Graph Theory and Complex Networks: An Introduction

Maarten van Steen

VU Amsterdam, Dept. Computer Science
Room R4.20, steen@cs.vu.nl

Chapter 04: Network traversal

Version: April 14, 2014

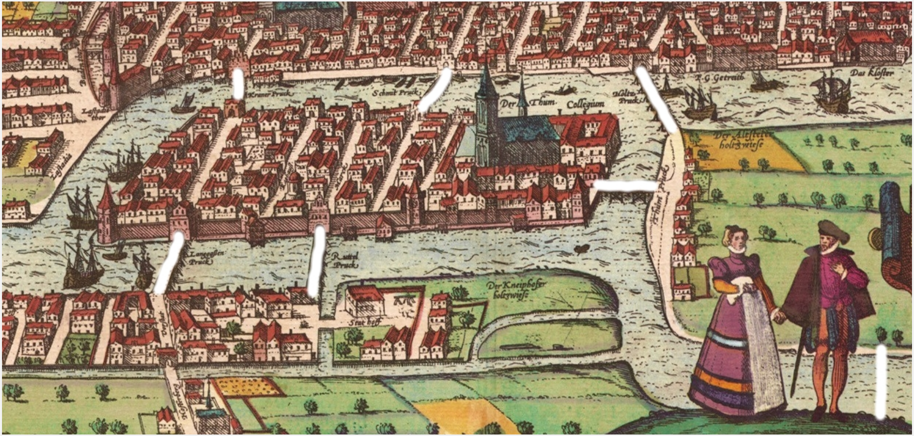
vrije Universiteit amsterdam

Introduction

Algorithms that allow one to move or route through a network

1. Euler tours: visit every edge exactly once.
2. Hamilton cycles: visit every vertex exactly once.

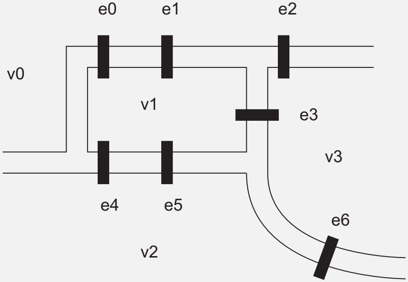
The Königsberg problem

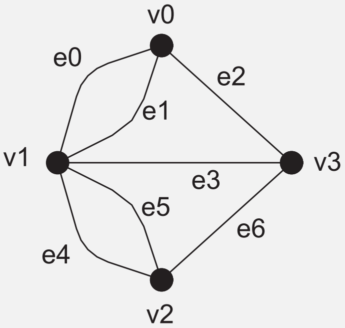


Question

Can one walk through the city and cross each of the seven bridges exactly once?

Modeling the problem in terms of graphs





Euler tours

Definition

A tour of a graph G is a (u, v)-walk in which $u = v$ (i.e., it is a closed walk) and that traverses each edge in G. An Euler tour is a tour in which all edges are traversed exactly once.

Related: Chinese postman problem

- So called because originally formulated by a Chinese mathematician.
- **Issue:** Schedule the round of a postman such that (1) all streets are passed at least once and (2) the total traveled distance is minimal.
- **Solution:** Extend map of streets to a Eulerian graph with minimal weight.

Euler tours

Definition

A tour of a graph G is a (u, v)-walk in which $u = v$ (i.e., it is a closed walk) and that traverses each edge in G. An Euler tour is a tour in which all edges are traversed exactly once.

Related: Chinese postman problem

- So called because originally formulated by a Chinese mathematician.
- **Issue:** Schedule the round of a postman such that (1) all streets are passed at least once and (2) the total traveled distance is minimal.
- **Solution:** Extend map of streets to a Eulerian graph with minimal weight.

Necessary and sufficient conditions

Theorem

A connected graph G (with more than one vertex) has an Euler tour iff it has no vertices of odd degree.

Proof: Euler tour ⇒ no odd-degree vertices

- Let C be an Euler tour starting/ending in vertex v. Let $u \neq v$
- $u \in V(C), \forall \langle w_{in}, u \rangle \in E(C) : \exists \langle u, w_{out} \rangle \in E(C)$.
- Every edge is traversed exactly once ⇒ unique pairing of edges $\langle w_{in}, u \rangle$ and $\langle u, w_{out} \rangle$
- $\delta(u)$ must be even.

Necessary and sufficient conditions

Theorem

A connected graph G (with more than one vertex) has an Euler tour iff it has no vertices of odd degree.

Proof: Euler tour $\Rightarrow$ no odd-degree vertices

- Let C be an Euler tour starting/ending in vertex v. Let $u \neq v$
- $u \in V(C), \forall \langle w_{in}, u \rangle \in E(C) : \exists \langle u, w_{out} \rangle \in E(C)$.
- Every edge is traversed exactly once $\Rightarrow$ unique pairing of edges $\langle w_{in}, u \rangle$ and $\langle u, w_{out} \rangle$
- $\delta(u)$ must be even.

Necessary and sufficient conditions

Theorem

A connected graph G (with more than one vertex) has an Euler tour iff it has no vertices of odd degree.

Proof: Euler tour $\Rightarrow$ no odd-degree vertices

- Let C be an Euler tour starting/ending in vertex v. Let $u \neq v$
- $u \in V(C), \forall\langle w_{in}, u\rangle \in E(C) : \exists\langle u, w_{out}\rangle \in E(C)$.
- Every edge is traversed exactly once $\Rightarrow$ unique pairing of edges $\langle w_{in}, u\rangle$ and $\langle u, w_{out}\rangle$
- $\delta(u)$ must be even.

Necessary and sufficient conditions

Theorem

A connected graph G (with more than one vertex) has an Euler tour iff it has no vertices of odd degree.

Proof: Euler tour $\Rightarrow$ no odd-degree vertices

- Let C be an Euler tour starting/ending in vertex v. Let $u \neq v$
- $u \in V(C), \forall \langle w_{in}, u \rangle \in E(C) : \exists \langle u, w_{out} \rangle \in E(C)$.
- Every edge is traversed exactly once $\Rightarrow$ unique pairing of edges $\langle w_{in}, u \rangle$ and $\langle u, w_{out} \rangle$
- $\delta(u)$ must be even.

Necessary and sufficient conditions

Theorem

A connected graph G (with more than one vertex) has an Euler tour iff it has no vertices of odd degree.

Proof: Euler tour $\Rightarrow$ no odd-degree vertices

- Let C be an Euler tour starting/ending in vertex v. Let $u \neq v$
- $u \in V(C), \forall \langle w_{in}, u \rangle \in E(C) : \exists \langle u, w_{out} \rangle \in E(C)$.
- Every edge is traversed exactly once $\Rightarrow$ unique pairing of edges $\langle w_{in}, u \rangle$ and $\langle u, w_{out} \rangle$
- $\delta(u)$ must be even.

Necessary and sufficient conditions

Proof: no odd-degree vertices $\Rightarrow$ exists Euler tour

- Select v and construct trail P until you need to cross an edge for the second time. Let P end in w.
- Assume $w \neq v \Rightarrow$ entered w once more than left it $\Rightarrow \delta(w)$ is odd. Contradiction. Hence P must end in v.
- $E(P) = E(G) \Rightarrow$ done. Assume $E(P) \subset E(G)$:
  - Let $u \in V(P)$ be incident with edges not in P. Consider $H = G[E(G) - E(P)]$.
  - $\forall x \in V(P) : \delta(x)$ is even $\Rightarrow \forall x \in V(H) : \delta(x)$ is even.
  - Let u lie in component $H' \Rightarrow$ construct similar largest trail P'
  - $P \leftarrow P \cup P'$ and repeat until $E(P) = E(G)$.

Necessary and sufficient conditions

Proof: no odd-degree vertices $\Rightarrow$ exists Euler tour

- Select v and construct trail P until you need to cross an edge for the second time. Let P end in w.
- Assume $w \neq v \Rightarrow$ entered w once more than left it $\Rightarrow \delta(w)$ is odd. Contradiction. Hence P must end in v.
- $E(P) = E(G) \Rightarrow$ done. Assume $E(P) \subset E(G)$:
  - Let $u \in V(P)$ be incident with edges not in P. Consider $H = G[E(G) - E(P)]$.
  - $\forall x \in V(P) : \delta(x)$ is even $\Rightarrow \forall x \in V(H) : \delta(x)$ is even.
  - Let u lie in component $H' \Rightarrow$ construct similar largest trail P'
  - $P \leftarrow P \cup P'$ and repeat until $E(P) = E(G)$.

Necessary and sufficient conditions

Proof: no odd-degree vertices $\Rightarrow$ exists Euler tour

- Select v and construct trail P until you need to cross an edge for the second time. Let P end in w.
- Assume $w \neq v \Rightarrow$ entered w once more than left it $\Rightarrow \delta(w)$ is odd. Contradiction. Hence P must end in v.
- $E(P) = E(G) \Rightarrow$ done. Assume $E(P) \subset E(G)$:
  - Let $u \in V(P)$ be incident with edges not in P. Consider $H = G[E(G) - E(P)]$.
  - $\forall x \in V(P) : \delta(x)$ is even $\Rightarrow \forall x \in V(H) : \delta(x)$ is even.
  - Let u lie in component $H' \Rightarrow$ construct similar largest trail P'
  - $P \leftarrow P \cup P'$ and repeat until $E(P) = E(G)$.

Necessary and sufficient conditions

Proof: no odd-degree vertices $\Rightarrow$ exists Euler tour

- Select v and construct trail P until you need to cross an edge for the second time. Let P end in w.
- Assume $w \neq v \Rightarrow$ entered w once more than left it $\Rightarrow \delta(w)$ is odd. Contradiction. Hence P must end in v.
- $E(P) = E(G) \Rightarrow$ done. Assume $E(P) \subset E(G)$:
  - Let $u \in V(P)$ be incident with edges not in P. Consider $H = G[E(G) - E(P)]$.
  - $\forall x \in V(P) : \delta(x)$ is even $\Rightarrow \forall x \in V(H) : \delta(x)$ is even.
  - Let u lie in component $H' \Rightarrow$ construct similar largest trail P'
  - $P \leftarrow P \cup P'$ and repeat until $E(P) = E(G)$.

Necessary and sufficient conditions

Proof: no odd-degree vertices $\Rightarrow$ exists Euler tour

- Select v and construct trail P until you need to cross an edge for the second time. Let P end in w.
- Assume $w \neq v \Rightarrow$ entered w once more than left it $\Rightarrow \delta(w)$ is odd. Contradiction. Hence P must end in v.
- $E(P) = E(G) \Rightarrow$ done. Assume $E(P) \subset E(G)$:
  - Let $u \in V(P)$ be incident with edges not in P. Consider $H = G[E(G) - E(P)]$.
  - $\forall x \in V(P) : \delta(x)$ is even $\Rightarrow \forall x \in V(H) : \delta(x)$ is even.
  - Let u lie in component $H' \Rightarrow$ construct similar largest trail P'
  - $P \leftarrow P \cup P'$ and repeat until $E(P) = E(G)$.

Necessary and sufficient conditions

Proof: no odd-degree vertices $\Rightarrow$ exists Euler tour

- Select v and construct trail P until you need to cross an edge for the second time. Let P end in w.
- Assume $w \neq v \Rightarrow$ entered w once more than left it $\Rightarrow \delta(w)$ is odd. Contradiction. Hence P must end in v.
- $E(P) = E(G) \Rightarrow$ done. Assume $E(P) \subset E(G)$:
  - Let $u \in V(P)$ be incident with edges not in P. Consider $H = G[E(G) - E(P)]$.
  - $\forall x \in V(P) : \delta(x)$ is even $\Rightarrow \forall x \in V(H) : \delta(x)$ is even.
  - Let u lie in component $H' \Rightarrow$ construct similar largest trail P'
  - $P \leftarrow P \cup P'$ and repeat until $E(P) = E(G)$.

Necessary and sufficient conditions

Proof: no odd-degree vertices $\Rightarrow$ exists Euler tour

- Select v and construct trail P until you need to cross an edge for the second time. Let P end in w.
- Assume $w \neq v \Rightarrow$ entered w once more than left it $\Rightarrow \delta(w)$ is odd. Contradiction. Hence P must end in v.
- $E(P) = E(G) \Rightarrow$ done. Assume $E(P) \subset E(G)$:
  - Let $u \in V(P)$ be incident with edges not in P. Consider $H = G[E(G) - E(P)]$.
  - $\forall x \in V(P) : \delta(x)$ is even $\Rightarrow \forall x \in V(H) : \delta(x)$ is even.
  - Let u lie in component $H' \Rightarrow$ construct similar largest trail P'
  - $P \leftarrow P \cup P'$ and repeat until $E(P) = E(G)$.

Fleury's algorithm

Algorithm (Fleury)

Consider an Eulerian graph G.

1. *Choose an arbitrary vertex $v_0 \in V(G)$ and set $W_0 = v_0$.*
2. *Assume that we have constructed a trail $W_k = [v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k]$. Choose an edge $e_{k+1} = \langle v_k, v_{k+1} \rangle$ from $E(G) \backslash E(W_k)$ such that, preferably, e_{k+1} is not a cut edge of the induced subgraph $G_k = G - E(W_k)$.*
3. *We now have a trail W_{k+1}. If there is no edge $e_{k+2} = \langle v_{k+1}, v_{k+2} \rangle$ to select from $E(G) \backslash E(W_{k+1})$, stop. Otherwise, repeat the previous step.*

Fleury's algorithm

Algorithm (Fleury)

Consider an Eulerian graph G.

1. *Choose an arbitrary vertex $v_0 \in V(G)$ and set $W_0 = v_0$.*
2. *Assume that we have constructed a trail $W_k = [v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k]$. Choose an edge $e_{k+1} = \langle v_k, v_{k+1} \rangle$ from $E(G) \backslash E(W_k)$ such that, preferably, e_{k+1} is not a cut edge of the induced subgraph $G_k = G - E(W_k)$.*
3. *We now have a trail W_{k+1}. If there is no edge $e_{k+2} = \langle v_{k+1}, v_{k+2} \rangle$ to select from $E(G) \backslash E(W_{k+1})$, stop. Otherwise, repeat the previous step.*

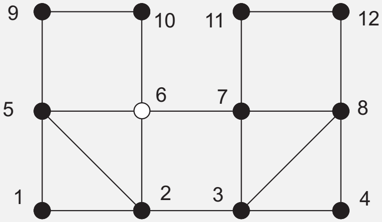
Fleury's algorithm

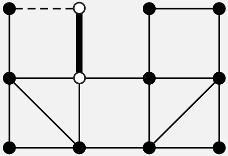
Algorithm (Fleury)

Consider an Eulerian graph G.

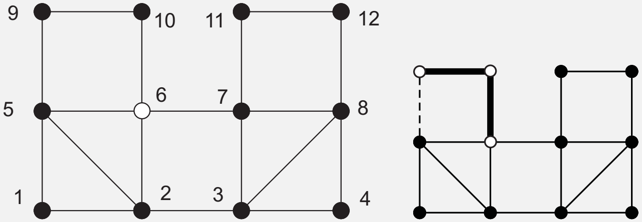
1. *Choose an arbitrary vertex $v_0 \in V(G)$ and set $W_0 = v_0$.*
2. *Assume that we have constructed a trail $W_k = [v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k]$. Choose an edge $e_{k+1} = \langle v_k, v_{k+1} \rangle$ from $E(G) \backslash E(W_k)$ such that, preferably, e_{k+1} is not a cut edge of the induced subgraph $G_k = G - E(W_k)$.*
3. *We now have a trail W_{k+1}. If there is no edge $e_{k+2} = \langle v_{k+1}, v_{k+2} \rangle$ to select from $E(G) \backslash E(W_{k+1})$, stop. Otherwise, repeat the previous step.*

Fleury's algorithm

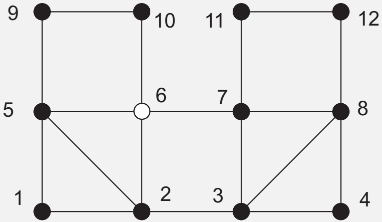


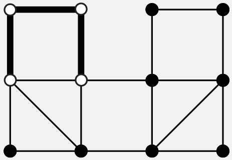


Fleury’s algorithm

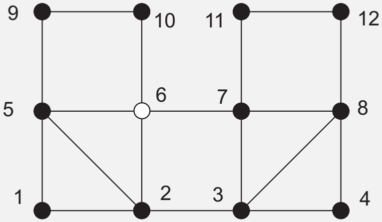


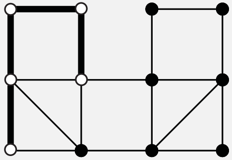
Fleury’s algorithm



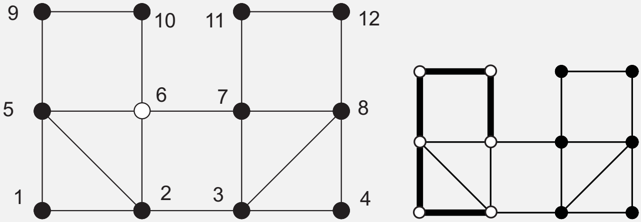


Fleury's algorithm

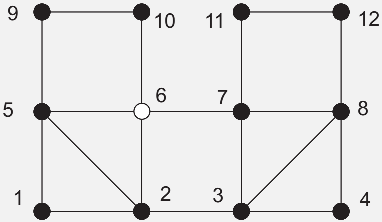


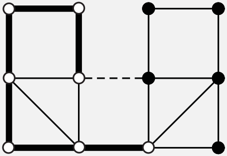


Fleury's algorithm

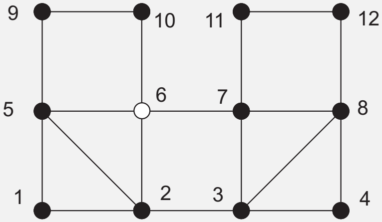


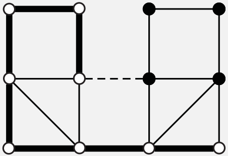
Fleury's algorithm



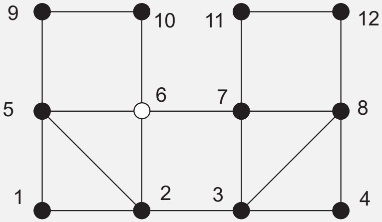


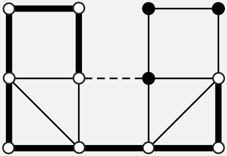
Fleury's algorithm



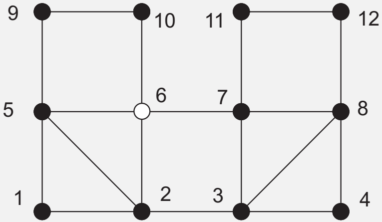


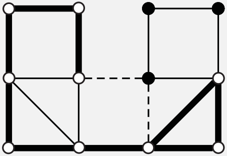
Fleury’s algorithm



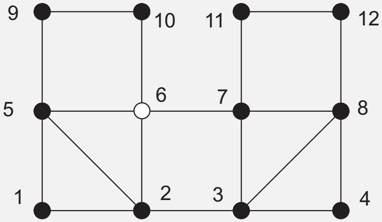


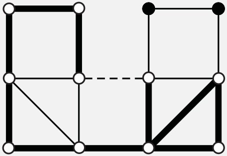
Fleury's algorithm



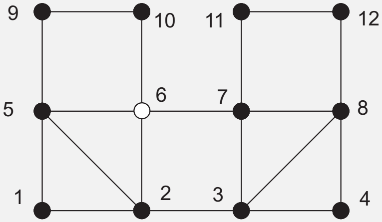


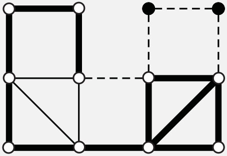
Fleury's algorithm



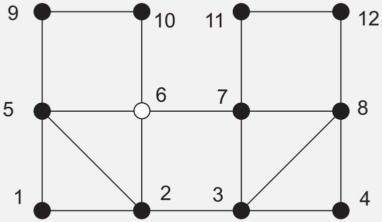


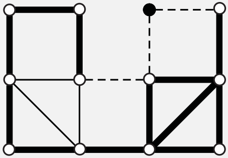
Fleury's algorithm



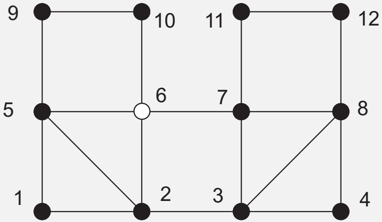


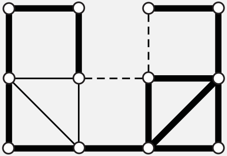
Fleury's algorithm



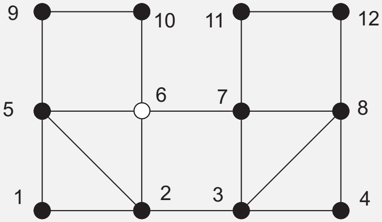


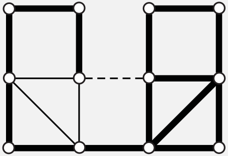
Fleury’s algorithm



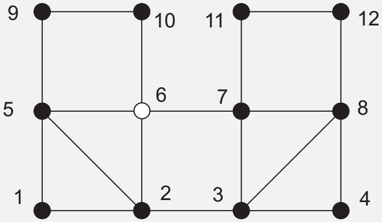


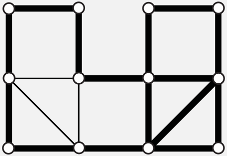
Fleury's algorithm



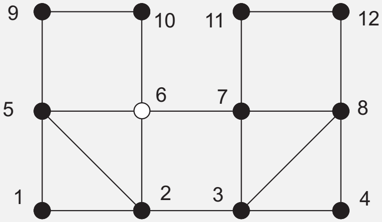


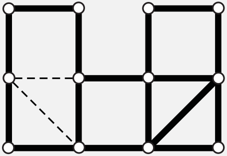
Fleury's algorithm



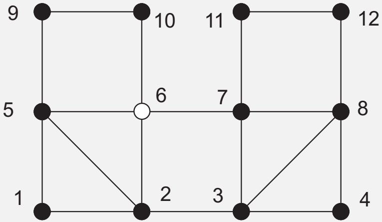


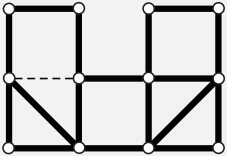
Fleury's algorithm



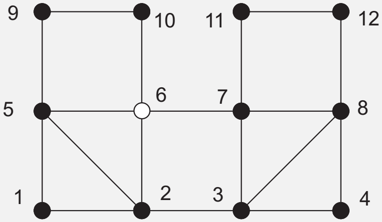


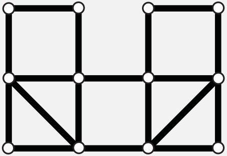
Fleury's algorithm





Fleury's algorithm





Chinese postman problem

Problem as a graph

Model city plan as a weighted graph:

- junction as a vertex
- street as edge, length represented by weight

Find a closed walk with minimal total weight.

Observation

We need to possibly make G Eulerian first by adding edges leading to G^* such that $\sum_{e \in E(G^*) \setminus E(G)} w(e)$ is minimal.

Question

Why may this be so difficult?

Chinese postman problem

Problem as a graph

Model city plan as a weighted graph:

- junction as a vertex
- street as edge, length represented by weight

Find a closed walk with minimal total weight.

Observation

We need to possibly make G Eulerian first by adding edges leading to G^* such that $\sum_{e \in E(G^*) \setminus E(G)} w(e)$ is minimal.

Question

Why may this be so difficult?

Chinese postman problem

Problem as a graph

Model city plan as a weighted graph:

- junction as a vertex
- street as edge, length represented by weight

Find a closed walk with minimal total weight.

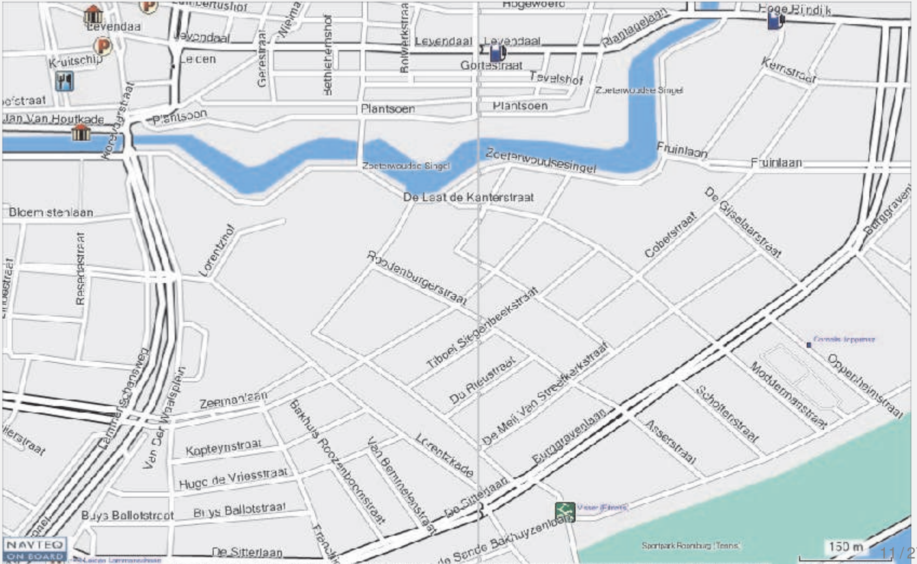
Observation

We need to possibly make G Eulerian first by adding edges leading to G^* such that $\sum_{e \in E(G^*) \setminus E(G)} w(e)$ is minimal.

Question

Why may this be so difficult?

Postman: example



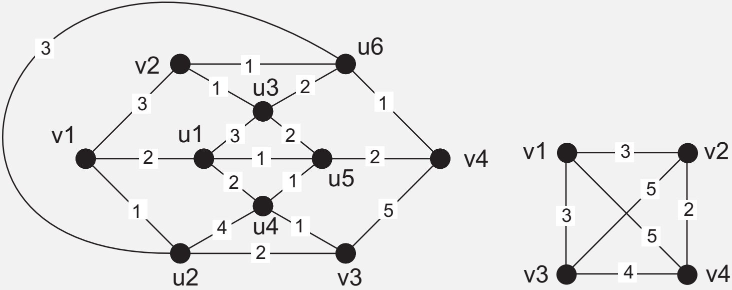
Postman: example

Postman: algorithm

Consider a weighted, connected graph G with odd-degree vertices $V_{odd} = \{v_1, \ldots, v_{2k}\}$ where $k \geq 1$.

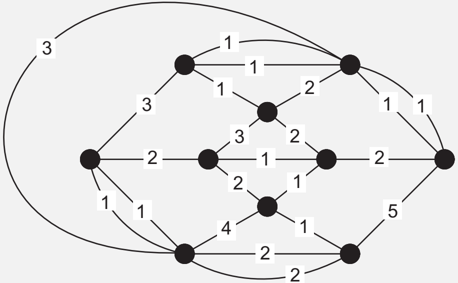
1. For each pair of distinct odd-degree vertices v_i and v_j, find a minimum-weight (v_i, v_j)-path $P_{i,j}$.
2. Construct a weighted complete graph on $2k$ vertices in which vertex v_i and v_j are joined by an edge having weight $w(P_{i,j})$.
3. Find the set E of k edges $e_1, \ldots, e_k$ such that $\sum w(e_i)$ is minimal and no two edges are incident with the same vertex.
4. For each edge $e \in E$, with $e = \langle v_i, v_j \rangle$, duplicate the edges of $P_{i,j}$ in graph G.

Postman: algorithm example



$P_{1,2} = [v_1, v_2]$ (weight: 3)
$P_{1,3} = [v_1, u_2, v_3]$ (weight: 3)
$P_{1,4} = [v_1, u_1, u_5, v_4]$ (weight: 5)
$P_{2,3} = [v_2, u_3, u_5, u_4, v_3]$ (weight: 5)
$P_{2,4} = [v_2, u_6, v_4]$ (weight: 2)
$P_{3,4} = [v_3, u_4, u_5, v_4]$ (weight: 4)

Postman: algorithm example



Hamilton cycles

Definition

A Hamilton path of a connected graph G is a path that contains every vertex of G. A Hamilton cycle is a cycle containing every vertex of G. G is called Hamiltonian if it has a Hamilton cycle.

Important note

There is no known **efficient** algorithm to determine whether a graph is Hamiltonian. Yet, finding Hamilton cycles is important: **Traveling Salesman Problem** (**TSP**).

Hamilton cycles

Definition

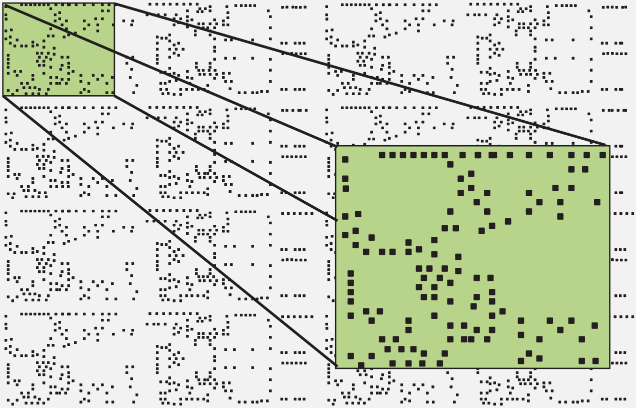
A Hamilton path of a connected graph G is a path that contains every vertex of G. A Hamilton cycle is a cycle containing every vertex of G. G is called Hamiltonian if it has a Hamilton cycle.

Important note

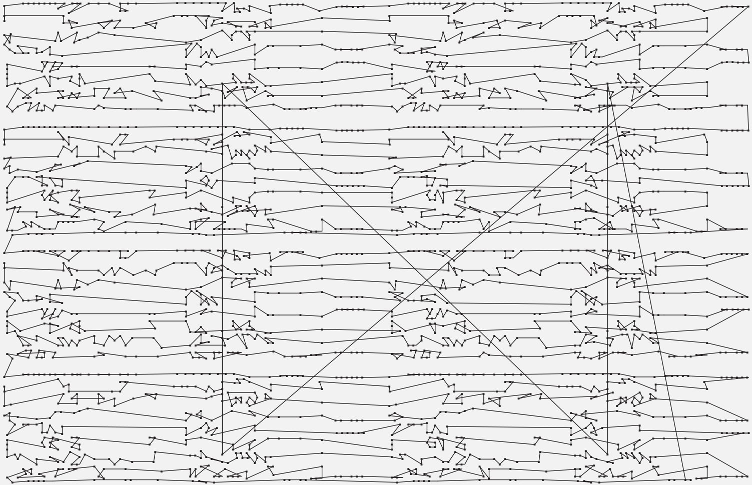
There is no known **efficient** algorithm to determine whether a graph is Hamiltonian. Yet, finding Hamilton cycles is important: **Traveling Salesman Problem** (**TSP**).

TSP: Example

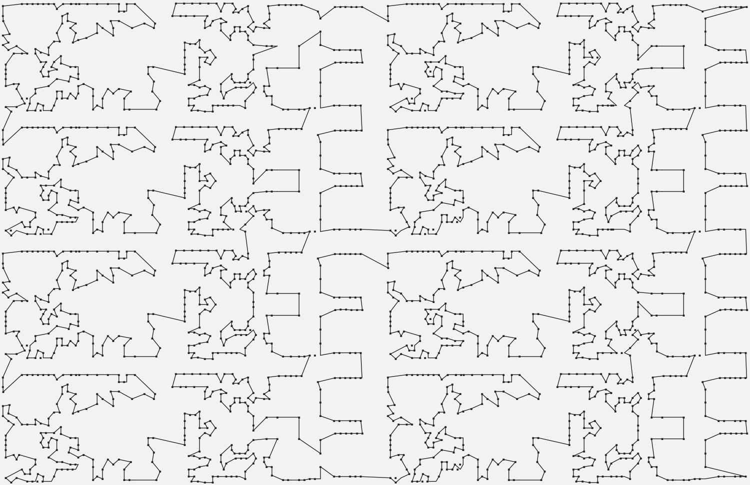
Drilling holes: Consider a board for electrical circuits. To fasten the components, we need to drill holes. **Issue:** Which track should the drilling machine follow?



TSP: Example



TSP: Example



Some formal properties

Theorem

G *Hamiltonian* $\Rightarrow \forall S \subset V(G), S \neq \emptyset : \omega(G - S) \leq |S|$.

Proof

- Let C be a Hamilton cycle $\Rightarrow$ every vertex is visited exactly once $\Rightarrow \omega(C - S) \leq |S|$.
- $V(C) = V(G) \Rightarrow \omega(G - S) \leq \omega(C - S)$.

Some formal properties

Theorem

G Hamiltonian $\Rightarrow \forall S \subset V(G), S \neq \emptyset : \omega(G - S) \leq |S|$.

Proof

- Let C be a Hamilton cycle $\Rightarrow$ every vertex is visited exactly once $\Rightarrow \omega(C - S) \leq |S|$.
- $V(C) = V(G) \Rightarrow \omega(G - S) \leq \omega(C - S)$.

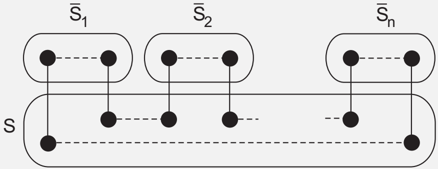
Some formal properties

Theorem

G Hamiltonian $\Rightarrow \forall S \subset V(G), S \neq \emptyset : \omega(G-S) \leq |S|$.

Proof

- Let C be a Hamilton cycle $\Rightarrow$ every vertex is visited exactly once $\Rightarrow \omega(C-S) \leq |S|$.
- $V(C) = V(G) \Rightarrow \omega(G-S) \leq \omega(C-S)$.



Some formal properties: Dirac

Theorem (Dirac)

G is simple with $n \geq 3$ vertices and $\forall v : \delta(v) \geq n/2 \Rightarrow G$ is Hamiltonian.

Proof: by induction

- For $n = 3$ vertices: trivial. Assume the theorem has been proven correct for graphs with $k \geq 3$ vertices.
- Let G have $k+1$ vertices, constructed from any graph G^* with k vertices, by adding a vertex u and joining u to at least $(k+1)/2$ other vertices.
- Let $C^* = [v_1, v_2, \ldots, v_k]$ be a Hamilton cycle in G^*.
- Vertex u is joined to at least $(k+1)/2$ vertices from C^*
  $\Rightarrow$ there is at least a pair v_i and v_{i+1} that are adjacent in C^*
- Construct a new cycle $C = [v_1, \ldots, v_i, u, v_{i+1}, v_k]$

Some formal properties: Dirac

Theorem (Dirac)

G is simple with $n \geq 3$ vertices and $\forall v : \delta(v) \geq n/2 \Rightarrow$ G is Hamiltonian.

Proof: by induction

- For $n = 3$ vertices: trivial. Assume the theorem has been proven correct for graphs with $k \geq 3$ vertices.
- Let G have $k + 1$ vertices, constructed from any graph G^* with k vertices, by adding a vertex u and joining u to at least $(k+1)/2$ other vertices.
- Let $C^* = [v_1, v_2, \ldots, v_k]$ be a Hamilton cycle in G^*.
- Vertex u is joined to at least $(k+1)/2$ vertices from C^*
  $\Rightarrow$ there is at least a pair v_i and v_{i+1} that are adjacent in C^*
- Construct a new cycle $C = [v_1, \ldots, v_i, u, v_{i+1}, v_k]$

Some formal properties: Dirac

Theorem (Dirac)

G is simple with $n \geq 3$ vertices and $\forall v : \delta(v) \geq n/2 \Rightarrow G$ is Hamiltonian.

Proof: by induction

- For $n = 3$ vertices: trivial. Assume the theorem has been proven correct for graphs with $k \geq 3$ vertices.
- Let G have $k+1$ vertices, constructed from any graph G^* with k vertices, by adding a vertex u and joining u to at least $(k+1)/2$ other vertices.
- Let $C^* = [v_1, v_2, \ldots, v_k]$ be a Hamilton cycle in G^*.
- Vertex u is joined to at least $(k+1)/2$ vertices from C^* $\Rightarrow$ there is at least a pair v_i and v_{i+1} that are adjacent in C^*
- Construct a new cycle $C = [v_1, \ldots, v_i, u, v_{i+1}, v_k]$

Some formal properties: Dirac

Theorem (Dirac)

G is simple with $n \geq 3$ vertices and $\forall v : \delta(v) \geq n/2 \Rightarrow G$ is Hamiltonian.

Proof: by induction

- For $n = 3$ vertices: trivial. Assume the theorem has been proven correct for graphs with $k \geq 3$ vertices.
- Let G have $k+1$ vertices, constructed from any graph G^* with k vertices, by adding a vertex u and joining u to at least $(k+1)/2$ other vertices.
- Let $C^* = [v_1, v_2, \ldots, v_k]$ be a Hamilton cycle in G^*.
- Vertex u is joined to at least $(k+1)/2$ vertices from C^*
  $\Rightarrow$ there is at least a pair v_i and v_{i+1} that are adjacent in C^*
- Construct a new cycle $C = [v_1, \ldots, v_i, u, v_{i+1}, v_k]$

Some formal properties: Dirac

Theorem (Dirac)

G is simple with $n \geq 3$ vertices and $\forall v : \delta(v) \geq n/2 \Rightarrow G$ is Hamiltonian.

Proof: by induction

- For $n = 3$ vertices: trivial. Assume the theorem has been proven correct for graphs with $k \geq 3$ vertices.
- Let G have $k+1$ vertices, constructed from any graph G^* with k vertices, by adding a vertex u and joining u to at least $(k+1)/2$ other vertices.
- Let $C^* = [v_1, v_2, \ldots, v_k]$ be a Hamilton cycle in G^*.
- Vertex u is joined to at least $(k+1)/2$ vertices from C^* $\Rightarrow$ there is at least a pair v_i and v_{i+1} that are adjacent in C^*
- Construct a new cycle $C = [v_1, \ldots, v_i, u, v_{i+1}, v_k]$

Some formal properties: Dirac

Theorem (Dirac)

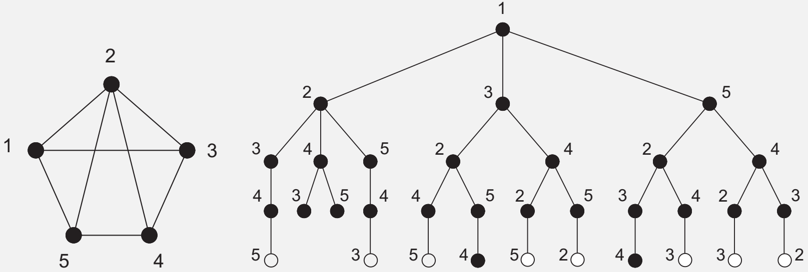
G is simple with $n \geq 3$ vertices and $\forall v : \delta(v) \geq n/2 \Rightarrow G$ is Hamiltonian.

Proof: by induction

- For $n = 3$ vertices: trivial. Assume the theorem has been proven correct for graphs with $k \geq 3$ vertices.
- Let G have $k+1$ vertices, constructed from any graph G^* with k vertices, by adding a vertex u and joining u to at least $(k+1)/2$ other vertices.
- Let $C^* = [v_1, v_2, \ldots, v_k]$ be a Hamilton cycle in G^*.
- Vertex u is joined to at least $(k+1)/2$ vertices from C^* $\Rightarrow$ there is at least a pair v_i and v_{i+1} that are adjacent in C^*
- Construct a new cycle $C = [v_1, \ldots, v_i, u, v_{i+1}, v_k]$

Finding Hamilton cycles

Brute force: Select a vertex *v*, and explore all possible Hamilton paths originating from *v*, and check whether they can be expanded to a cycle:



Posa: applying rotational transformations

Algorithm (Posa)

Randomly select $u \in V(G)$, forming the starting point of path P. Let $last(P) = u$ denote the current end point of P.

1. *Randomly select $v \in N(last(P))$, such that*
   1. *Preferably, $v \notin V(P)$*
   2. *If $v \in V(P) \Rightarrow v$ has not been previously selected as neighbor of an end point before.*

   If no such vertex exists, stop.
2. *If $v \notin V(P)$, set $P \leftarrow P + \langle last(P), v \rangle$.*

Posa: applying rotational transformations

Algorithm (Posa)

Randomly select $u \in V(G)$, forming the starting point of path P. Let $last(P) = u$ denote the current end point of P.

1. *Randomly select $v \in N(last(P))$, such that*
   1. *Preferably, $v \notin V(P)$*
   2. *If $v \in V(P) \Rightarrow v$ has not been previously selected as neighbor of an end point before.*

   If no such vertex exists, stop.
2. *If $v \notin V(P)$, set $P \leftarrow P + \langle last(P), v\rangle$.*

Posa: applying rotational transformations

Algorithm (Posa)

Randomly select $u \in V(G)$, forming the starting point of path P. Let $last(P) = u$ denote the current end point of P.

1. *Randomly select $v \in N(last(P))$, such that*
    1. *Preferably, $v \notin V(P)$*
    2. *If $v \in V(P) \Rightarrow v$ has not been previously selected as neighbor of an end point before.*

    If no such vertex exists, stop.
2. *If $v \notin V(P)$, set $P \leftarrow P + \langle last(P), v \rangle$.*

Posa: applying rotational transformations

Algorithm (Posa - cntd)

3. *If $v \in V(P)$, apply a* ***rotational transformation of P*** *using edge $\langle last(P), v\rangle$:*

leading to P^*. If $last(P^*)$ has not yet been end point for paths of the current length, $P \leftarrow P^*$.

4. *$V(P) = V(G)$ and $\langle u, last(P)\rangle \in E(G) \Rightarrow$ found a Hamilton cycle. Otherwise, continue with step 1.*

Posa: applying rotational transformations

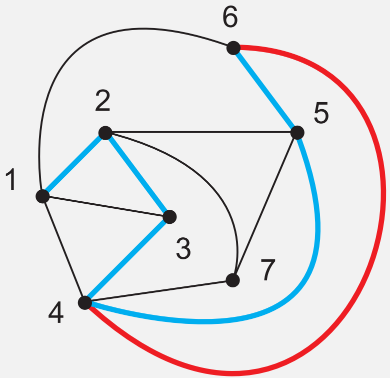
Algorithm (Posa - cntd)

3. *If $v \in V(P)$, apply a* ***rotational transformation of** P* *using edge $\langle last(P), v \rangle$:*

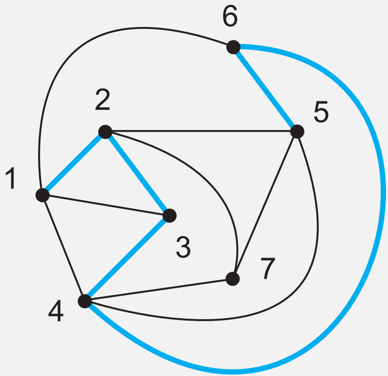
leading to P^*. If $last(P^*)$ has not yet been end point for paths of the current length, $P \leftarrow P^*$.

4. *$V(P) = V(G)$ and $\langle u, last(P) \rangle \in E(G) \Rightarrow$ found a Hamilton cycle. Otherwise, continue with step 1.*

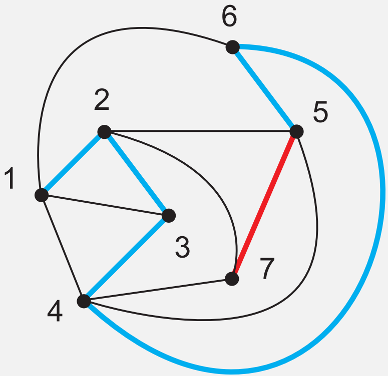
Posa example



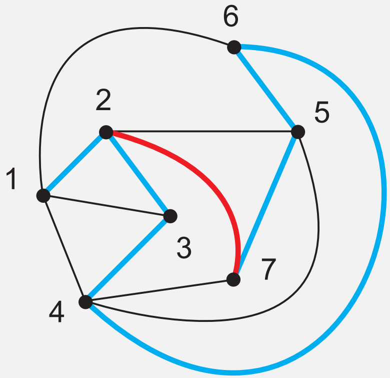
Posa example



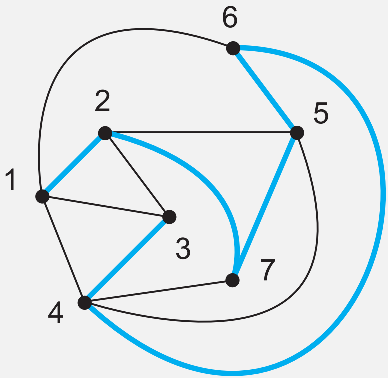
Posa example



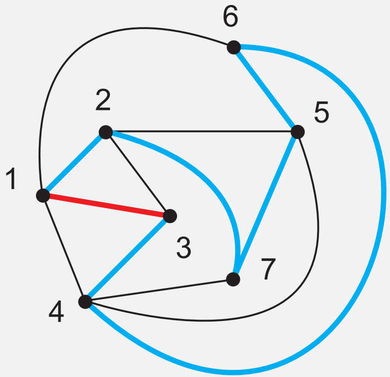
Posa example



Posa example



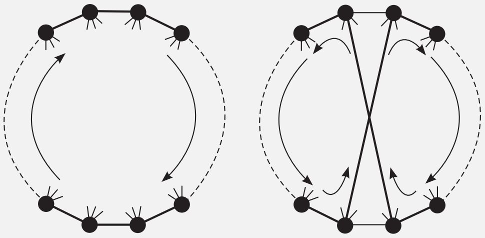
Posa example



Optimal Hamilton cycle

Basic idea

We want to find a Hamilton cycle with **minimal weight** $\Rightarrow$ extend graph to a complete one in which distance between two vertices reflects real-world distance.

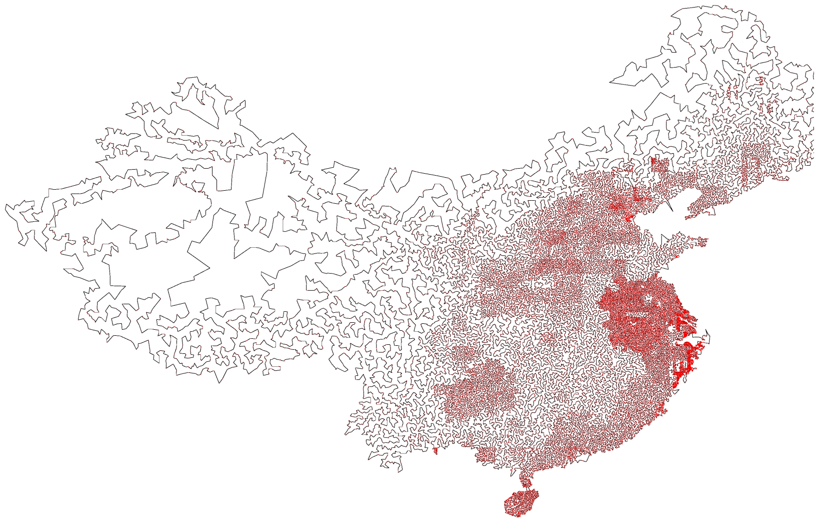


(a) (b)

(a) Start with an arbitrary cycle

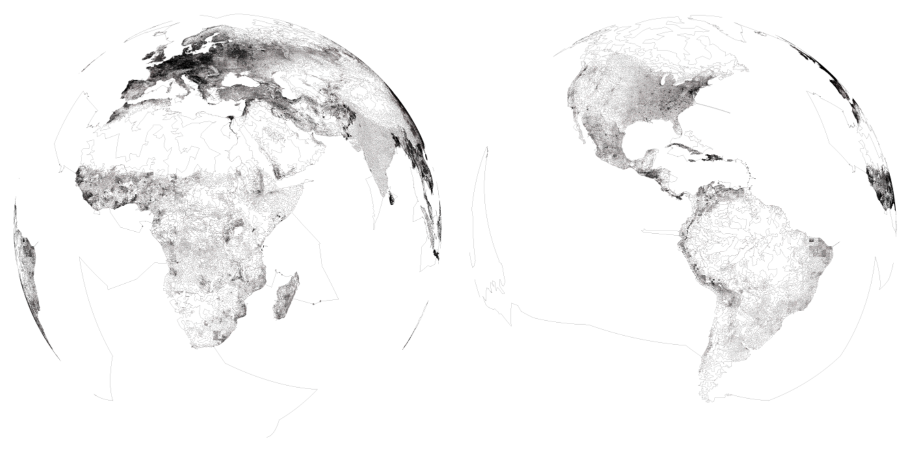
(b) If swapping edges improve weight $\Rightarrow$ better cycle

Hamilton example: China



71,000 cities, 4,566,563 edges $\leq$ 0.024% longer than optimal one.

Hamilton example: The world



1,904,711 cities, 7,516,353,779 edges $\leq$ 0.076% longer than optimal one.