_short_name,route_type
17515_400,796,"U3",400  ← urban railway service route U3 with ID 17515_400

**trips.txt**

route_id,service_id,trip_id,trip_headsign
17515_400,913,74049828,"U Krumme Lanke"  ← trip 74049828 on southbound U3

**stop_times.txt**

trip_id,arrival_time,departure_time,stop_id,stop_sequence
74049828,3:12:00,3:12:00,070201013101,0
74049828,3:14:00,3:14:00,070201013101,1
...  
74049828,3:27:00,3:27:00,070201033901,9
74049828,3:28:30,3:28:30,070201034001,10  ← stop at U Dahlem-Dorf
74049828,3:30:30,3:30:30,070201034101,11
74049828,3:32:00,3:32:00,070201034201,12
74049828,3:34:00,3:34:00,070201034301,13
74049828,3:36:00,3:36:00,070201034402,14
Appendix: GTFS resources

- The GTFS documentation is available at [https://developers.google.com/transit/gtfs](https://developers.google.com/transit/gtfs).
- Several transport associations provide their timetable as open data in GTFS format, e.g., VBB (Berlin-Brandenburg), VRN (Rhein-Neckar).