The Chinese postman problem

Giovanni Righini

University of Milan

Definitions

Given an undirected, connected, edge-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the so-called Chinese Postman Problem is the problem of finding a minimum cost tour in $\mathcal{G}$, traversing all its edges at least once.

An Euler graph is defined by two properties:

- it is connected;
- all vertex degrees are even.

An Euler graph always allows for an Euler tour, i.e. a tour traversing each edge exactly once.

Therefore, if $\mathcal{G}$ is an Euler graph, then the minimum cost is $\sum_{e \in E} c_e$.

A mathematical model

$$
\begin{aligned}
\text{minimize } z = \sum_{e \in \mathcal{E}} c_e x_e & \\
\text{s.t. } \sum_{e \in \delta(i)} x_e = 2y_i & \qquad \forall i \in \mathcal{V} \\
y_i \in \mathbb{Z}_+ & \qquad \forall i \in \mathcal{V} \\
x_e \geq 1 & \qquad \forall e \in \mathcal{E}.
\end{aligned}
$$

Insert additional copies of some edges in order to trasform $\mathcal{G}$ into an Euler graph.

The algorithm

If $\mathcal{G}$ is connected but it is not an Euler graph, then it has some vertices with odd degree.

Let $\mathcal{O}$ be the set of vertices with odd degree in $\mathcal{G}$.

The cardinality of $\mathcal{O}$ is certainly even (easy to prove).

To obtain an Euler graph from $\mathcal{G}$, it is necessary and sufficient to increase by one unit the degree of all vertices in $\mathcal{O}$.

Hence, a minimum cost set of $|\mathcal{O}|/2$ paths connecting the vertices of $\mathcal{O}$ in pairs must be found: it is a perfect matching in a complete auxiliary graph $\mathcal{H} = (\mathcal{O}, \mathcal{P})$, where the cost of each edge in $\mathcal{P}$ is the cost of a shortest path between its endpoints.

The algorithm

Hence, the algorithm works as follows:

1. Find all vertices with odd degree: $\mathcal{O}$.
2. Compute the all-pairs shortest paths between them (costs of the edges in $\mathcal{P}$).
3. Find a minimum cost perfect matching on the auxiliary graph $\mathcal{H} = (\mathcal{O}, \mathcal{P})$.
4. Add the edges of the selected shortest paths to the initial edge set $\mathcal{E}$, generating an Euler multi-graph (the edge set is a multi-set, where each edge has an associated multiplicity).
5. Find an Euler tour on the resulting Euler multi-graph.

All these steps can be done in strictly polynomial time.

Complexity

Finding all vertices with odd degree can be done in $O(m)$.
The cardinality $|\mathcal{O}|$ is upper bounded by n.

All-pairs shortest paths on a complete graph with $|\mathcal{O}|$ vertices can be computed in $O(|\mathcal{O}|^3)$, i.e. $O(n^3)$ (Floyd-Warshall algorithm).

A minimum cost perfect matching in a graph with $|\mathcal{O}|$ vertices can be computed in $O(|\mathcal{O}|^3)$, i.e. $O(n^3)$ (Edmonds algorithm).

The Euler multi-graph is obtained in $O(n^2)$, since $O(n)$ paths are made by $O(n)$ edges each. The multiplicity for each edge is $O(n)$: hence the total multiplicity m' is $O(mn)$.

An Euler tour in an Euler multi-graph with total multiplicity m', can be computed in $O(m')$, i.e. $O(mn)$ (Hierholzer algorithm).

Overall complexity: $O(n^3)$.

Hierholzer algorithm (1893)

Algorithm *Hierholzer algorithm*

Initialization
while $(k < m)$ **do**
  $p \leftarrow Select$
  $C \leftarrow FindCycle(p)$
  $InsertList(S, p, C)$
$Return(S)$

Hierholzer algorithm: *Initialization*

Algorithm *Initialization*

for $v \in \mathcal{V}$ **do**
  $degree(v) \leftarrow 0$
  $\delta(v) \leftarrow \emptyset$
$m \leftarrow 0$
for $e \in \mathcal{E}$ **do**
  $m \leftarrow m + \mu(e)$
  $\lambda(e) \leftarrow \mu(e)$
  $[i, j] \leftarrow e$
  $\delta(i) \leftarrow \delta(i) \cup \{e\}$
  $degree(i) \leftarrow degree(i) + 1$
  $\delta(j) \leftarrow \delta(j) \cup \{e\}$
  $degree(j) \leftarrow degree(j) + 1$
for $v \in \mathcal{V}$ **do**
  $q(v) \leftarrow \delta(v)$
$S \leftarrow \{1\}$
$p \leftarrow Head(S)$
$k \leftarrow 0$

S: sequence of vertices in the current tour (linked list).

p: pointer to current vertex in S.

$degree(v)$: residual degree of $v \in \mathcal{V}$.

$\delta(v)$: star of $v \in \mathcal{V}$.

$q(v)$: current pointer within $\delta(v)$.

$\mu(e)$: multiplicity of $e \in \mathcal{E}$.

$\lambda(e)$: residual multiplicity of $e \in \mathcal{E}$.

m: total multiplicity of edges in $\mathcal{E}$.

k: n. of edges in the current tour.

Complexity: $O(m)$.

Hierholzer algorithm: *Select*

Select scans S and returns the first vertex with strictly positive residual degree.

Algorithm *Select*

---

while $(\mathit{degree}(p.\mathit{vertex}) = 0)$ **do**
  $p \leftarrow \mathit{Next}(p)$
$\mathit{Return}(p)$

Complexity: $O(m)$ overall.

Hierholzer algorithm: *FindCycle*

Algorithm *FindCycle*

$C \leftarrow nil$
$i \leftarrow p.vertex$
repeat
  $j \leftarrow FindSuccessor(i)$
  $Append(C, j)$
  $degree(i) \leftarrow degree(i) - 1$
  $degree(j) \leftarrow degree(j) - 1$
  $e \leftarrow [i, j]$
  $\lambda(e) \leftarrow \lambda(e) - 1$
  $k \leftarrow k + 1$
  $i \leftarrow j$
until $i = p.vertex$
$Return(C)$

Complexity: $O(m)$ calls, $O(m)$ complexity for each call, but $O(m)$ complexity overall.

Hierholzer algorithm: *FindSuccessor*

Algorithm *FindSuccessor*(i)

while $(\lambda(q(i).edge) = 0)$ **do**
  $q(i) \leftarrow Next(q(i))$
$[i, j] \leftarrow q(i).edge$
$Return(j)$

Complexity: $O(m)$ overall. Each edge $e \in \delta(i)$ is scanned $\mu(e)$ times.

Hierholzer algorithm: *InsertList*

Algorithm $\mathit{InsertList}(S, p, C)$

$\mathit{Next}(\mathit{Tail}(C)) \leftarrow \mathit{Next}(p)$
$\mathit{Next}(p) \leftarrow \mathit{Head}(C)$

Complexity: $O(1)$ for each call.

Hierholzer algorithm: complexity

Initialization has complexity $O(m)$.

Select: the degree of each vertex is tested $O(|\delta(i)|)$ times. The total time for tests is $O(m)$.
The step forward is done $O(m)$ times in constant time; the total times for steps is $O(m)$.

FindCycle without *FindSuccessor*: since each edge is considered $\mu(e)$ times, the total complexity is $O(m)$.

FindSuccessor: each star is scanned once and, then, each edge is considered $2\mu(e)$ times.
Hence, the total time complexity is $O(m)$.

InsertList: it takes constant time. Each cycle includes at least two edges. Then, the n. of cycle insertions is $O(m)$.

Hierholzer algorithm complexity: $O(m)$.

The Chinese postman problem on digraphs

A similar method works for the directed version of the problem.

A digraph is an Euler digraph if and only if

- it is strongly connected;
- the indegree and the outdegree of each node are equal.

If the given digraph $\mathcal{D} = (\mathcal{N}, \mathcal{A})$ is not an Euler digraph, one must select a minimum cost set of arcs to add to the digraph, to balance its in- and out-degrees.

A mathematical model

$$
\begin{aligned}
\text{minimize } z = & \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} \\
\text{s.t. } & \sum_{(i,j)\in\delta^+(i)} x_{ij} = \sum_{(j,i)\in\delta^-(i)} x_{ji} && \forall i \in \mathcal{N} \\
& x_{ij} \geq 1 && \forall (i,j) \in \mathcal{A} \\
& x_{ij} \in \mathbb{Z}_+ && \forall (i,j) \in \mathcal{A}.
\end{aligned}
$$

This originates an instance of the minimum cost transportation problem.

Reformulation as a min cost transportation problem

$$
\begin{aligned}
\text{minimize } z' = & \sum_{(i,j)\in\mathcal{A}} c'_{ij} x'_{ij} \\
\text{s.t.} & \sum_{(i,j)\in\delta^+(i)} x'_{ij} = s_i && \forall i \in \mathcal{S} \\
& \sum_{(i,j)\in\delta^-(j)} x'_{ij} = d_j && \forall j \in \mathcal{T} \\
& x'_{ij} \in \mathbb{Z}_+ && \forall (i,j) \in \mathcal{S} \times \mathcal{T}.
\end{aligned}
$$

Origins $\mathcal{S}$: nodes with in-degree larger than the out-degree: the difference is s_i $\forall i \in \mathcal{S}$.

Destinations $\mathcal{T}$: nodes with out-degree larger than the in-degree: the difference is d_j $\forall j \in \mathcal{T}$.

Cost c'_{ij}: cost of an $i - j$ shortest path $\forall (i,j) \in \mathcal{S} \times \mathcal{T}$.

Integrality requirements on the x'_{ij} variables can be dropped, thanks to the integrality property (s and d are integer).

Complexity: the same of a min cost flow problem.