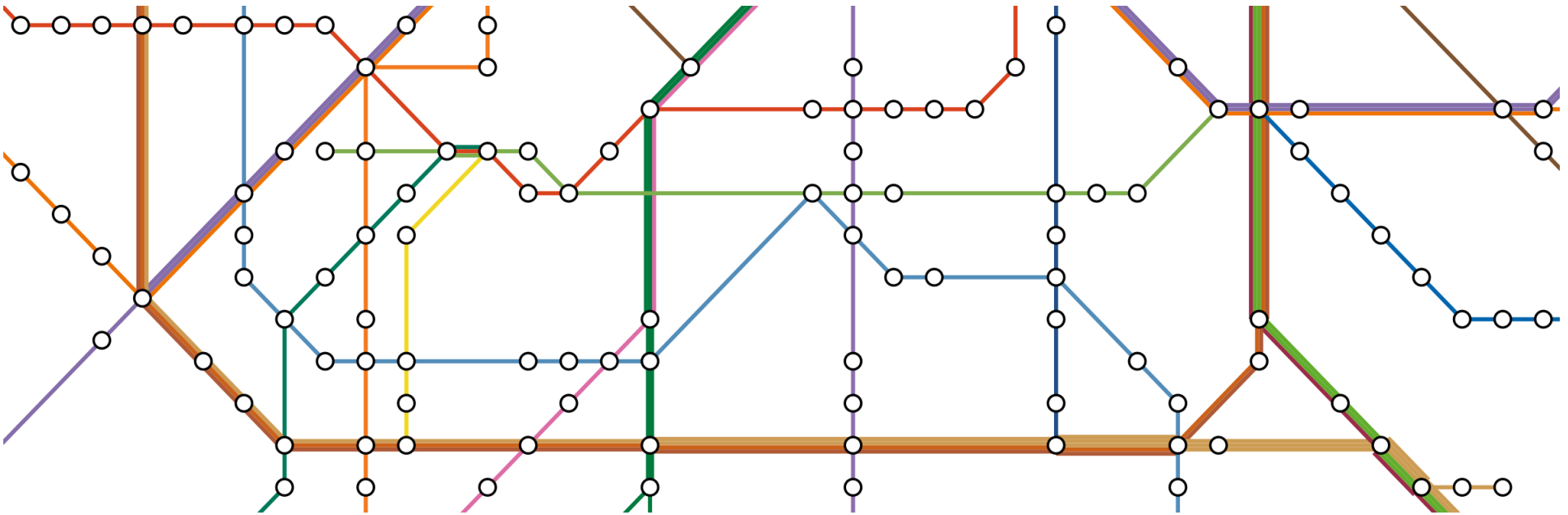
Mathematical Aspects of Public Transportation Networks

Niels Lindner



April 16, 2018

Organization

- Lecture: Mon 14-16, ZIB 2006
- Tutorials: Thu 12-14, ZIB 2006 (new time!)

- Problem sets: published online, due on Mondays
- Exam: written, need 50% of problem sets

- Office: ZIB 3007
- E-Mail: `lindner@zib.de`
- Course website: `www.zib.de/node/3447`

Topics

Course outline

1. S-Bahn Challenge
2. Shortest Routes
3. Periodic Timetabling
4. More on Traffic Optimization
5. Metro Map Drawing

Seminar on Shortest Paths

First meeting: Tomorrow, 10 am, ZIB lecture hall

Lecture Optimization III

Mon 10-12, Wed 10-12, ZIB lecture hall

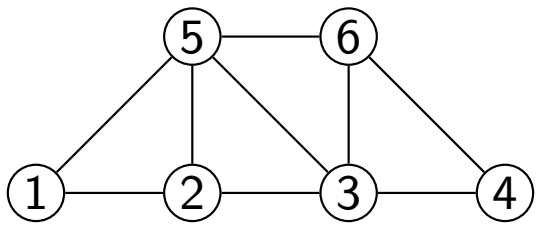
Chapter 1

S-Bahn Challenge

§1.1 Walks in graphs

Walks in undirected graphs

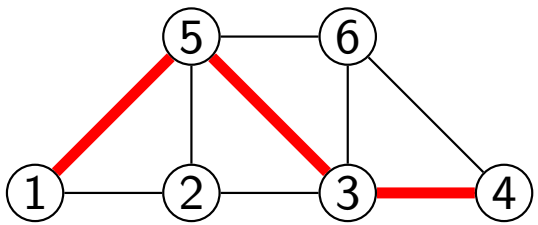
Let G be an undirected graph with vertex set V and edge set E.



Walks in undirected graphs

Let G be an undirected graph with vertex set V and edge set E.

- A **walk** in G is a sequence of adjacent edges (vertices) in G.

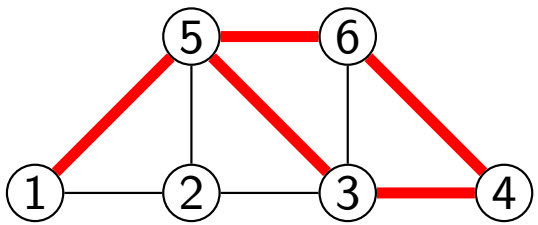


walk: 1, 5, 3, 4

Walks in undirected graphs

Let G be an undirected graph with vertex set V and edge set E.

- A **walk** in G is a sequence of adjacent edges (vertices) in G.

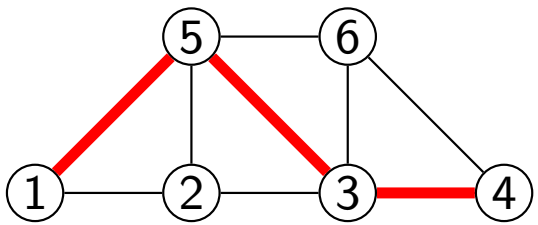


walk: 1, 5, 3, 4, 6, 5

Walks in undirected graphs

Let G be an undirected graph with vertex set V and edge set E.

- A **walk** in G is a sequence of adjacent edges (vertices) in G.
- A walk is called a **path** if all visited vertices (hence edges) are distinct.

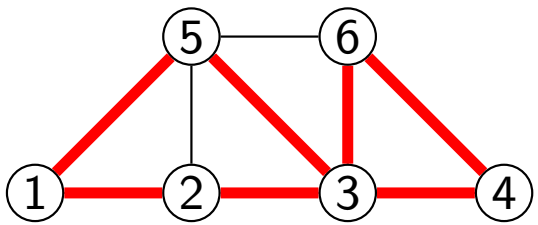


path: 1, 5, 3, 4

Walks in undirected graphs

Let G be an undirected graph with vertex set V and edge set E.

- A **walk** in G is a sequence of adjacent edges (vertices) in G.
- A walk is called a **path** if all visited vertices (hence edges) are distinct.
- A walk is **closed** if first and last vertex coincide.

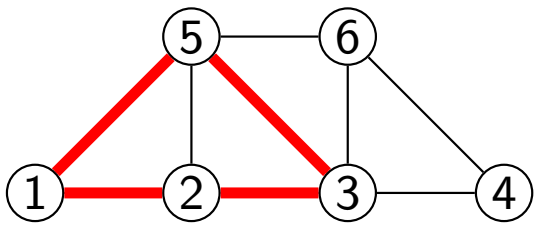


closed walk: 1, 2, 3, 4, 6, 3, 5, 1

Walks in undirected graphs

Let G be an undirected graph with vertex set V and edge set E.

- A **walk** in G is a sequence of adjacent edges (vertices) in G.
- A walk is called a **path** if all visited vertices (hence edges) are distinct.
- A walk is **closed** if first and last vertex coincide.
- A closed walk where all intermediate vertices are distinct is a **circuit**.

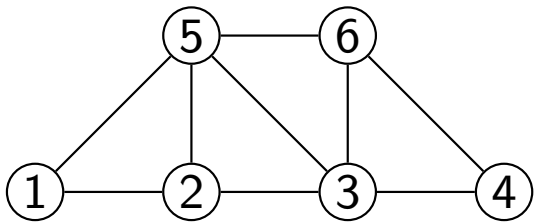


circuit: 1, 2, 3, 5, 1

Walks in undirected graphs

Let G be an undirected graph with vertex set V and edge set E.

- A **walk** in G is a sequence of adjacent edges (vertices) in G.
- A walk is called a **path** if all visited vertices (hence edges) are distinct.
- A walk is **closed** if first and last vertex coincide.
- A closed walk where all intermediate vertices are distinct is a **circuit**.
- G is **connected** if there is a path between any pair of distinct vertices.

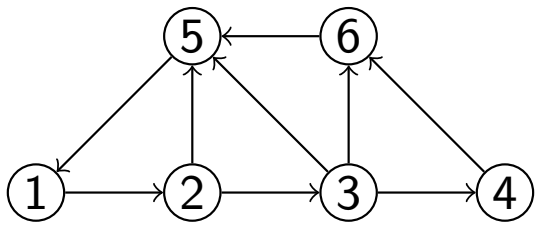


is connected

Walks in directed graphs

For directed graphs:

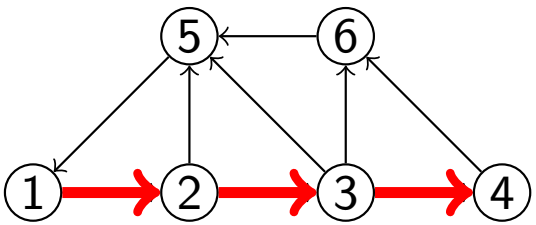
- only *forward* edges: **directed** walk/path/circuit, strongly connected
- *forward* and *backward* edges: **oriented** walk/path/circuit, weakly connected



Walks in directed graphs

For directed graphs:

- only *forward* edges: **directed** walk/path/circuit, strongly connected
- *forward* and *backward* edges: **oriented** walk/path/circuit, weakly connected

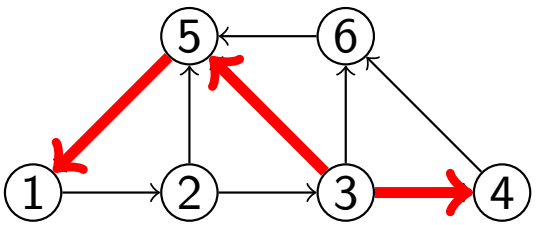


directed path: 1, 2, 3, 4

Walks in directed graphs

For directed graphs:

- only *forward* edges: **directed** walk/path/circuit, strongly connected
- *forward* and *backward* edges: **oriented** walk/path/circuit, weakly connected



oriented path: 1, 5, 3, 4
(1,5) and (5,3) are backward edges

Königsberg bridges

Images: Bogdan Giuşcă, Xiong, Mark Foskey, all CC-BY-SA 3.0

Question (Euler, 1736)

Is there a closed walk visiting all seven bridges exactly once?

Observation

Every part of the city (aka vertex) will be entered and left, and is therefore accessed by an even number of bridges.

$\rightsquigarrow$ There is no such walk.

Euler tours

Let $G = (V, E)$ be a connected undirected graph.

- An **Euler tour (Euler walk)** is a closed walk (walk) that visits each edge in E exactly once.
- G is called **Eulerian** if it admits an Euler tour.

Theorem (Euler, 1736; Hierholzer, 1873)

- *G has an Euler tour*
  $\Leftrightarrow$ *every vertex of G has even degree*
  $\Leftrightarrow$ *G is a union of edge-disjoint circuits.*
- *G has an Euler walk*
  $\Leftrightarrow$ *exactly zero or two vertices of G have odd degree.*

Proof.

Euler tour/walk $\Rightarrow$ degree condition: Every vertex needs to be entered and left, except for the two endpoints of an Euler walk.
Rest: Hierholzer's algorithm. □

Hierholzer's algorithm for an Euler tour

Given a graph G whose vertices have all even degree, the following computes a decomposition into circuits and an Euler tour:

Basic Algorithm (Hierholzer, 1873)

1. Set $\mathcal{C} := \emptyset$.
2. While G contains a non-isolated vertex:
   - Pick a non-isolated vertex of G.
   - Traverse G until a previously visited vertex is reached again.
   - Add the corresponding circuit C to $\mathcal{C}$.
   - Remove the edges of C from G.
3. Circuit decomposition $\leftarrow \mathcal{C}$.
4. While $|\mathcal{C}| \geq 2$:
   - Pop two circuits C_1, C_2 sharing a common vertex from $\mathcal{C}$.
   - Insert C_2 into C_1 at the common vertex.
   - Push the new cycle back to $\mathcal{C}$.
5. Euler tour $\leftarrow$ unique element of $\mathcal{C}$.

Remarks on Hierholzer's algorithm

- Removing circuits from a graph changes each vertex degree by either 0 or 2, so Step 2 will terminate.
- In Step 4, the union of all cycles in $\mathcal{C}$ is always E. Since G is connected, one can always find two cycles with a common vertex.
- Constructing an Euler walk can be reduced to computing an Euler tour by connecting the two odd-degree vertices with an edge.
- Hierholzer's algorithm can be improved to give a linear-time algorithm.

Finding Euler tours in linear time

Let G be a connected undirected graph where all vertices have even degree.

Improved Hierholzer's Algorithm (Read, Fleischner)

Data structures: Store G in an adjacency list. Let *head* and *tail* be stacks.

1. Let $v \in V$. Set $head := \{v\}$ and $tail := \emptyset$.
2. While $head \neq \emptyset$:
   - While top vertex v of *head* is not isolated:
     - Pick an edge $\{v, w\}$ and remove it from G.
     - Push w to *head*.
   - While $head \neq \emptyset$ and top vertex v of *head* is isolated:
     - Pop v from *head*.
     - Push v to *tail*.
3. Euler tour $\leftarrow$ *tail*.

This algorithm runs in $\mathcal{O}(|E|)$ time. It traverses the graph and finds closed walks. The insertion point of the next cycle is on top of the stack *head*.

Results for directed graphs

Let $G = (V, E)$ be a weakly connected directed graph.

Theorem (Directed Euler tours)

G contains a directed Euler tour
$\Leftrightarrow$ *at every vertex, the number of ingoing edges equals the number of outgoing edges,*
$\Leftrightarrow$ *G can be decomposed into edge-disjoint directed circuits.*

Theorem (Directed Euler walks)

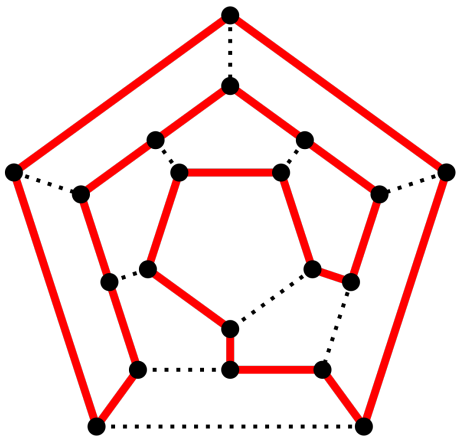
G contains a directed Euler walk from s to t if and only if

$$\begin{aligned} \text{indeg}(s) &= \text{outdeg}(s) - 1, \\ \text{indeg}(t) &= \text{outdeg}(t) + 1, \\ \text{indeg}(v) &= \text{outdeg}(v) \qquad \text{for all } v \in V \setminus \{s, t\}. \end{aligned}$$

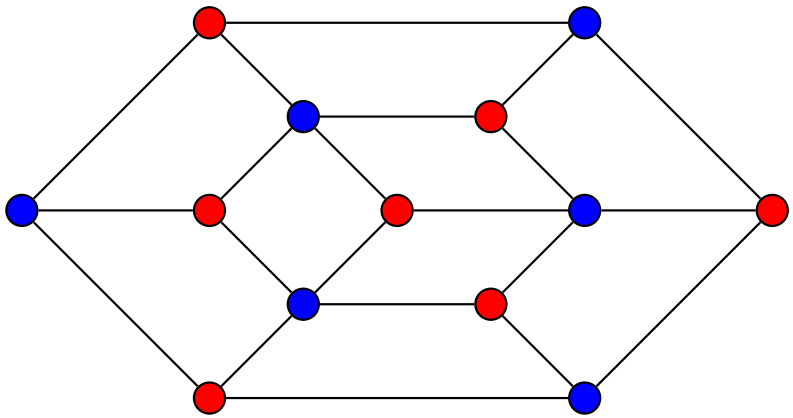
Hamiltonian circuits

Let G be a connected undirected graph.

- A **Hamiltonian circuit (Hamiltonian path)** is a circuit (path) visiting each vertex of G exactly once.
- If G admits a Hamiltonian circuit, then G is called **Hamiltonian**.



Icosian game
(Hamilton, 1857)



Cell of a bee
(Kirkman, 1856)

Hamiltonian circuits: Hardness

Theorem (Karp, 1972)

- *Given a connected undirected graph G, it is* NP*-complete to decide whether G contains a Hamiltonian circuit (or path).*
- *Given a weakly connected digraph G, it is* NP*-complete to decide whether G contains a directed Hamiltonian circuit (or path).*

Proof.

Usually by polynomial-time reduction of 3-SAT or Vertex Cover. □

Conclusion

Unless P = NP, there is no polynomial-time algorithm for the Hamiltonian circuit problem. (Euler tour works in linear time!)

Hamiltonian circuits: Hardness

Why is Hamilton more difficult than Euler?

- An Euler tour visits every vertex, and it almost always visits vertices more than once, so it is unlikely to be a circuit.
- A (closed) walk visiting all vertices exactly once is always a path (circuit). In particular, the Hamilton circuit problem looks for more delicate structures.
- Seen as a set of edges, a graph has at most one Euler tour, but there may be many Hamilton circuits.
- If a particular edge is part of an Euler tour, then all edges are. This is false for Hamilton circuits. As a consequence, a graph-traversing algorithm for finding Hamilton circuits has to make more decisions.

Euler and Hamilton are not enough

Public transport networks usually have lots of tree structures, e.g., east of Ostkreuz station in the Berlin S-Bahn network. This prevents the underlying undirected graph from being Eulerian or Hamiltonian.

Optimization versions

Consider a graph with a cost function on the edges.

Informal definitions:

- The **Chinese Postman Problem** is to find the cheapest way to make a graph Eulerian.
- The **Traveling Salesman Problem** is to find the cheapest way to make a graph Hamiltonian and then to find the cheapest Hamiltonian circuit.

Chapter 1

S-Bahn Challenge

§1.2 The Chinese Postman Problem

Definition

Formulation (Guan, 1960)

A postman has to deliver letters to a given neighborhood. He needs to walk through all the streets in the neighborhood and back to the post-office. How can he design his route so that he walks the shortest distance?

Definition

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

- A **Chinese Postman tour** is a closed walk $(e_1, \ldots, e_k)$ in G visiting each edge *at least* once, i.e., $\{e_1, \ldots, e_k\} = E$.
- The **Chinese Postman Problem (CPP)** is to find an optimal tour, i.e., a Chinese Postman tour $C := (e_1, \ldots, e_k)$ that minimizes the total cost $c(C) := \sum_{i=1}^{k} c(e_i)$.

Remarks

Remarks

- The problem is sometimes also called *Route Inspection Problem*.
- If G is Eulerian, an Euler tour is an optimal solution to the CPP.
- Any connected undirected graph has a Chinese Postman tour: If we duplicate all edges, then all nodes have even degree, and we can take the corresponding Euler tour. However, this will lead to a Chinese Postman tour where each original edge is visited at least twice.

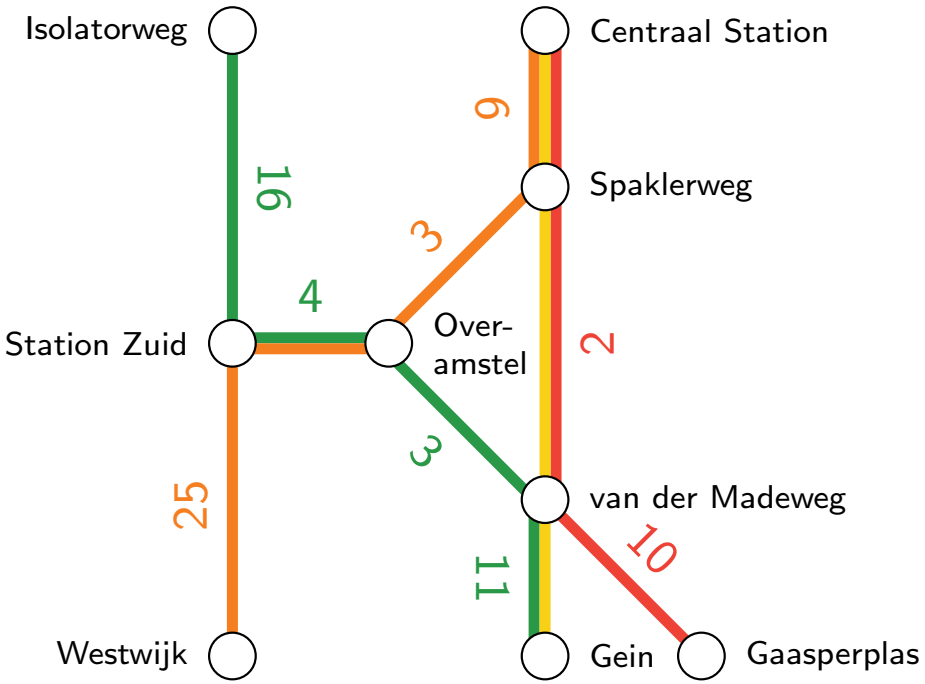
Strategy

The strategy to solve the CPP is as follows:

- Duplicate edges in a clever way, so that all vertex degrees become even.
- Take the corresponding Euler tour.

Example

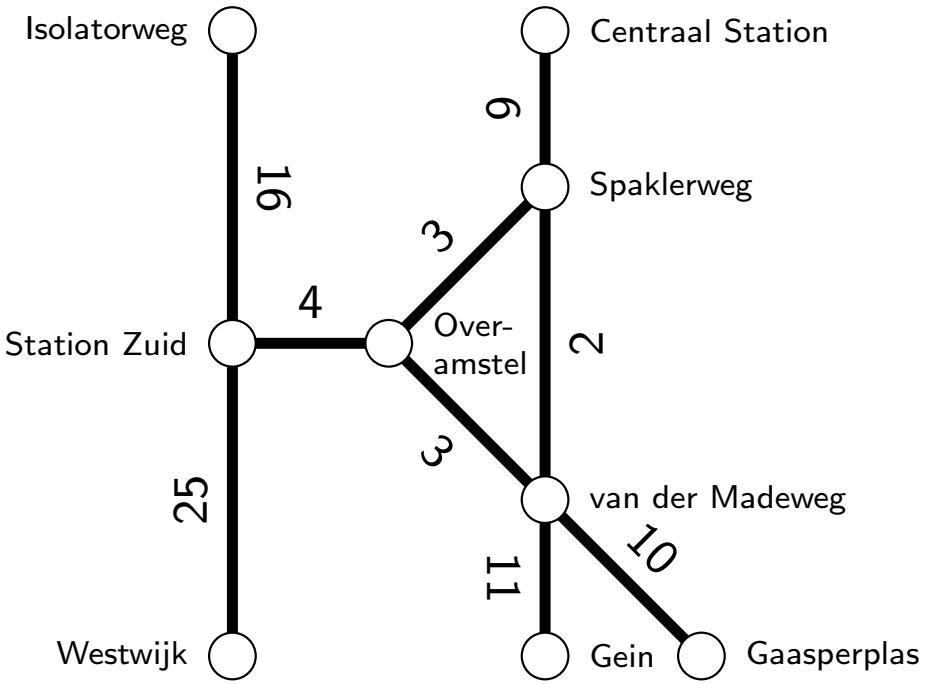
Example (Amsterdam metro)



- 9 vertices (odd: 8)
- 9 edges
- not Eulerian
- not Hamiltonian
- edge cost: average travel time in minutes
- sum of edge cost: 83

Example

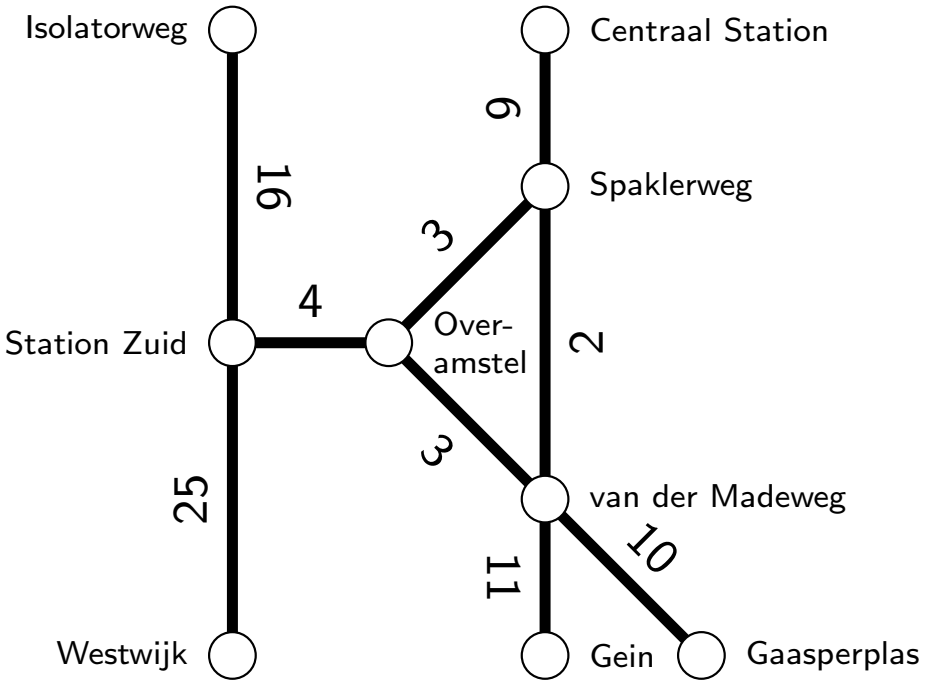
Example (Amsterdam metro)



- 9 vertices (odd: 8)
- 9 edges
- not Eulerian
- not Hamiltonian
- edge cost: average travel time in minutes
- sum of edge cost: 83

Example

Example (Amsterdam metro)



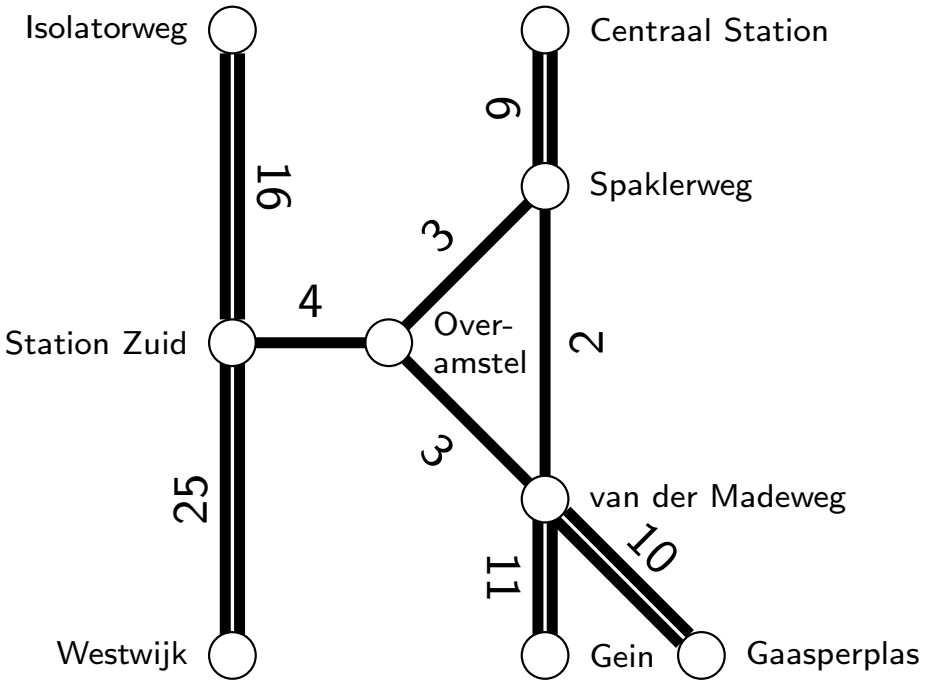
- 9 vertices (odd: 8)
- 9 edges
- not Eulerian
- not Hamiltonian
- edge cost: average travel time in minutes
- sum of edge cost: 83

Observation

- A chinese Postman tour enters and leaves all vertices of degree 1.
  $\rightsquigarrow$ Need to travel twice edges leading to degree 1 vertices.

Example

Example (Amsterdam metro)



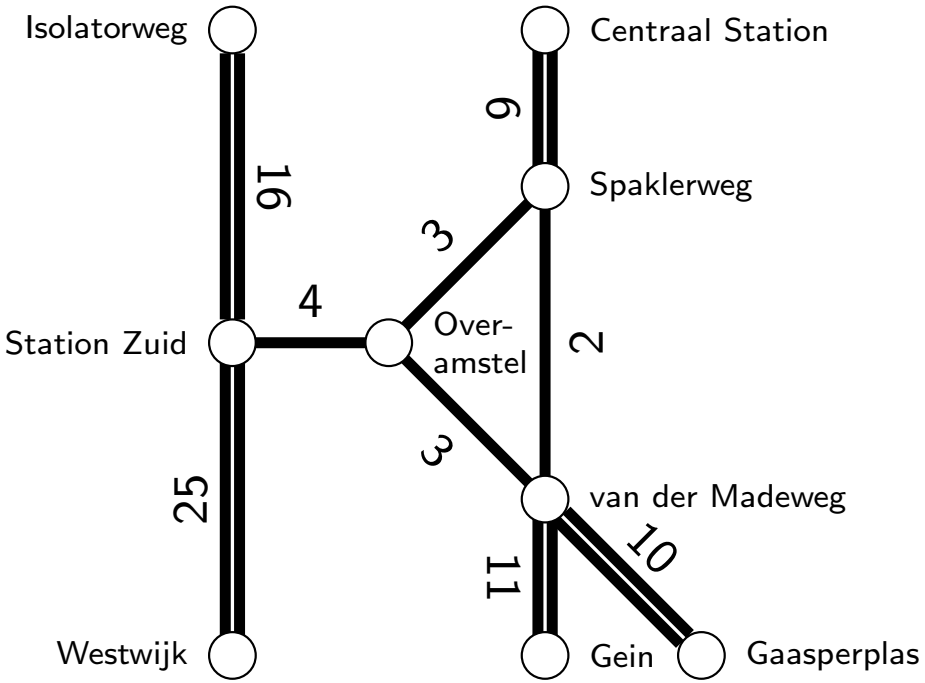
- 9 vertices (odd: 2)
- 9 edges (+5)
- not Eulerian
- not Hamiltonian
- edge cost: average travel time in minutes
- sum of edge cost: 83 (+71)

Observation

- A chinese Postman tour enters and leaves all vertices of degree 1.
  $\rightsquigarrow$ Need to travel twice edges leading to degree 1 vertices.

Example

Example (Amsterdam metro)



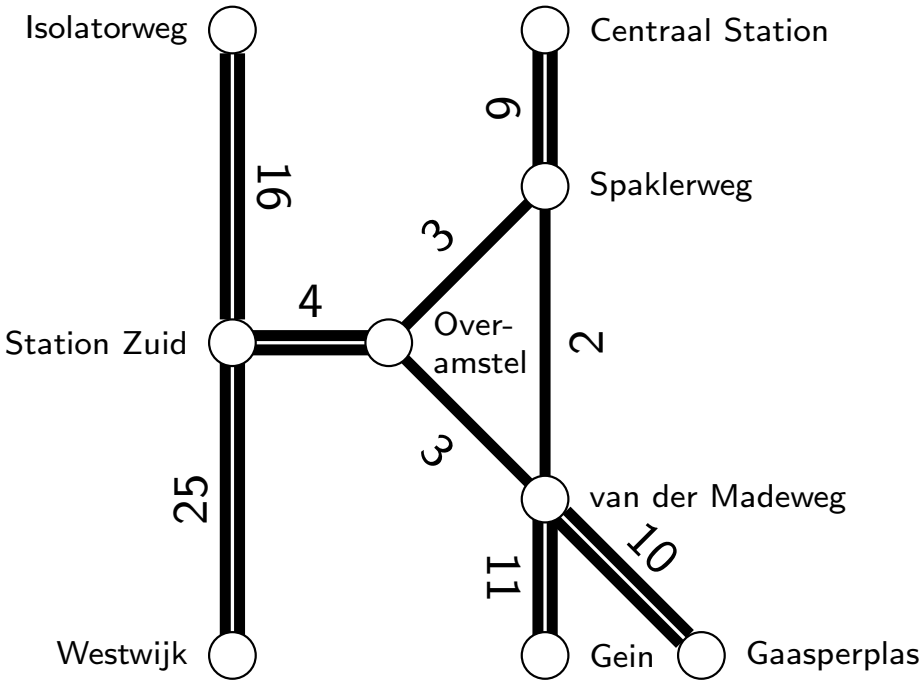
- 9 vertices (odd: 2)
- 9 edges (+5)
- not Eulerian
- not Hamiltonian
- edge cost: average travel time in minutes
- sum of edge cost: 83 (+71)

Observation

- A chinese Postman tour enters and leaves all vertices of degree 1.
  $\rightsquigarrow$ Need to travel twice edges leading to degree 1 vertices.
- The edge *Station Zuid–Overamstel* needs to be duplicated as well.

Example

Example (Amsterdam metro)



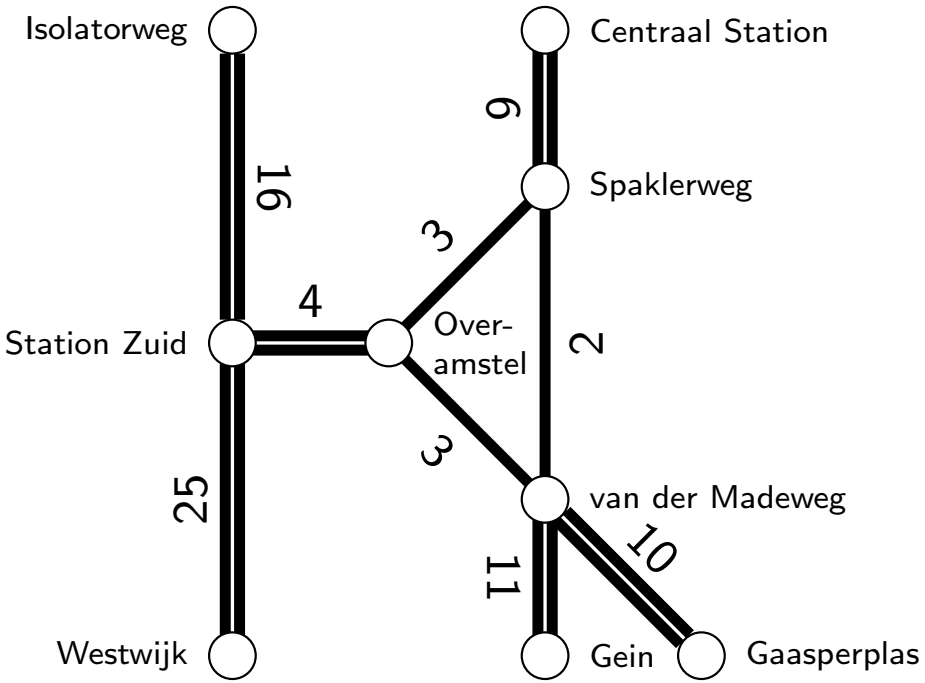
- 9 vertices (odd: 0)
- 9 edges (+6)
- ~~not~~ Eulerian
- not Hamiltonian
- edge cost: average travel time in minutes
- sum of edge cost: 83 (+75)

Observation

- A chinese Postman tour enters and leaves all vertices of degree 1.
  $\rightsquigarrow$ Need to travel twice edges leading to degree 1 vertices.
- The edge *Station Zuid–Overamstel* needs to be duplicated as well.

Example

Example (Amsterdam metro)



- 9 vertices (odd: 0)
- 9 edges (+6)
- ~~not~~ Eulerian
- not Hamiltonian
- edge cost: average travel time in minutes
- sum of edge cost: 83 (+75)

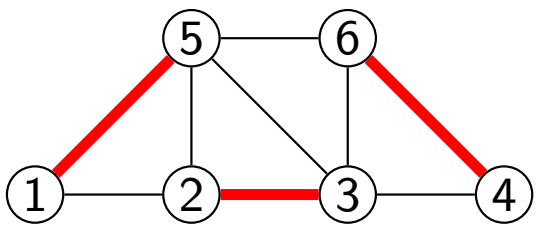
Conclusion

- The graph has become Eulerian. The optimal tour has cost 158.

Interlude: Matchings

Let $G = (V, E)$ be an undirected graph.

- A **matching** M in G is a subset of E such that each vertex $v \in V$ is contained in at most one edge $e \in M$.
- A matching M is **perfect** if each vertex is contained in exactly one edge $e \in M$.
- Attention: G does not necessarily have a perfect matching.
  (Criteria: Hall's marriage theorem, Tutte-Berge formula)
- E.g., for trivial reasons, graphs with an odd number of vertices cannot have a perfect matching.



a perfect matching

Interlude: Finding Min-Weight Perfect Matchings

Let $G = (V, E)$ be an undirected graph with $|V| = n$, $|E| = m$, equipped with a weight function $w : E \to \mathbb{R}$.

Theorem (Edmonds, 1965)

A minimum weight perfect matching can be computed in strongly polynomial time.

Proof.

Edmonds' blossom algorithm runs in $\mathcal{O}(n^2 m)$ time. □

Remarks

- The weight of a matching $M \subseteq E$ is defined as $\sum_{e \in M} w(e)$.
- A similar algorithm computes a maximum weight (not necessarily perfect) matching.
- There are various asymptotically faster techniques, e.g., Gabow's algorithm with a running time of $\mathcal{O}(nm + n^2 \log n)$.

Algorithm

Let $G = (V, E)$ be a connected undirected graph.

Algorithm (Edmonds/Johnson, 1973)

1. Let $T = \{v_1, \ldots, v_k\}$ be the set of odd vertices in G.
   If $T = \emptyset$, go to Step 5.
2. Construct a complete graph K_k on $\{1, \ldots, k\}$ with weight function

$$w(\{i, j\}) := \text{length of shortest path from } v_i \text{ to } v_j \text{ in } G,$$

   where $1 \leq i < j \leq k$.
3. Find a minimum weight perfect matching M in K_k w.r.t. w.
4. For each edge $\{i, j\} \in M$, duplicate the edges in G along the shortest path from v_i to v_j.
5. Compute an Euler tour and return it.

Algorithm: Correctness

Theorem

The Edmonds-Johnson algorithm solves CPP.

Proof.

- Handshaking Lemma (Euler, 1736): The number k of odd vertices in a graph is even. In particular, K_k has a perfect matching.
- Duplicating the edges along a path from v_i to v_j will cause the degree of v_i and v_j to go up by 1, whereas the degree of the intermediate vertices increases by 2. Since we matched all odd degree vertices in the graph, this forces all vertex degrees to become even.

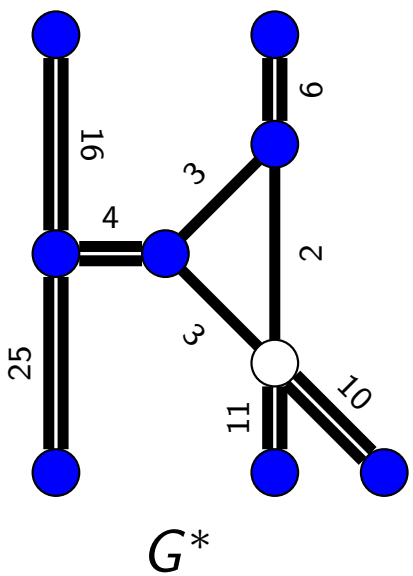
$\rightsquigarrow$ The algorithm terminates and returns a Chinese Postman tour C.
The cost of C is

$$c(C) = \sum_{e \in E} |\{\text{traversals of } e\}| \cdot c(e) = \sum_{e \in E} c(e) + \sum_{\{i,j\} \in M} w(\{i,j\}).$$

Algorithm: Optimality

Proof (cont.)

Let C^* be the optimal Chinese Postman tour. Let G^* be the Eulerian graph where all edges are multiplied according to their multiplicity in C^*.



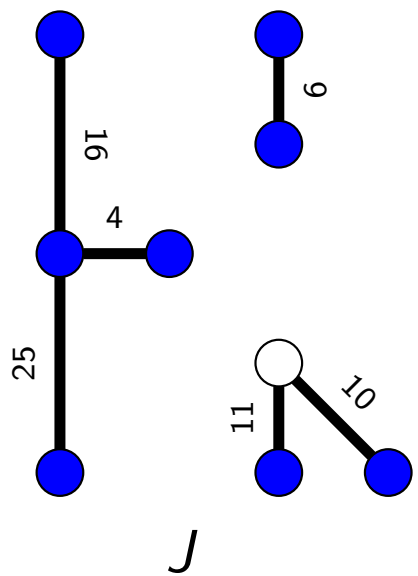
G^*

- No edge $e \in E$ is used by C^* more than twice: Otherwise, G^* remains Eulerian if two copies of e are removed, producing a shorter Euler tour.

Algorithm: Optimality

Proof (cont.)

Let C^* be the optimal Chinese Postman tour. Let G^* be the Eulerian graph where all edges are multiplied according to their multiplicity in C^*.

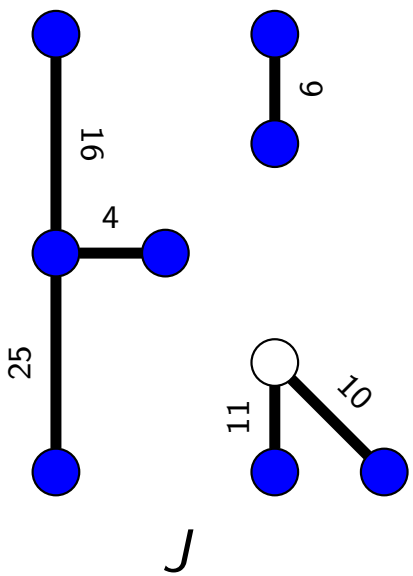


J

- No edge $e \in E$ is used by C^* more than twice: Otherwise, G^* remains Eulerian if two copies of e are removed, producing a shorter Euler tour.
- Let E^* denote the set of edges C^* uses twice. Set $J := (V, E^*)$.
- The odd nodes in J are precisely the odd nodes in G.
- Handshaking Lemma again: Any connected component of J contains an even number of odd vertices of G.

Algorithm: Optimality

Proof (cont.)

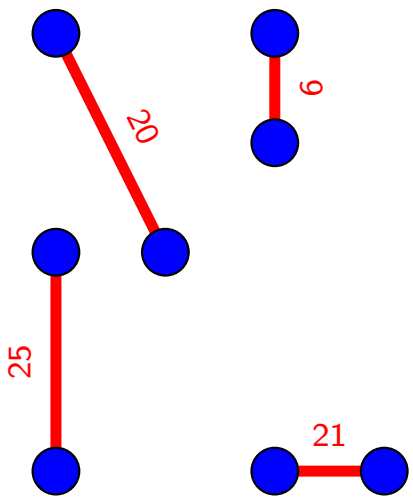


J

- By minimality of C^*, we can assume that J does not contain circuits and is hence a forest.
- So any two odd vertices in the same component are connected by a unique path, which must - again by minimality - be a shortest path.
- Construct a matching in M^* in K_k as follows: Start at an odd vertex v_i and traverse J until the next odd vertex v_j, remove the edges along the path, and continue.

Algorithm: Optimality

Proof (cont.)



M^*
(weight: 75)

- By minimality of C^*, we can assume that J does not contain circuits and is hence a forest.
- So any two odd vertices in the same component are connected by a unique path, which must - again by minimality - be a shortest path.
- Construct a matching in M^* in K_k as follows: Start at an odd vertex v_i and traverse J until the next odd vertex v_j, remove the edges along the path, and continue.
- Each step adds weight $w(\{i,j\})$ to M^*. The total weight of M^* equals $\sum_{e \in E^*} c(e)$.

Algorithm

Proof (cont.)

- The cost of C^* is now

$$\begin{aligned} c(C^*) := \sum_{e \in E} c(e) + \sum_{e' \in E^*} c(e') &= \sum_{e \in E} c(e) + \sum_{\{i,j\} \in M^*} w(\{i,j\}) \\ &\geq \sum_{e \in E} c(e) + \sum_{\{i,j\} \in M} w(\{i,j\}) \\ &= c(C), \end{aligned}$$

  because M is a minimum weight perfect matching.
- Thus $c(C) = c(C^*)$ and hence C is optimal.

□

Algorithm

Corollary

The CPP on a graph G with n vertices is solvable in $\mathcal{O}(n^3)$ time.

Proof.

Denote by k the number of odd-degree vertices in G, clearly $k \leq n$.

- All-pairs shortest paths can be done in $\mathcal{O}(k^3)$ time.
- Minimum weight perfect matching in K_k can be done in $\mathcal{O}(k \cdot k^2 + k^2 \log k)$, hence $\mathcal{O}(k^3)$ time.
- Computing an Euler tour takes $\mathcal{O}(m)$ time.

□

Integer Program

Let $G = (V, E)$ be a connected undirected graph.
The **incidence matrix** of G is the $|V| \times |E|$-matrix with entries

$$a_{ve} := \begin{cases} 1 & \text{if } v \in e, \\ 0 & \text{otherwise,} \end{cases} \qquad v \in V, e \in E.$$

Theorem

The CPP on G w.r.t. a cost function $c : E \to \mathbb{R}_{\geq 0}$ is equivalent to the following integer program:

$$\begin{aligned} \textit{Minimize} \quad & \sum_{e \in E} c(e) x_e \\ \textit{s. t.} \quad & \sum_{e \in E} a_{ve} x_e - 2y_v = \sum_{e \in E} a_{ve}, && v \in V, \\ & x_e \in \{0, 1\}, && e \in E, \\ & y_v \in \mathbb{Z}, && v \in V. \end{aligned}$$

Integer Program

Proof.

Tutorial. □

Remarks

- CPP can hence be solved by a general-purpose integer programming software. However, since the LP relaxation leads in general to a non-integral polytope (in fact, the system is totally dual half-integral), there is no guaranteed polynomial runtime.
- It is possible to compute the convex hull of the integer points in polynomial time by separating *blossom inequalities*. This gives another polynomial-time algorithm to solve CPP (Edmonds/Johnson, 1973).

Directed Case

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

Definition

- A **directed Chinese Postman tour** is a closed directed walk $(e_1, \ldots, e_k)$ in G visiting each edge *at least* once.
- The **Directed Chinese Postman Problem (DCPP)** is to find a directed Chinese Postman tour $C := (e_1, \ldots, e_k)$ that minimizes the total cost $c(C) := \sum_{i=1}^{k} c(e_i)$.

Lemma

The DCPP has a solution if and only if G is strongly connected.

Theorem

The DCPP is equivalent to a minimum-cost network flow problem.

Proof.

Tutorial. □

More

There are a lot of CPP variations:

- **windy**: undirected graph, but direction-dependent cost on the edges
- **mixed**: both directed and undirected edges in the graph
- **clustered**: edges are partitioned into clusters, and all edges in a cluster must be visited before moving to the next cluster
- **rural**: only a subset of edges needs to be traversed
- **generalized rural**: some edges form clusters, only one edge per cluster needs to be visited

Remark

All of these variations lead to NP-complete decision problems.

Rural Postmen

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

Definition

Given a subset $S \subseteq E$, the **Rural Postman Problem (RPP)** is to find a closed walk $(e_1, \ldots, e_k)$ such that $S \subseteq \{e_1, \ldots, e_k\}$ and $\sum_{i=1}^{k} c(e_i)$ is minimal.

Theorem (Lenstra/Rinnooy-Kan, 1976)

RPP is NP-complete.

Proof (Hamiltonian Circuit $\leq$ RPP).

Let H be a graph on n vertices. Construct G by copying H and adding self-loops (v, v) at each vertex v. Choose $c \equiv 1$. Then H has a Hamiltonian circuit if and only if G has a Rural Postman tour of cost $\leq 2n$ visiting all self-loops. □