Mathematical Aspects of Public Transportation Networks

Niels Lindner

April 23, 2018
Chapter 1

S-Bahn Challenge

§1.3 The Traveling Salesman Problem
Interlude: P vs. NP

Informal definitions

- A decision problem is a problem whose solution is either yes or no.
- The complexity class P consists of all decision problems that can be solved in polynomial time.
- The complexity class NP consists of all decision problems that can be verified in polynomial time.

P vs. NP

The question whether \( P = NP \) is a millennium problem.

Notation

For a decision problem \( \Pi \) with an input \( x \), we write \( x \in \Pi \) iff \( x \) is a “yes”-instance for \( \Pi \).
How to show membership to P or NP

Let Π be a decision problem.

- Π ∈ P ⇔ ∃ polynomial \( p \) and an algorithm \( A \) that decides for each input \( x \) if \( x \in Π \), and the running time of \( A \) is \( \leq p(\text{size}(x)) \).
- Π ∈ NP ⇔ ∃ polynomial \( p \) and a problem \( Λ \in P \) such that each input \( x \) has a certificate \( c(x) \) satisfying \( x \in Π ⇔ (x, c(x)) \in Λ \), and \( \text{size}(c(x)) \leq p(\text{size}(x)) \).

Examples

- “Does a graph \( G \) admit an Euler tour?” is in P.
- “Is a graph \( G \) Hamiltonian?” is in NP.
  (certificate: a Hamiltonian circuit \( C \))
- “Is a graph \( G \) not Hamiltonian?” is not known to be in NP.
  (certificate: all circuits in \( G \) – too large!)
1.3 The Traveling Salesman Problem

Polynomial-time reduction

Definition
Let \( \Pi \) and \( \Lambda \) be decision problems. \( \Pi \) reduces polynomially to \( \Lambda \) (short: \( \Pi \leq \Lambda \)) if there is a function \( f \) on the inputs for \( \Pi \) such that

\[
x \in \Pi \Leftrightarrow f(x) \in \Lambda,
\]

and \( f \) can be computed by a polynomial-time algorithm.

Remarks
- This is a partial order.
- Intuitively, \( \Pi \leq \Lambda \) if and only if \( \Pi \) is at most as hard to solve as \( \Lambda \).
- If \( \Pi \leq \Lambda \) and \( \Lambda \leq \Pi \), then \( \Pi \) and \( \Lambda \) are polynomially equivalent.

Lemma
- \( \Pi \in P \Leftrightarrow \Pi \leq \Lambda \ for \ some \ \Lambda \in P. \)
- \( \Pi \in NP \Leftrightarrow \Pi \leq \Lambda \ for \ some \ \Lambda \in NP. \)
NP-completeness

Definition
Let $\Pi$ be a decision problem.

- $\Pi$ is **NP-hard** if $\Lambda \leq \Pi$ for each $\Lambda \in \text{NP}$.
- $\Pi$ is **NP-complete** if $\Pi$ is NP-hard and $\Pi \in \text{NP}$.

Lemma (How to show NP-hardness)
*Suppose there is an NP-hard problem $\Lambda$ with $\Lambda \leq \Pi$. Then $\Pi$ is NP-hard.*

Optimization problems
We also call a minimization problem $\min_{x \in X} f(x)$ **NP-hard/-complete** if the decision problem

"Given $q \in \mathbb{Q}$, is there an $x \in X$ with $f(x) \leq q$?"

is NP-hard/-complete. (Similar: maximization with “$\geq$”.)
1.3 The Traveling Salesman Problem

Complete Graphs

Let $n \in \mathbb{N}$. The complete graph $K_n$ is the graph with

- vertex set $V(K_n) = \{1, \ldots, n\}$,
- edge set $E(K_n) = \{\{i, j\} \mid 1 \leq i < j \leq n\}$.

**Definition**

The **Traveling Salesman Problem (TSP)** on a complete graph $K_n$ is to find a minimum-cost Hamilton circuit in $K_n$ w.r.t. a cost function $c : E(K_n) \to \mathbb{R}_{\geq 0}$.

<table>
<thead>
<tr>
<th>distance/km</th>
<th>HH</th>
<th>K</th>
<th>S</th>
<th>M</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamburg</td>
<td>0</td>
<td>366</td>
<td>534</td>
<td>613</td>
<td>256</td>
</tr>
<tr>
<td>Köln</td>
<td>366</td>
<td>0</td>
<td>288</td>
<td>456</td>
<td>478</td>
</tr>
<tr>
<td>Stuttgart</td>
<td>534</td>
<td>288</td>
<td>0</td>
<td>191</td>
<td>512</td>
</tr>
<tr>
<td>München</td>
<td>613</td>
<td>456</td>
<td>191</td>
<td>0</td>
<td>505</td>
</tr>
<tr>
<td>Berlin</td>
<td>256</td>
<td>478</td>
<td>512</td>
<td>505</td>
<td>0</td>
</tr>
</tbody>
</table>
Let $n \in \mathbb{N}$. The **complete graph** $K_n$ is the graph with

- vertex set $V(K_n) = \{1, \ldots, n\}$,
- edge set $E(K_n) = \{\{i, j\} \mid 1 \leq i < j \leq n\}$.

**Definition**

The **Traveling Salesman Problem (TSP)** on a complete graph $K_n$ is to find a minimum-cost Hamilton circuit in $K_n$ w.r.t. a cost function $c : E(K_n) \to \mathbb{R}_{\geq 0}$.

---

**optimal cost: 1606**
The Traveling Salesman Problem

Hardness

Theorem

*TSP is NP-hard.*

Proof.

Let $G$ be a graph on $n$ vertices with edge set $E(G)$. Define a cost function on $E(K_n)$ via

$$c(\{i,j\}) := \begin{cases} 1 & \text{if } \{i,j\} \in E(G), \\ 2 & \text{otherwise,} \end{cases} \quad 1 \leq i < j \leq n.$$ 

Then $G$ contains a Hamiltonian circuit if and only if $K_n$ has a Hamiltonian circuit with cost $\leq n$. \hfill $\square$

Combinatorial Explosion

$K_n$ contains $(n - 1)!/2$ Hamilton circuits.
Approximation hardness

Definition
Let $P$ be an optimization problem with non-negative cost and $k \geq 1$. A $k$-factor approximation algorithm for $P$ is a polynomial-time algorithm $A$ for $P$ such that

$$\frac{1}{k} \cdot \text{OPT}(I) \leq A(I) \leq k \cdot \text{OPT}(I)$$

for all instances $I$ of $P$. Here, $\text{OPT}(I)$ denotes the cost of an optimal solution, and $A(I)$ is the cost of the solution computed by $A$.

A $k$-factor approximation algorithm is a polynomial-time heuristic with a worst-case estimate on the solution quality (the lower $k$, the better).

Theorem
Let $A$ be a $k$-factor approximation algorithm for TSP for some $k \geq 1$. Then $P = NP$. 
1.3 The Traveling Salesman Problem

Approximation hardness

Proof.

- Let $A$ be such an algorithm, i.e., for every TSP instance $I = (K_n, c)$ with optimal solution $\text{OPT}(I)$, $A$ computes a Hamiltonian circuit of cost $A(I) \leq k \cdot \text{OPT}(I)$.

- Let $G$ be a graph with edge set $E(G)$ and $n$ vertices. Define a cost function on $E(K_n)$ via

$$c(\{i, j\}) := \begin{cases} 1 & \text{if } \{i, j\} \in E(G), \\ 2 + (k - 1)n & \text{otherwise}, \end{cases} \quad 1 \leq i < j \leq n.$$  

- If $A(I) \leq n$, then $G$ admits a Hamiltonian circuit.

- Otherwise $k \cdot \text{OPT}(I) \geq A(I) \geq n - 1 + 2 + (k - 1)n = kn + 1$, thus $\text{OPT}(I) > n$ and $G$ cannot have a Hamiltonian circuit.

- $A$ is a polynomial-time algorithm deciding the NP-complete Hamilton circuit problem on an arbitrary graph. This implies $P = NP$. 

April 23, 2018
§1.3 The Traveling Salesman Problem

Metric TSP

Definition
A TSP instance \((K_n, c)\) is called **metric** if the triangle inequality
\[ c(\{i, j\}) \leq c(\{i, k\}) + c(\{k, j\}) \]
holds for all \(1 \leq i, j, k \leq n\).

Theorem (Christofides, 1976)
*There is a \(\frac{3}{2}\)-factor approximation algorithm for metric TSP.*

Christofides’ algorithm

1. Compute a minimum spanning tree \(T\) in \(K_n\) w.r.t. \(c\).
2. Find a min-weight perfect matching \(M\) of the odd-degree vertices of \(T\) w.r.t. \(c\).
3. Take the Hamiltonian circuit by sorting the vertices by order of appearance in an Euler tour in \((V(K_n), E(T) \cup M)\).
1.3 The Traveling Salesman Problem

**Metric TSP**

**Definition**

A TSP instance \((K_n, c)\) is called **metric** if the triangle inequality
\[ c(\{i, j\}) \leq c(\{i, k\}) + c(\{k, j\}) \]
holds for all \(1 \leq i, j, k \leq n\).

**Theorem (Christofides, 1976)**

*There is a \(\frac{3}{2}\)-factor approximation algorithm for metric TSP.*

**Christofides’ algorithm**

1. Compute a minimum spanning tree \(T\) in \(K_n\) w.r.t. \(c\).
2. Find a min-weight perfect matching \(M\) of the odd-degree vertices of \(T\) w.r.t. \(c\).
3. Take the Hamiltonian circuit by sorting the vertices by order of appearance in an Euler tour in \((V(K_n), E(T) \cup M)\).
1.3 The Traveling Salesman Problem

Metric TSP

Definition
A TSP instance \((K_n, c)\) is called metric if the triangle inequality
\[ c(\{i, j\}) \leq c(\{i, k\}) + c(\{k, j\}) \]
holds for all \(1 \leq i, j, k \leq n\).

Theorem (Christofides, 1976)
There is a \(\frac{3}{2}\)-factor approximation algorithm for metric TSP.

Christofides’ algorithm

1. Compute a minimum spanning tree \(T\) in \(K_n\) w.r.t. \(c\).
2. Find a min-weight perfect matching \(M\) of the odd-degree vertices of \(T\) w.r.t. \(c\).
3. Take the Hamiltonian circuit by sorting the vertices by order of appearance in an Euler tour in \((V(K_n), E(T) \cup M)\).
1.3 The Traveling Salesman Problem

Christofides’ Algorithm

Proof.

- Let $I = (K_n, c)$ be a TSP instance. Removing a single edge from any Hamilton circuit gives a spanning tree. Hence for a minimum spanning tree $T$ of $K_n$ w.r.t. $c$, we have $\text{OPT}(I) \geq c(T) := \sum_{e \in E(T)} c(e)$.

- A shortest path from $i$ to $j$ is simply given by the edge $\{i, j\}$ because of the triangle inequality.

- Denote by $c(M)$ the weight of the min-weight perfect matching $M$. Each Hamiltonian circuit decomposes into two matchings of the odd-degree nodes of $T$. Hence $\text{OPT}(I) \geq 2c(M)$ (triangle inequality).

- The graph $(V(K_n), E(T) \cup M)$ is clearly Eulerian.

- Computing a Hamiltonian circuit from an Euler tour does not increase the cost (again triangle inequality).

- Thus $A(I) \leq c(T) + c(M) \leq \text{OPT}(I) + \frac{1}{2}\text{OPT}(I) = \frac{3}{2}\text{OPT}(I)$.

- The algorithm runs in polynomial time.
The $k$-opt heuristic

For non-metric TSP instances $I = (K_n, c)$, there is a family of heuristics based on local search:

$k$-opt heuristic

Fix an integer $k \geq 2$.

1. Let $C$ be any Hamiltonian circuit.
2. Let $S$ be the collection of all $k$-element subsets of $E(C)$.
3. Let $C' := \arg \min \{ c(C') \mid C' \text{ Ham. circuit, } E(C) \setminus S \subseteq E(C'), S \in S \}$.
4. If $c(C') < c(C)$, set $C := C'$ and go to 2. Otherwise return $C'$.

Remarks

- For all $k \geq 2$, the worst-case running time is exponential in $n$.
- $n$-opt would be exact, but enumerates all possibilities.
- In Step 3, 2-opt simply replaces two edges $(i, j), (k, \ell)$ by $(i, k), (j, \ell)$. 
The TSP on \((K_n, c)\) has the following classical formulation as an IP:

\[
\begin{align*}
\text{Minimize} & \quad \sum_{e \in E(K_n)} c(e)x_e \\
\text{s. t.} & \quad \sum_{e \in E(K_n) : v \in e} x_e = 2, \quad \forall v \in V(K_n), \\
& \quad \sum_{e \in E(K_n) : e \in S \times S} x_e \leq |S| - 1, \quad \emptyset \subsetneq S \subsetneq V(K_n), \\
& \quad x_e \in \{0, 1\}, \quad e \in E(K_n).
\end{align*}
\]

The second constraint is called *subtour elimination* constraint. It excludes solutions that are unions of disjoint circuits. Unfortunately, there are exponentially many of those.
1.3 The Traveling Salesman Problem

Separating Subtour Constraints

Theorem

Let \( x \in [0, 1]^{E(K_n)} \) satisfy \( \sum_{e: v \in e} x_e = 2 \). Then there is a polynomial-time algorithm that decides if there is a subset \( \emptyset \subsetneq S \subsetneq V(K_n) \) such that \( x \) violates the subtour elimination constraint w.r.t. \( S \).

Proof.
Tutorial.

This yields the following IP-based solution method:

1. Let \( S := \emptyset \).
2. Solve the IP with subtour elimination constraints only for \( S \in S \).
3. If the optimal solution violates the constraint for some \( S \), add it to \( S \). Otherwise, an optimal solution is found.

There are also IP formulations for the TSP with a polynomial number of constraints, but they have weaker LP relaxations and are hence harder for IP solvers.
1.3 The Traveling Salesman Problem

Solving TSP: Summary

Heuristics

- Metric TSP: Christofides’ $\frac{3}{2}$-factor approximation algorithm
- Local search: 2-opt, 3-opt, Lin-Kernighan (combines both, implementation: LKH)

Exact algorithms

- Integer programming: Branch-and-cut (implementation: concorde)
- Dynamic programming: Held-Karp $O(2^n n^2)$ algorithm

TSP Record

In 2006, conorde computed a solution for a TSP instance on 85,900 vertices, and proved optimality. LKH can solve this instance as well nowadays, but cannot provide lower bounds.
1.3 The Traveling Salesman Problem

Directed graphs

Let $n \in \mathbb{N}$. The complete directed graph $K_n^*$ is the digraph with

- vertex set $V(K_n^*) = \{1, \ldots, n\}$,
- edge set $E(K_n^*) = \{(i, j) \mid 1 \leq i \neq j \leq n\}$.

Definition

The Asymmetric Traveling Salesman Problem (ATSP) on $K_n^*$ is to find a minimum-cost directed Hamiltonian circuit w.r.t. a cost function $c : E(K_n^*) \to \mathbb{R}_{\geq 0}$.

Remarks

- If $c(i, j) = c(j, i)$ for all $1 \leq i \neq j \leq n$, then the problem is called symmetric and is equivalent to the TSP on the undirected complete graph $K_n$ with cost function $c(\{i, j\}) := c(i, j)$.
- ATSP is NP-complete.
Theorem (Jonker-Volgenant, 1983)

TSP is polynomially equivalent to ATSP.

Proof.

- Clearly, any TSP instance can be transformed into an ATSP instance by replacing each undirected edge \( \{i, j\} \) with cost \( c(\{i, j\}) \) by the two anti-parallel edges \((i, j)\) and \((j, i)\), and setting \( c(i, j) := c(j, i) := c(\{i, j\}) \).

- Conversely, let \( I = (K^*_n, c) \) be an ATSP instance. Create a TSP instance \( I' = (K_{2n}, c') \) as follows: For \( i \in V(K^*_n) \), let \( i^+ := 2i \) and \( i^- := 2i - 1 \). Set

\[
\begin{align*}
    c'_{\{i^+, j^-\}} &:= c(i, j) + M, \\
    c'_{\{i^-, i^+\}} &:= 0, \\
    c' &:= (n + 1)M + 1 \\
\end{align*}
\]

and let \( c' \) have value \((n + 1)M + 1\) on all other edges.
1.3 The Traveling Salesman Problem

Asymmetric TSP

Proof.

- Then any directed Hamiltonian circuit \((i_1, \ldots, i_n, i_1)\) in \(K^*_n\) yields a Hamiltonian circuit \((i_1^+, i_2^-, i_2^+, \ldots, i_n^-, i_n^+, i_1^-)\) in \(K_{2n}\), the cost increases by \(n \cdot M\). This shows \(\text{OPT}(I) + n \cdot M \geq \text{OPT}(I')\).

- Let \(C'\) be the optimal solution to \(I'\). Suppose \(M > \text{OPT}(I)\). Then \(C'\) contains all \(n\) edges \(i^- \rightarrow i^+\), as otherwise

\[
\text{OPT}(I') \geq (n+1)M = n \cdot M + M > n \cdot M + \text{OPT}(I).
\]

- Moreover, \(C'\) contains none of the \((n+1)M + 1\) cost edges, because otherwise also

\[
\text{OPT}(I') \geq (n+1)M + 1 > n \cdot M + \text{OPT}(I).
\]

- Hence \(C'\) can be transformed to a Hamiltonian circuit in \(K^*_n\), the cost decreasing by \(n \cdot M\). Thus \(\text{OPT}(I) + n \cdot M = \text{OPT}(I')\).

- Take e.g. \(M := 1 + \text{sum of } n\) heaviest edges of \(K^*_n\) w.r.t. \(c\).
Asymmetric TSP

Summary

- A TSP instance on $n$ nodes can be transformed into an ATSP instance on $n$ nodes, with the same optimal cost.
- An ATSP instance on $n$ nodes can be transformed into a TSP instance on $2n$ nodes, the cost increasing by $n \cdot M$ for a large $M$. 

1.3 The Traveling Salesman Problem

**General undirected graphs**

Let $G = (V, E)$ be a not necessarily complete undirected graph with a cost function $c : E \to \mathbb{R}_{\geq 0}$.

**Definition**

- A **Traveling Salesman tour** is a closed walk $(e_1, \ldots, e_k)$ in $G$ such that every vertex in $G$ is visited at least once.
- The **Traveling Salesman Problem (TSP)** is to find a Traveling Salesman tour $(e_1, \ldots, e_k)$ of minimum cost $\sum_{i=1}^{k} c(e_i)$.

**Lemma**

*If $G$ is Hamiltonian and $c$ satisfies the triangle inequality, then the optimal TSP solution is one of the Hamiltonian circuits of $G$.*

In particular, it is important to know if TSP refers to the “exactly once” or “at least once” version.
§1.3 The Traveling Salesman Problem

Reducing $\geq 1$ to $= 1$

Let $G$ be an undirected graph on $n$ nodes, $c : E(G) \rightarrow \mathbb{R}_{\geq 0}$ a cost function.

Theorem

The “at least once” TSP on $G$ w.r.t. $c$ can be polynomially transformed to an “exactly once” metric TSP instance on $K_n$ with the same optimal cost.

Proof.

1. Let $v_1, \ldots, v_n$ denote the vertices of $G$. For $1 \leq i < j \leq n$, set
   \[ c'(\{i, j\}) := \text{length of shortest path from } v_i \text{ to } v_j \text{ in } G \text{ w.r.t. } c. \]

2. The optimal Hamiltonian circuit on $(K_n, c')$ produces a closed walk in $G$ by transforming $i \rightarrow j$ to the shortest path from $v_i \rightarrow v_j$. The cost does not change, hence $\text{OPT}(G, c) \leq \text{OPT}(K_n, c').$

3. The optimal TSP tour in $G$ w.r.t. $c$. gives a Hamiltonian circuit in $K_n$ by sorting the vertices in their order of appearance. Since $c'$ consists of the shortest distances, we have $\text{OPT}(G, c) \geq \text{OPT}(K_n, c').$
1. Compute all shortest paths (e.g., using Floyd-Warshall).
1.3 The Traveling Salesman Problem

**Example: Amsterdam metro**

1. Compute all shortest paths (e.g., using Floyd-Warshall).

```
<table>
<thead>
<tr>
<th></th>
<th>Amsterdam Zuid</th>
<th>Centraal Station</th>
<th>Gaasperplas</th>
<th>Gein</th>
<th>Isolatorweg</th>
<th>Overamstel</th>
<th>Spaklerweg</th>
<th>van der Madeweg</th>
<th>Westwijk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amsterdam Zuid</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>4</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Centraal Station</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gaasperplas</td>
<td>16</td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gein</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Isolatorweg</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Overamstel</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Spaklerweg</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>van der Madeweg</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Westwijk</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
1.3 The Traveling Salesman Problem

Example: Amsterdam metro

<table>
<thead>
<tr>
<th></th>
<th>Amsterdam Zuid</th>
<th>Centraal Station</th>
<th>Gaasperplas</th>
<th>Gein</th>
<th>Isolatorweg</th>
<th>Overamstel</th>
<th>Spaklerweg</th>
<th>van der Madeweg</th>
<th>Westwijk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amsterdam Zuid</td>
<td>0</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>16</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>Centraal Station</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>22</td>
<td>32</td>
<td>12</td>
<td>9</td>
<td>11</td>
<td>41</td>
</tr>
<tr>
<td>Gaasperplas</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>21</td>
<td>33</td>
<td>13</td>
<td>12</td>
<td>10</td>
<td>42</td>
</tr>
<tr>
<td>Gein</td>
<td>0</td>
<td>0</td>
<td>34</td>
<td>0</td>
<td>20</td>
<td>14</td>
<td>13</td>
<td>11</td>
<td>43</td>
</tr>
<tr>
<td>Isolatorweg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>34</td>
<td>34</td>
<td>3</td>
<td>13</td>
<td>11</td>
<td>43</td>
</tr>
<tr>
<td>Overamstel</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>Spaklerweg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>van der Madeweg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>Westwijk</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>

1. Compute all shortest paths (e.g., using Floyd-Warshall).
2. Solve the TSP on the complete graph (20,160 Hamiltonian circuits).
1. Compute all shortest paths (e.g., using Floyd-Warshall).
2. Solve the TSP on the complete graph (20,160 Hamiltonian circuits).
3. Trace back the shortest paths.
1.3 The Traveling Salesman Problem

**Example: Amsterdam metro**

<table>
<thead>
<tr>
<th></th>
<th>Amsterdam Zuid</th>
<th>Centraal Station</th>
<th>Gaasperplas</th>
<th>Gein</th>
<th>Isolatorweg</th>
<th>Overamstel</th>
<th>Spaklerweg</th>
<th>van der Madeweg</th>
<th>Westwijk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amsterdam Zuid</td>
<td>0</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>16</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>Centraal Station</td>
<td>0</td>
<td>21</td>
<td>22</td>
<td>32</td>
<td>12</td>
<td>9</td>
<td>11</td>
<td>41</td>
<td>44</td>
</tr>
<tr>
<td>Gaasperplas</td>
<td>0</td>
<td>21</td>
<td>33</td>
<td>13</td>
<td>12</td>
<td>10</td>
<td>13</td>
<td>42</td>
<td>43</td>
</tr>
<tr>
<td>Gein</td>
<td>0</td>
<td>0</td>
<td>34</td>
<td>14</td>
<td>13</td>
<td>11</td>
<td>32</td>
<td>41</td>
<td>29</td>
</tr>
<tr>
<td>Isolatorweg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>23</td>
<td>33</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Overamstel</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Spaklerweg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>van der Madeweg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Westwijk</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1. Compute all shortest paths (e.g., using Floyd-Warshall).
2. Solve the TSP on the complete graph (20,160 Hamiltonian circuits).
3. Trace back the shortest paths.

Result: The optimal TSP tour is identical to the optimal CPP tour.
Chapter 1

S-Bahn Challenge

§1.4 Generalized Routing Problems
\section*{1.4 Generalized Routing Problems}

**GATSP and GDRPP**

Let $G = (V, E)$ be a directed graph with a cost function $c : E \to \mathbb{R}_{\geq 0}$.

**Definition**

Let $V_1, \ldots, V_k$ be disjoint subsets of $V$ (clusters). The **Generalized Asymmetric Traveling Salesman Problem (GATSP)** is to find a directed closed walk $C = (e_1, \ldots, e_k)$ in $G$ such that

- $C$ visits at least one vertex from each cluster at least once,
- $C$ has minimal cost w.r.t. $c$.

**Definition**

Let $E_1, \ldots, E_k$ be disjoint subsets of $E$ (clusters). The **Generalized Directed Rural Postman Problem (GDRPP)** is to find a directed closed walk $C = (e_1, \ldots, e_k)$ in $G$ such that

- $C$ visits at least one edge from each cluster at least once,
- $C$ has minimal cost w.r.t. $c$.

We will model the S-Bahn Challenge problem as GDRPP.
1.4 Generalized Routing Problems

GATSP and GDRPP: Example

GATSP instance with 2 clusters

GDRPP instance with 2 clusters

an optimal GATSP tour of cost 24

an optimal GDRPP tour of cost 28
GATSP and GDRPP: Equivalence

Theorem (Drexl, 2007)

GATSP and GDRPP are polynomially equivalent.

Proof (GATSP \( \leq \) GDRPP).

Let \( I = (G, c, \{V_1, \ldots, V_k\}) \) be a GATSP instance. Set

\[
E_i := \{(v, w) \mid v \in V_i, w \notin V_i\}, \quad i = 1, \ldots, k,
\]

and define a GDRPP instance \( I' := (G, c, \{E_1, \ldots, E_k\}) \).

For each \( i \), any solution to the GDRPP on \( I' \) visits at least one edge of \( E_i \), and hence at least one vertex of \( V_i \). We conclude

\[ \text{OPT}(I') \geq \text{OPT}(I). \]

Conversely, any solution to the GATSP on \( I \) visits at least one edge of \( E_i \), because \( E_i \) comprises all outgoing edges from \( V_i \). Hence

\[ \text{OPT}(I') \leq \text{OPT}(I). \]
1.4 Generalized Routing Problems

GATSP and GDRPP: Equivalence

Proof \((\text{GDRPP} \leq \text{GATSP})\).

- Let \(I' = (G, c, \{E_1, \ldots, E_k\})\) be a GDRPP instance. Split each edge \(e = (v, w) \in \bigcup_{i=1}^{k} E_i\) by a new vertex \(z_e\). That is, remove \(e\), and add the edges \((v, z_e)\) and \((z_e, w)\) with cost \(c(e)\) and 0, respectively. Set

\[
V_i := \{z_e \mid e \in E_i\}, \quad i = 1, \ldots, k.
\]

and define a GATSP instance \(I := (G, c, \{V_1, \ldots, V_k\})\).

- Any solution to the GDRPP on \(I'\) visits at least one edge \(e_i \in E_i\) for all \(i\), and hence gives rise to a GATSP solution visiting at least one vertex \(z_{e_i}\) for all \(i\). The cost does not change, thus \(\text{OPT}(I') \geq \text{OPT}(I)\).

- Conversely, any solution to the GATSP on \(I\) visits at least one vertex \(z_{e_i} \in V_i\) for all \(i\), and yields a GDRPP solution visiting at least one edge \(e_i\) for all \(i\). Therefore \(\text{OPT}(I') \leq \text{OPT}(I)\).
GATSP and GDRPP: Equivalence Example

GATSP instance

GDRPP instance

equivalent GDRPP instance

equivalent GATSP instance
1.4 Generalized Routing Problems

GATSP and GDRPP: NP-hardness

**Theorem**

*GATSP and GDRPP are NP-hard.*

**Proof.**

- It suffices to show NP-hardness for GDRPP. We already know that the Rural Postman Problem (RPP) is NP-hard.
- Let \( G = (V, E) \) be an undirected graph with a cost function \( c : E \to \mathbb{R}_{\geq 0} \), and let \( S \subseteq E \) be a subset of edges. The RPP is to find a closed walk \((e_1, \ldots, e_k)\) in \( G \) covering \( S \) of minimal cost w.r.t. \( c \).
- Let \( D \) be the digraph obtained from \( G \) where each undirected edge \( \{v, w\} \) is replaced by two anti-parallel edges \((v, w), (w, v)\). Extend the cost function \( c \) to \( D \) by defining \( c(v, w) := c(w, v) := c(\{v, w\}) \). For each edge \( e = \{v, w\} \in S \), add a cluster \( E_e = \{(v, w), (w, v)\} \).
- The GDRPP on \((D, c, \{E_e \mid e \in S\})\) is equivalent to the RPP on \((G, c, S)\).
Theorem (Noon/Bean, 1991)

\[\text{GATSP} \leq \text{ATSP}.\]

Proof.

- Let \( I = (G, c, \{V_1, \ldots, V_k\}) \) be an arbitrary GATSP instance, let \( n = \sum_{i=1}^{k} |V_i| \). We will define an ATSP instance \( I' = (K_n^*, c') \).
- For each \( i = 1, \ldots, k \), choose any ordering \((v_{i,1}, v_{i,2}, \ldots, v_{i,r_i})\) of \( V_i \).
- Set \( M := 1 + \text{sum of lengths of the } k \text{ longest shortest paths in } G \) and 
  \[c'(v_{i,1}, v_{i,2}) := c'(v_{i,2}, v_{i,3}) := \cdots := c'(v_{i,r_i}, v_{i,1}) := 0,\]
  \[c'(v_{i,j}, v_{p,q}) := M + \text{shortest path length from } v_{i,(j+1) \mod r_i} \text{ to } v_{p,q} \text{ in } G\]
  for all \( i \) resp. all \((i, j), (p, q)\) with \( i \neq p \).
- All other edges receive cost \( M \).
§1.4 Generalized Routing Problems

GATSP and ATSP: Example

GATSP instance

equivalent ATSP instance

<table>
<thead>
<tr>
<th>$c'$</th>
<th>1,1</th>
<th>1,2</th>
<th>1,3</th>
<th>2,1</th>
<th>2,2</th>
<th>2,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>$M$</td>
<td>0</td>
<td>$M$</td>
<td>$M + 11$</td>
<td>$M + 14$</td>
<td>$M + 15$</td>
</tr>
<tr>
<td>1,2</td>
<td>0</td>
<td>$M$</td>
<td>0</td>
<td>$M + 12$</td>
<td>$M + 15$</td>
<td>$M + 16$</td>
</tr>
<tr>
<td>1,3</td>
<td>$M$</td>
<td>0</td>
<td>$M + 8$</td>
<td>$M + 11$</td>
<td>$M + 15$</td>
<td>$M + 12$</td>
</tr>
<tr>
<td>2,1</td>
<td>$M + 15$</td>
<td>$M + 14$</td>
<td>$M + 11$</td>
<td>0</td>
<td>$M$</td>
<td>0</td>
</tr>
<tr>
<td>2,2</td>
<td>$M + 12$</td>
<td>$M + 11$</td>
<td>$M + 8$</td>
<td>$M$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2,3</td>
<td>$M + 16$</td>
<td>$M + 15$</td>
<td>$M + 12$</td>
<td>0</td>
<td>$M$</td>
<td>0</td>
</tr>
</tbody>
</table>

$M = 33$
1.4 Generalized Routing Problems

GATSP and ATSP: Example

GATSP instance
optimal tour length: 24

equivalent ATSP instance
optimal tour length: $2M + 24$

<table>
<thead>
<tr>
<th>$c'$</th>
<th>1,1</th>
<th>1,2</th>
<th>1,3</th>
<th>2,1</th>
<th>2,2</th>
<th>2,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td></td>
<td>0</td>
<td>$M$</td>
<td>$M + 11$</td>
<td>$M + 14$</td>
<td>$M + 15$</td>
</tr>
<tr>
<td>1,2</td>
<td>$M$</td>
<td></td>
<td>0</td>
<td>$M + 12$</td>
<td>$M + 15$</td>
<td>$M + 16$</td>
</tr>
<tr>
<td>1,3</td>
<td>0</td>
<td>$M$</td>
<td></td>
<td>$M + 8$</td>
<td>$M + 11$</td>
<td>$M + 12$</td>
</tr>
<tr>
<td>2,1</td>
<td>$M + 15$</td>
<td>$M + 14$</td>
<td>$M + 11$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,2</td>
<td>$M + 12$</td>
<td>$M + 11$</td>
<td>$M + 8$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,3</td>
<td>$M + 16$</td>
<td>$M + 15$</td>
<td>$M + 12$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$M = 33$
Proof (cont.)

Let $C'$ be a Hamiltonian circuit in $I' = (K_n^*, c')$ that visits all vertices of a cluster $V_i$ in ascending order before moving to another cluster. This way, $C$ contains precisely $k$ edges of weight $\geq M$, and we have $c(C') < kM + M$.

We claim that the optimal solution to $I'$ is such a circuit. Otherwise, $\text{OPT}(I') \geq (k + 1)M = kM + M > c(C')$.

In particular, if the optimal solution enters $V_i$ at $v_{i,j}$, it leaves $V_i$ at $v_{i,(j-1) \mod r_i}$. The cost of the edge to $v_{i,j}$ is $(M +)$ the shortest path length to $v_{i,j}$, and the cost of the edge from $v_{i,(j-1) \mod r_i}$ is $(M +)$ the shortest path length from $v_{i,j}$ (note the shift!).

Hence we find a GATSP tour by tracing the shortest paths back. For the cost we find $\text{OPT}(I) \leq \text{OPT}(I') - kM$. 
1.4 Generalized Routing Problems

GATSP and ATSP

Proof (cont.)

Consider an optimal GATSP tour $C$. Create a Hamiltonian circuit $C'$ in $K_n^*$ by sorting the clusters by their order of appearance in $C$ and traversing the whole cluster before proceeding. The cost of $C'$ increases at most by $kM$. Thus

$$\text{OPT}(I') \leq c(C') \leq c(C) + kM = \text{OPT}(I) + kM.$$ 

Corollary

$GDRPP \leq GATSP \leq ATSP \leq TSP.$

Corollary

A GDRPP with clusters $E_1, \ldots, E_k$ can be polynomially transformed into a TSP on $2 \sum_{i=1}^{k} |E_i|$ vertices (with a large increase in cost).

Lemma (Exercise)

Metric TSP $\leq$ Metric ATSP $\leq$ GATSP.
§1.4 Generalized Routing Problems

**GDRPP to TSP: Example**

**GDRPP instance**
- Optimal tour: 30

**GATSP instance**
- Optimal tour: 30

**ATSP instance** ($M = 39$)
- Optimal tour: $108 = 2 \cdot 39 + 30$

**TSP instance** ($M = 225$)
- Optimal tour: $1008 = 4 \cdot 225 + 108$
Chapter 1

S-Bahn Challenge

§1.5 Public Transportation Networks
1.5 Public Transportation Networks

Line Networks

**Definition**

A **line network** is a graph $G$ together with a **line cover** $\mathcal{L}$, i.e., $\mathcal{L}$ is a set of walks in $G$ such that $E(G) = \bigcup_{L \in \mathcal{L}} E(L)$. 
Line Networks and Event-Activity Networks

Remarks

▶ Depending on the application, line networks may be undirected or directed.
▶ The vertices of a line network are stations or stops.
▶ The elements of $L$ are lines or routes.
▶ The two directions of a classical path-shaped line can be modeled by two separated walks or by a closed walk.

Definition

An event-activity network (EAN) is a directed graph $E$ whose vertices are called events and whose edges are called activities.
1.5 Public Transportation Networks

Timetables for Line Networks

Definition

Let $\mathcal{N} = (G, \mathcal{L})$ be a line network.

- A trip of a line $L = (e_1, \ldots, e_k) \in \mathcal{L}$ is a pair $(\tau_{\text{dep}}, \tau_{\text{arr}})$ of maps $\tau_{\text{dep}}, \tau_{\text{arr}} : \{1, \ldots, k\} \to \mathbb{R}$ such that
  
  \[ \tau_{\text{dep}}(i) \leq \tau_{\text{arr}}(i), \quad i = 1, \ldots, k \]
  \[ \tau_{\text{arr}}(i) \leq \tau_{\text{dep}}(i + 1), \quad i = 1, \ldots, k - 1. \]

- A schedule for $L$ is a collection of trips of $L$.

- A timetable for $\mathcal{N}$ assigns a schedule to each line.
1.5 Public Transportation Networks

Time Expansion

Definition

Consider a timetable $T$ for a line network $\mathcal{N}$. The time expansion of $\mathcal{N}$ w.r.t. $T$ is the event-activity network $\mathcal{E}$, together with the length function $\ell : E(\mathcal{E}) \to \mathbb{R}_{\geq 0}$, constructed as follows:

1. For each trip $\tau = (\tau_{\text{dep}}, \tau_{\text{arr}})$ of a line $L = (e_1, \ldots, e_k)$ in $\mathcal{N}$:
   - Add departure events $(L, \tau, i, \text{dep})$ for $i = 1, \ldots, k$.
   - Add arrival events $(L, \tau, i, \text{arr})$ for $i = 1, \ldots, k$.
   - Add driving activities $(L, \tau, i, \text{dep}) \rightarrow (L, \tau, i, \text{arr})$ with length $\tau_{\text{arr}}(i) - \tau_{\text{dep}}(i)$, $i = 1, \ldots, k$.
   - Add waiting activities $(L, \tau, i, \text{arr}) \rightarrow (L, \tau, i + 1, \text{dep})$ with length $\tau_{\text{dep}}(i + 1) - \tau_{\text{arr}}(i)$, $i = 1, \ldots, k - 1$.

2. Add a transfer activity $(L, \tau, i, \text{arr}) \rightarrow (L', \tau', i', \text{dep})$ with length $\tau'_{\text{dep}}(i') - \tau_{\text{arr}}(i)$ for each pair of trips $(\tau, \tau')$ associated to a pair of lines $(L, L')$ whenever:
   - $\tau'_{\text{dep}}(i') - \tau_{\text{arr}}(i) \geq 0$, and
   - the $(i + 1)$-st vertex of $L$ and the $i'$-th vertex of $L'$ coincide in $\mathcal{N}$,
   - $(L, \tau, i, \text{arr})$ and $(L', \tau', i', \text{dep})$ are not connected by a waiting activity.
1.5 Public Transportation Networks

Time Expansion

Remarks

- Trips correspond to certain disjoint directed paths in the EAN.
- The EAN is bipartite, as there are no departure-departure and no arrival-arrival activities.
- No activity goes “backward in time”: Circuits can only have length 0.
- The number of driving and waiting activities is linear in the number of trips, whereas the number of transfer activities is quadratic.
- A transfer activity between two trips of a line at one of its endpoints is called a *turnaround activity*.
- Often there is no point in a transfer between trips of parallel lines, and the corresponding transfer activities can be removed.
- Sometimes we want to establish a minimum transfer time, and hence only add transfer activities where $\tau_{\text{dep}}^{'}(i^{'}) - \tau_{\text{arr}}(i)$ is large enough.
- Footpath information can also be included using transfer activities.
Time Expansion: Example

Trip 1: 10:12 → 10:44
Trip 2: 11:12 → 11:48
Trip 1: 10:46 → 10:58
Trip 2: 11:49 → 12:02
Trip 1: 11:00 → 11:37
1.5 Public Transportation Networks

**Time Expansion: Example**

- **Trip 1:** 10:12 → 10:44
  - Line 1
- **Trip 2:** 11:12 → 11:48
  - Line 2
- **Trip 1:** 10:46 → 10:58
  - Line 1
- **Trip 2:** 11:49 → 12:02
  - Line 2
- **Trip 1:** 11:00 → 11:37
  - Line 2
1.5 Public Transportation Networks

Time Expansion: Example

Trip 1: 10:12 → 10:44
Trip 2: 11:12 → 11:48

Trip 1: 10:46 → 10:58
Trip 2: 11:49 → 12:02

Trip 1: 11:00 → 11:37
§1.5 Public Transportation Networks

Time Expansion: Example

Trip 1: 10:12 → 10:44
Trip 2: 11:12 → 11:48

Trip 1: 10:46 → 10:58
Trip 2: 11:49 → 12:02

Trip 1: 11:00 → 11:37

1,1,1,dep 10:12
1,2,1,dep 11:12

1,1,1,arr 10:44
1,2,1,arr 11:48

1,1,2,dep 10:46
1,2,2,dep 11:49

1,1,2,arr 10:58
1,2,2,arr 12:02

2,1,1,dep 11:00
2,1,2,dep 11:00

2,1,1,arr 11:37
2,1,2,arr 11:37
Time Expansion: Example

Trip 1: 10:12 → 10:44
Trip 2: 11:12 → 11:48

Trip 1: 10:46 → 10:58
Trip 2: 11:49 → 12:02

Trip 1: 11:00 → 11:37