Connection Scan

Julian Dibbelt, Thomas Pajor, Ben Strasser, Dorothea Wagner
What is a Timetable?

Karlsruhe
What is a Timetable?

Karlsruhe

8:00 → 8:31

Mannheim
What is a Timetable?

Karlsruhe

8:00 → 8:31

Mannheim

Rome

8:31 → 11:00

Milan
What is a Timetable?

Karlsruhe

8:00 → 8:31

Mannheim

8:31 → 9:08

Frankfurt

Rome

8:31 → 11:00

Milan
What is a Timetable?

A train station
What is a Timetable?

Can the user do this instantely?
What is a Timetable?

One stop with minimum change time 10 min
What is a Timetable?

Many stops with footpaths
No minimum change times
What is a Timetable?

Mix change times and footpaths
What is a Timetable?

Karlsruhe

8:00 → 8:31

Mannheim

8:31 → 9:08

Frankfurt

Rome

8:31 → 11:00

Milan

For simplicity: We ignore footpaths in this lecture.
What is a Timetable?

Karlsruhe / 10 min

8:00 → 8:31

Mannheim / 9 min

8:31 → 9:08

Rome / 10 min

8:31 → 11:00

Milan / 12 min

Frankfurt / 12 min
What is a Timetable?

Karlsruhe HBf Vorplatz / 5 min

12 min

Karlsruhe / 10 min

8:00 → 8:31

Mannheim / 9 min

8:31 → 9:08

Rome / 10 min

8:31 → 11:00

Karlsruhe HBf Vorplatz / 5 min

For simplicity: We ignore footpaths in this lecture.
What is a Timetable?

Karlsruhe HBf Vorplatz / 5 min

Karlsruhe / 10 min

8:00 → 8:31

Mannheim / 9 min

8:31 → 9:08

Rome / 10 min

8:31 → 11:00

Milan / 12 min

Frankfurt / 12 min

For simplicity: We ignore footpaths in this lecture.
What is a Timetable?

- A timetable contains stops, connections, trips.
- A connection is a train that drives from one stop to another one without intermediate halt.
- A trip is a sequence of connections operated by the same train.
- A connection has a departure and arrival time and a departure and arrival stop and a trip ID.
- The timetable is **aperiodic**: Connections do not repeat.
- A path through a timetable is called journey.
- Switching between trains of different trips is a transfer.
- Transfers require time. This is formalized as following:
  - Stops have a minimum change time.
    Each transfer (within the stop) needs at least this amount of time.
  - Footpaths exist with a constant walking time between adjacent stops.
    (The footpath graph usually is highly disconnected, i.e., main station platforms are connected to subway platforms, but not neighbouring tram stops on the same line.)
# Basic Connection Scan

## Earliest Arrival Time Problem

**Input:** Ordered list of connections, source stop, source time, target stop  
**Output:** Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connections</th>
<th>depstop</th>
<th>arrstop</th>
<th>depertime</th>
<th>arrtime</th>
<th>tripid</th>
<th>depstop</th>
<th>arrstop</th>
<th>depertime</th>
<th>arrtime</th>
<th>tripid</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderd by departure time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>is trip reachable?</th>
<th>F</th>
<th>F</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

**Ben Strasser – Connection Scan**
## Basic Connection Scan

### Earliest Arrival Time Problem

**Input:** Ordered list of connections, source stop, source time, target stop

**Output:** Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connections</th>
<th>dep</th>
<th>arr</th>
<th>tripid</th>
<th>dep</th>
<th>arr</th>
<th>tripid</th>
<th>dep</th>
<th>arr</th>
<th>tripid</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderd by departure time</td>
<td>9:00</td>
<td>9:25</td>
<td></td>
<td>9:15</td>
<td>9:45</td>
<td></td>
<td>9:25</td>
<td>9:55</td>
<td></td>
</tr>
</tbody>
</table>

| is trip reachable?   | F | F |       |     |     |        |     |     |        |

Ben Strasser – Connection Scan
## Basic Connection Scan

### Earliest Arrival Time Problem

**Input:** Ordered list of connections, source stop, source time, target stop

**Output:** Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>$\infty$</td>
<td>8:00</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>ordered by</td>
</tr>
<tr>
<td>departure time</td>
</tr>
<tr>
<td>dep</td>
</tr>
<tr>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>is trip reachable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
</tr>
</tbody>
</table>
Basic Connection Scan

### Earliest Arrival Time Problem

**Input:** Ordered list of connections, source stop, source time, target stop

**Output:** Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>+∞</td>
<td>8:00</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connections</th>
<th>dep</th>
<th>arr</th>
<th>9:00</th>
<th>9:25</th>
<th>tripid</th>
<th>dep</th>
<th>arr</th>
<th>9:15</th>
<th>9:45</th>
<th>tripid</th>
<th>dep</th>
<th>arr</th>
<th>9:25</th>
<th>9:55</th>
<th>tripid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>is trip reachable?</th>
<th>F</th>
<th></th>
<th>F</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ben Strasser – Connection Scan 6/35
Basic Connection Scan

Earliest Arrival Time Problem

Input: Ordered list of connections, source stop, source time, target stop
Output: Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>+∞</td>
<td>8:00</td>
<td>+∞</td>
<td>9:25</td>
<td>+∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connections</th>
<th>dep</th>
<th>arr</th>
<th>dep</th>
<th>arr</th>
<th>dep</th>
<th>arr</th>
<th>dep</th>
<th>arr</th>
<th>dep</th>
<th>arr</th>
<th>dep</th>
<th>arr</th>
<th>dep</th>
<th>arr</th>
<th>dep</th>
<th>arr</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderd by</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>departure time</td>
<td>9:00</td>
<td>9:25</td>
<td>9:15</td>
<td>9:45</td>
<td>9:25</td>
<td>9:55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| is trip reachable?   | T   |     | F   |     |     |     |     |     |     |     |     |     |     |     |     |     |

Ben Strasser – Connection Scan
### Earliest Arrival Time Problem

**Input:** Ordered list of connections, source stop, source time, target stop

**Output:** Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>$+\infty$</td>
<td>8:00</td>
<td>$+\infty$</td>
<td>9:25</td>
<td>$+\infty$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connections ordered by departure time</th>
</tr>
</thead>
<tbody>
<tr>
<td>dep</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>T</td>
</tr>
</tbody>
</table>

**is trip reachable?**

<table>
<thead>
<tr>
<th>T</th>
<th>F</th>
<th>...</th>
</tr>
</thead>
</table>
**Basic Connection Scan**

### Earliest Arrival Time Problem

**Input:** Ordered list of connections, source stop, source time, target stop

**Output:** Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>+∞</td>
<td>8:00</td>
<td>+∞</td>
<td>9:25</td>
<td>+∞</td>
</tr>
</tbody>
</table>

**Connections ordered by departure time**

<table>
<thead>
<tr>
<th>dep</th>
<th>arr</th>
<th>9:00</th>
<th>9:25</th>
<th>tripid</th>
<th>dep</th>
<th>arr</th>
<th>9:15</th>
<th>9:45</th>
<th>tripid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Is trip reachable?**

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Basic Connection Scan

## Earliest Arrival Time Problem

**Input:** Ordered list of connections, source stop, source time, target stop  
**Output:** Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>$+\infty$</td>
<td>8:00</td>
<td>$+\infty$</td>
<td>9:25</td>
<td>9:55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connections</th>
<th>dep</th>
<th>arr</th>
<th>9:00</th>
<th>9:25</th>
<th>tripid</th>
<th>dep</th>
<th>arr</th>
<th>9:15</th>
<th>9:45</th>
<th>tripid</th>
</tr>
</thead>
<tbody>
<tr>
<td>ordered by</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>departure time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>is trip reachable?</th>
<th>T</th>
<th></th>
<th>F</th>
<th></th>
<th></th>
</tr>
</thead>
</table>

Ben Strasser – Connection Scan
## Basic Connection Scan

### Earliest Arrival Time Problem

**Input:** Ordered list of connections, source stop, source time, target stop  
**Output:** Earliest (on time) arrival time

<table>
<thead>
<tr>
<th>change time (in min)</th>
<th>4</th>
<th>10</th>
<th>5</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival time</td>
<td>$+\infty$</td>
<td>8:00</td>
<td>$+\infty$</td>
<td>9:25</td>
<td>9:55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connections ordered by departure time</th>
</tr>
</thead>
<tbody>
<tr>
<td>dep</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>is trip reachable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
</tr>
</tbody>
</table>
Observation: Trains departing before the source time are never needed.

⇒ Do a binary search to determine the first connection that departs no earlier than the source time and start the scan there.
Stopcriterion

**So far:** We solve the one-to-all problem.

**Question:** Can we do better given a target stop $t$, i.e., solve the one-to-one problem?
Stopcriterion

**So far:** We solve the one-to-all problem.

**Question:** Can we do better given a target stop $t$, i.e., solve the one-to-one problem?

**Observation:** Trains departing after the arrival time at $t$ are never useful.

$\Rightarrow$ Abort the scan if the time at $t$ is not bigger than the departure time of the current connection.
Profile Queries

**Problem:** The user often does not know his departure or arrival time.  
**Solution:** Provide journeys for a whole time range.
Profile Queries

Problem: The user often does not know his departure or arrival time.
Solution: Provide journeys for a whole time range.
Profile Queries

**Problem**: The user often does not know his departure or arrival time.

**Solution**: Provide journeys for a whole time range.

```
<table>
<thead>
<tr>
<th>Source Stop</th>
<th>Target Stop</th>
<th>Minimum Departure Time</th>
<th>Maximum Arrival Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karlsruhe Hbf</td>
<td>Leipzig Hbf</td>
<td>15:00</td>
<td>19:01</td>
</tr>
<tr>
<td>Karlsruhe Hbf</td>
<td>Leipzig Hbf</td>
<td>16:00</td>
<td>20:18</td>
</tr>
<tr>
<td>Karlsruhe Hbf</td>
<td>Leipzig Hbf</td>
<td>18:01</td>
<td>22:55</td>
</tr>
<tr>
<td>Karlsruhe Hbf</td>
<td>Leipzig Hbf</td>
<td>18:51</td>
<td>00:10</td>
</tr>
<tr>
<td>Karlsruhe Hbf</td>
<td>Leipzig Hbf</td>
<td>18:51</td>
<td>00:47</td>
</tr>
<tr>
<td>Karlsruhe Hbf</td>
<td>Leipzig Hbf</td>
<td>19:01</td>
<td>00:10</td>
</tr>
<tr>
<td>Karlsruhe Hbf</td>
<td>Leipzig Hbf</td>
<td>19:01</td>
<td>00:47</td>
</tr>
</tbody>
</table>
```

Screenshot of bahn.de

Ben Strasser – Connection Scan
Profile Connection Scan

Earliest Arrival Backward Profile Problem

Input: Timetable, target stop $t$
Output: (full) $st$-Profile for every stop $s$ (except $t$)
Profile Connection Scan

Departing journeys correspond to jumps.

Earliest Arrival Backward Profile Problem

Input: Timetable, target stop $t$
Output: (full) $st$-Profile for every stop $s$ (except $t$)
Profile Connection Scan

Core Idea: Dynamic Programming

At the arrival of each connection the user has three options:
  - He remains seated.
  - He exits the train and waits for another one.
  - He finishes his journey (only valid at target stop).

Profile-Representation: Store a profile function as dynamic array of ordered \((\text{deptime}, \text{arrtime})\)-pairs.

Note: The pairs are simultaneously ordered increasing by \text{deptime} and \text{arrtime}. 
Profile Connection Scan

For every connection $c$ decreasing by departure time:

profile at $c$’s departure stop

profile at $c$’s arrival stop
For every connection $c$ decreasing by departure time:

New pairs must be inserted on blue line.
Profile Connection Scan

For every connection $c$ decreasing by departure time:

- Profile at $c$'s departure stop
- Profile at $c$'s arrival stop

In practice: very short linear scan. (Can be made constant, see later.)
Profile Connection Scan

For every connection \( c \) decreasing by departure time:

Profile at \( c \)'s departure stop

Profile at \( c \)'s arrival stop

Only pair needed for domination test.

Test if new jump is above or below existing one.
Profile Connection Scan

For every connection $c$ decreasing by departure time:

- Profile at $c$’s departure stop
- Profile at $c$’s arrival stop

Insert new jump if below. (In the example it is below.)
Profile Connection Scan

\( P(s) \) : profile of stop \( s \), an array of pairs;
\( T(t) \) : earliest arrival time of trip \( t \), a single timestamp;

\( P(s) \leftarrow \{(\infty, \infty)\} \) for every stop \( s \);
\( T(t) \leftarrow \infty \) for every trip \( t \);

for every connection \( c \) by decreasing \( c_{deptime} \)

\[
\begin{align*}
t & \leftarrow \text{evaluate} \ P(\text{c}_{\text{arrstop}}) \ \text{at} \ \text{c}_{\text{arrrtime}} + \text{minchange}(\text{c}_{\text{arrstop}}); \\
t & \leftarrow \min(t, T(\text{c}_{\text{tripid}})); \\
\text{if} \ \text{c}_{\text{arrstop}} = \text{target stop} \ \text{then} \\
& \quad t \leftarrow \min(t, \text{c}_{\text{arrrtime}}); \\
T(\text{c}_{\text{tripid}}) & \leftarrow t; \\
\text{if} \ t < P(\text{c}_{\text{depstop}})[\text{front}].\text{arrtime} \ \text{then} \\
& \quad \text{Insert} \ (\text{c}_{\text{deptime}}, \ t) \ \text{pair at the front of} \ P(\text{c}_{\text{depstop}}); \\
\end{align*}
\]
How to Evaluate Profiles?

Evaluating a profile at moment $t$ consists of finding the first pair $(d, a)$ such that $t \leq d$.

Option 1:

- $c_{arrtime} - c_{deptime}$ is small compared to the time horizon of the whole profile.
- A linear scan works very good in practice.
- Less than 2 pairs touched on average.
- Quasi $O(1)$. 
How to Evaluate Profiles?

Option 2: Modify the algorithm slightly. Replace:

```
if \( t < P(c_{\text{depstop}})[\text{front}]. \text{arrtime} \) then
  Insert \((c_{\text{deptime}}, t)\) pair at the front of \( P(c_{\text{depstop}}) \);
```

with

```
Insert \((c_{\text{deptime}}, \min(t, P(c_{\text{depstop}})[\text{front}]. \text{arrtime}))\) pair at the front of \( P(c_{\text{depstop}}) \);
```

- The departure times are now independent of the target stop.
- The same pairs with the same departure times (but different arrival times) are generated in the same order for each algorithm execution.
- In a quick preprocessing step: Store for each connection the position \( c_{\text{eval}} \) of the corresponding pair.
- True \( O(1) \) evaluation \( \rightarrow \) Complete running time in \( O(\#\text{connections}) \)
Profile Connection Scan

\[ P(s) : \text{profile of stop } s; \]
\[ T(t) : \text{earliest arrival time of trip } t, \text{ a single integer}; \]
\[ c_{\text{eval}} : \text{pair ID needed at evaluation, independent of the target stop}; \]

\[ P(s) \leftarrow \{ (\infty, \infty) \} \text{ for every stop } s; \]
\[ T(t) \leftarrow \infty \text{ for every trip } t; \]

\textbf{for every connection } c \textbf{ by decreasing } c_{\text{deptime}} \textbf{ do}

\[ t \leftarrow P(c_{\text{arrstop}})[c_{\text{eval}}]; \]
\[ t \leftarrow \min(t, T(c_{\text{tripid}})); \]
\[ \textbf{if } c_{\text{arrstop}} = \text{target stop } \textbf{ then} \]
\[ \quad t \leftarrow \min(t, c_{\text{arrtime}}); \]
\[ \quad T(c_{\text{tripid}}) \leftarrow t; \]
\[ \text{Insert } (c_{\text{deptime}}, \min(t, P(c_{\text{depstop}})[\text{front}].\text{arrtime})) \text{ pair} \]
\[ \text{at the front of } P(c_{\text{depstop}}); \]
Restricted Range

The input also contains:

- a minimum departure time $d$
- a maximum arrival time $a$

Optimization:

- Scan only connections $c$ with $d \leq c_{\text{deptime}} \leq a$.
- This subarray can be determined using two binary searches.
Restricted Range

The input also contains:
- a minimum departure time $d$
- a maximum arrival time $a$
- a source stop $s$
- a target stop $t$

Optimization:
- For every stop $x$ determine the earliest arrival time $\tau(x)$ from $s$ with source time $d$. This can be done using a basic connection scan, restricted to the same subrange.
- Before processing a connection $c$ in the profile scan check whether $\tau(c_{\text{depstop}}) \leq c_{\text{deptime}}$. If this does not hold skip the connection, as it can not be part of any desired journey.
The user wants to get from A to C.

Earliest arrival journeys:

1. $A \rightarrow B \rightarrow C$
2. $A \rightarrow B \rightarrow D \rightarrow E \rightarrow F \rightarrow B \rightarrow C$

The user wants journey 1.

However, journey 2 is also optimal according to arrival time.

→ One solution: Minimize Number of Transfers
Problems with Earliest Arrival Time

**Easy solution:** Chose among all earliest arrival journeys one with a minimum number of transfers.

**Algorithm modification:** Add a small constant to the arrival time returned by the $P(c_{arrstop})$ evaluation.

Good enough in most networks.
Problems with Earliest Arrival Time

Complex solution: Compute all Pareto-optimal journeys.

Pareto-optimal: A journey is Pareto-optimal if no other journey exists that is faster and uses fewer transfers.

Algorithm modification:

- We are only interested in journeys with at most $n$ train entries (i.e. $n - 1$ transfers).
- Replace all arrival times with $(a_0, a_1, a_2 \ldots a_n)$-tuples. $a_i$ indicates the earliest arrival time when entering at most $i$ trains.
- Inserting a pair consits of entering a train. → Shift all components, i.e., shift$(a_0, a_1, \ldots, a_n) = (\infty, a_0, a_1, \ldots, a_{n-1})$
- Minimum is replace with componentwise minimum.
Pareto Profile Connection Scan

\( P(s) \): profile of stop \( s \);
\( T(t) \): earliest arrival time of trip \( t \);
\( c_{\text{eval}} \): pair ID needed at profile evaluation;

\( P(s) \leftarrow \{ (\infty, (\infty, \infty \ldots \infty)) \} \) for every stop \( s \);
\( T(t) \leftarrow (\infty, \infty \ldots \infty) \) for every trip \( t \);

for every connection \( c \) by decreasing \( c_{\text{deptime}} \) do

\[
\begin{align*}
t &\leftarrow P(c_{\text{arrstop}})[c_{\text{eval}}]; \\
t &\leftarrow \min(t, T(c_{\text{tripid}})); \\
\text{if } c_{\text{arrstop}} = \text{target stop} \text{ then} & \\
\quad t &\leftarrow \min(t, (c_{\text{arrtime}}, c_{\text{arrtime}} \ldots c_{\text{arrtime}})); \\
\quad T(c_{\text{tripid}}) &\leftarrow t; \\
\quad \text{Insert } (c_{\text{deptime}}, \min(\text{shift}(t), P(c_{\text{depstop}})[\text{front}].\text{arrtime})) \text{ pair} & \\
\quad \text{at the front of } P(c_{\text{depstop}}); \\
\end{align*}
\]
SIMD/SSE

- SIMD : Single Instruction Multiple Data
- Special CPU instructions that work on fixed length vectors of integers (or floats) in a single CPU cycle.
- Available on all modern x86 processors.
- 128bit registers à 4×32 bit or à 8×16 bit integers.
- The x86 implementation is called SSE.
- SIMD is the general concept.
Without SSE:

```c
int a[8], b[8], c[8];
c[0] = a[0] + b[0];
c[1] = a[1] + b[1];
```

8 time units used

With SSE:

```c
__m128i a[2], b[2], c[2];
c[0] = _mm_add_epi32(a[0], b[0]);
c[1] = _mm_add_epi32(a[1], b[1]);
```

2 time units used

Speedup of 4 only observed for compute-bound algorithms.
**SIMD/SSE**

Without SSE:

```c
short a[8], b[8], c[8];
```

8 time units used

With SSE:

```c
__m128i a, b, c;
c = _mm_add_epi16(a, b);
```

1 time units used

Speedup of 8 only observed for compute-bound algorithms.
**SIMD/SSE**

Simple if-else-constructs are also supported.

Without SSE:

```c
short a[8], b[8], c[8];
for (int i = 0; i < 8; ++i)
    c[i] = a[i] < b[i] ? a[i] : b[i];
```

With SSE:

```c
__m128i a, b, c;
c = _mm_blendv_epi8(a, b, _mm_cmplt_epi16(a, b));
```

Or use the minimum instruction:

```c
__m128i a, b, c;
c = _mm_min_epi16(a, b);
```
The arrival time vector operations can all be done using SSE-instructions.
Time Compression

**Problem:** Algorithm is memory-bound. The memory is nearly completely filled with arrival times. → Compress arrival times.

**Observation:** Not at every second a train leaves and departs.

**Idea:** For every stop compute an ordered array $t_0, t_1, \ldots, t_n$ of the time points at which a train departs or arrives.

- Indices respect time order, i.e., $t_i < t_j \iff i < j$.
- Often the indices fit into 16 bit integers.
- Propagate the indices instead of the time points.
Problems with Delays

<table>
<thead>
<tr>
<th>Station/Stop</th>
<th>Date</th>
<th>Time</th>
<th>Platform</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karlsruhe Hbf</td>
<td>Fr, 31.05.13</td>
<td>dep 08:00</td>
<td>2</td>
<td>ICE 5, Intercity-Express</td>
</tr>
<tr>
<td>Roma Termini</td>
<td>Fr, 31.05.13</td>
<td>arr 18:30</td>
<td>10</td>
<td>Bordrestaurant</td>
</tr>
<tr>
<td>Zürich HB</td>
<td>Fr, 31.05.13</td>
<td>dep 11:09</td>
<td>5</td>
<td>Eurocity, Subject to compulsory reser available</td>
</tr>
<tr>
<td>Milano Centrale</td>
<td>Fr, 31.05.13</td>
<td>arr 14:50</td>
<td>5</td>
<td>EuroStar Italia, Subject to compulsory reser wheelchairs</td>
</tr>
</tbody>
</table>

Transfer time 9 min.

Transfer time 20 min.

Connecting train may not be reached in time.

composed out of several screenshots of bahn.de, specific situation was not observed
Problems with Delays

composed out of several screenshots of bahn.de, specific situation was not observed

25 min delay vs 20 min transfer
Problems with Delays

<table>
<thead>
<tr>
<th>Station/Stop</th>
<th>Date</th>
<th>Time</th>
<th>Platform</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karlsruhe Hbf</td>
<td>Fr, 31.05.13</td>
<td>dep 08:00</td>
<td>2</td>
<td>ICE 5</td>
</tr>
<tr>
<td></td>
<td>Fr, 31.05.13</td>
<td>arr 11:00</td>
<td>10</td>
<td>Intercity-Express</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bordrestaurant</td>
</tr>
<tr>
<td>Zürich HB</td>
<td>Fr, 31.05.13</td>
<td>dep 11:09</td>
<td>5</td>
<td>EC 17</td>
</tr>
<tr>
<td>Milano Centrale</td>
<td>Fr, 31.05.13</td>
<td>arr 14:30</td>
<td></td>
<td>Eurocity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subject to compulsory reser available</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>→ Adjust the transfer time</td>
</tr>
<tr>
<td>Roma Termini</td>
<td>Fr, 31.05.13</td>
<td>dep 15:10</td>
<td></td>
<td>ES 9541</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EuroStar Italia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subject to compulsory reser wheelchairs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>→ Adjust the transfer time</td>
</tr>
</tbody>
</table>

Transfer time 9 min.

Connect train may not be reached in time.

What if 9 min are not enough?

composed out of several screenshots of bahn.de, specific situation was not observed
Problems with Delays

<table>
<thead>
<tr>
<th>Station/Stop</th>
<th>Date</th>
<th>Time</th>
<th>Platform</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karlsruhe Hbf</td>
<td>Fr, 31.05.13</td>
<td>dep 08:00</td>
<td>2</td>
<td>ICE 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>arr 18:30</td>
<td></td>
<td>Intercity-Express</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bordrestaurant</td>
</tr>
<tr>
<td>Zürich HB</td>
<td>Fr, 31.05.13</td>
<td>dep 11:00</td>
<td>10</td>
<td>Intercity-Express</td>
</tr>
<tr>
<td></td>
<td></td>
<td>arr 11:05</td>
<td></td>
<td>Eurocity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subject to compulsory reser available</td>
</tr>
<tr>
<td>Milano Centrale</td>
<td>Fr, 31.05.13</td>
<td>dep 14:00</td>
<td>5</td>
<td>EC 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>arr 14:00</td>
<td></td>
<td>Eurocity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subject to compulsory reser available</td>
</tr>
<tr>
<td>Roma Termini</td>
<td>Fr, 31.05.13</td>
<td>dep 18:30</td>
<td>ES 9541</td>
<td>Eurostar Italia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subject to compulsory reser wheelchairs</td>
</tr>
</tbody>
</table>

- Transfer time 9 min.
- Transfer time 20 min.
- Connecting train may not be reached in time.

... but perhaps they are enough ...

→ Backup journeys are needed
If all goes well
9 min change time → likely to fail
The backup journey
Tight change on the backup → Backup for the backup needed
Decision Graph

Karlsruhe 8:00


Zurich 11:00 11:09 11:34 14:26 15:09

Milano 14:50 15:20 16:38 17:20 18:50 21:10

Roma 18:45 20:45 29:51

Genova 22:42 23:53

Backup of the Backup
Delay Model

Assumptions:
- Connections arrive with a random delay.
- Connections have a maximum delay.
- Distributions are known.
- All random variables are independent.
- Connections always depart on time.

\[
\begin{align*}
\text{c's departure time} & \\
\text{c's departure stop} & \\
\text{c's on time arrival time} & \\
\text{c's latest arrival time} & \\
\text{c's arrival stop} &
\end{align*}
\]
Expected Arrival Time

- $e_0 \ldots e_4$: expected arrival times
- $t_1 \ldots t_6$: fixed points in time
  - blue: in the decision graph
  - red: not in the decision graph
Expected Arrival Time

$t$: actual arrival time

$$e_0 = P(t_1 \leq t \leq t_2) \cdot e_1 + P(t_2 \leq t \leq t_5) \cdot e_4$$
Expected Arrival Time

\[ e_0 = P(t_1 \leq t \leq t_3) \cdot e_2 + P(t_3 \leq t \leq t_5) \cdot e_4 \]

t: actual arrival time
Minimum Expected Arrival Time (MEAT)

Computing Decision Graphs

- **Input:** Timetable, delay probabilities, target stop
- **Output:** Subset of connections with *minimum expected arrival time* for every source stop and time
Profile maps departure time onto MEAT at target.
Profile maps departure time onto MEAT at target.
Scan a bit further to collect all relevant trains.
This scan is not $O(1)$. 

**MEAT Connection Scan**

- Profile at $c$'s departure stop
- Profile at $c$'s arrival stop

$c$'s departure time

$c$'s arrival interval
Average weighted by probability of a getting train.
Inserting is $O(1)$ just as before.
Running Times

London instance with 4,850,431 connections.

Earliest Arrival One-to-One:
- Time-Expanded: 64.4 ms
- Time-Dependent: 10.9 ms
- Connection Scan: 2.0 ms

Earliest Arrival One-to-All:
- Time-Expanded: 876.2 ms
- Time-Dependent: 18.9 ms
- Connection Scan: 9.7 ms

(Time-Dependent can be made slightly faster, with ideas not covered in this course.)
Running Times

Non-Pareto Profile All-to-One:

- Self-Pruning-Connection-Setting : 1 262 ms
- Connection Scan: 177 ms
- + constant eval: 134 ms
- + time compress: 104 ms

Pareto Profile All-to-One (journeys with at most 8 trains):

- RAPTOR : 1 179 ms
- Connection Scan: 255 ms
- + SSE: 221 ms

MEAT: 272 ms