Chvátal Rank

Procedure to obtain a linear description of P_I:

i. Let $P^{(0)} = P$ and $i = 0$;
ii. Add all possible Gomory-Chvátal cuts to $P^{(i)}$ to obtain
$$P^{(i+1)} = \{x \in P^{(i)} \mid x \text{ satisfies every GC-cut for } P^{(i)}\}.$$
iii. Iterate with $i = i + 1$;

This gives a chain $P = P^{(0)} \supseteq P^{(1)} \supseteq \ldots \supseteq P_I$.

Theorem 14.37 (Schrijver 1980).

If $P^{(i)}$ is a rational polytope, then $P^{(i+1)}$ is also a rational polytope.

Proof:... □

Corollary 14.38.

$P^{(k)} = P_I$ for some integer k.

The least k for which $P^{(k)} = P_I$ is called Chvátal rank of P.

Cutting-Plane Algorithms

Let P be a polyhedron, and consider an ILP of type $\max\{w^T \mid x \in P,\ x \text{ integral}\}$. Given some classes of cut-inequalities, we may try the following procedure:

1. Find an optimal solution x^* of $\max\{w^T \mid x \in P\}$.
2. If x^* is integral, we are done.
3. Otherwise, try to find a cut-inequality violated by x^*, add it to P, and iterate.

If we are lucky, the cutting-plane algorithm terminates with an integral optimal solution, and thus solves the ILP.

But, off course, it might terminate at some point where we do not find a violated cut.

In any case: the optimal value of the LP-relaxation in each iteration provides an upper bound!

Useful in, for example, *branch-and-bound* (or *branch-and-cut*) algorithms.

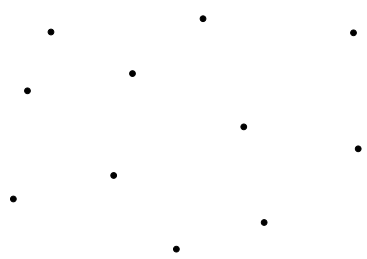
Chapter 15:
The Traveling Salesperson Problem

(cp. Cook, Cunningham, Pulleyblank & Schrijver, Chapter 7)

The Traveling Salesperson Problem

Definition 15.1.

Given a finite set of n points V and a cost c_{uv} of travel between each pair $u, v \in V$, a tour (or Hamiltonian circuit) is a circuit that passes through each point in V exactly once. The traveling salesperson problem (TSP) asks for a tour of minimal cost w.r.t. c.



TSP is one of the "classical", most famous NP-hard problems.

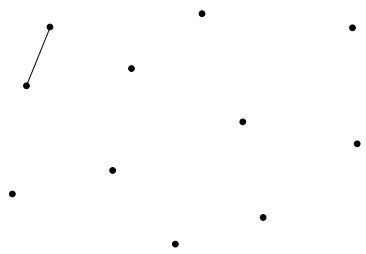
Note: Enumerating all possible tours and searching for an optimal one takes way too long. Example: ...

TSP problems frequently appear in practice, and relatively large ones can now be solved efficiently to optimality.

The Traveling Salesperson Problem

Definition 15.1.

Given a finite set of n points V and a cost c_{uv} of travel between each pair $u, v \in V$, a tour (or Hamiltonian circuit) is a circuit that passes through each point in V exactly once. The traveling salesperson problem (TSP) asks for a tour of minimal cost w.r.t. c.



TSP is one of the "classical", most famous NP-hard problems.

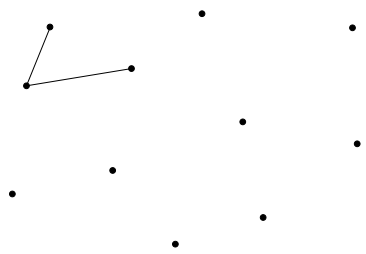
Note: Enumerating all possible tours and searching for an optimal one takes way too long. Example: ...

TSP problems frequently appear in practice, and relatively large ones can now be solved efficiently to optimality.

The Traveling Salesperson Problem

Definition 15.1.

Given a finite set of n points V and a cost c_{uv} of travel between each pair $u, v \in V$, a tour (or Hamiltonian circuit) is a circuit that passes through each point in V exactly once. The traveling salesperson problem (TSP) asks for a tour of minimal cost w.r.t. c.



TSP is one of the "classical", most famous NP-hard problems.

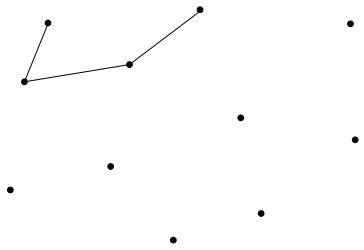
Note: Enumerating all possible tours and searching for an optimal one takes way too long. Example: ...

TSP problems frequently appear in practice, and relatively large ones can now be solved efficiently to optimality.

The Traveling Salesperson Problem

Definition 15.1.

Given a finite set of n points V and a cost c_{uv} of travel between each pair $u, v \in V$, a tour (or Hamiltonian circuit) is a circuit that passes through each point in V exactly once. The traveling salesperson problem (TSP) asks for a tour of minimal cost w.r.t. c.



TSP is one of the "classical", most famous NP-hard problems.

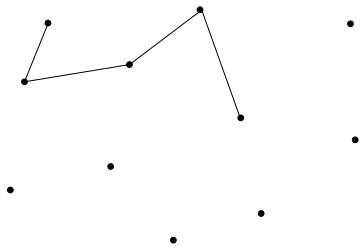
Note: Enumerating all possible tours and searching for an optimal one takes way too long. Example: ...

TSP problems frequently appear in practice, and relatively large ones can now be solved efficiently to optimality.

The Traveling Salesperson Problem

Definition 15.1.

Given a finite set of n points V and a cost c_{uv} of travel between each pair $u, v \in V$, a tour (or Hamiltonian circuit) is a circuit that passes through each point in V exactly once. The traveling salesperson problem (TSP) asks for a tour of minimal cost w.r.t. c.



TSP is one of the "classical", most famous NP-hard problems.

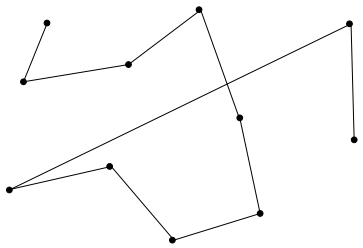
Note: Enumerating all possible tours and searching for an optimal one takes way too long. Example: ...

TSP problems frequently appear in practice, and relatively large ones can now be solved efficiently to optimality.

The Traveling Salesperson Problem

Definition 15.1.

Given a finite set of n points V and a cost c_{uv} of travel between each pair $u, v \in V$, a tour (or Hamiltonian circuit) is a circuit that passes through each point in V exactly once. The traveling salesperson problem (TSP) asks for a tour of minimal cost w.r.t. c.



TSP is one of the "classical", most famous NP-hard problems.

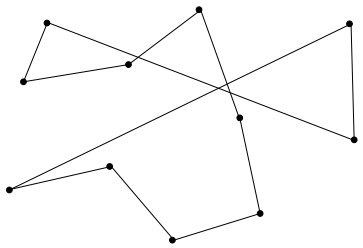
Note: Enumerating all possible tours and searching for an optimal one takes way too long. Example: ...

TSP problems frequently appear in practice, and relatively large ones can now be solved efficiently to optimality.

The Traveling Salesperson Problem

Definition 15.1.

Given a finite set of n points V and a cost c_{uv} of travel between each pair $u, v \in V$, a tour (or Hamiltonian circuit) is a circuit that passes through each point in V exactly once. The traveling salesperson problem (TSP) asks for a tour of minimal cost w.r.t. c.



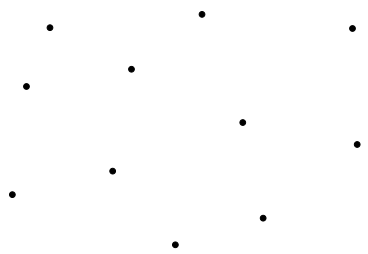
TSP is one of the "classical", most famous NP-hard problems.

Note: Enumerating all possible tours and searching for an optimal one takes way too long. Example: ...

TSP problems frequently appear in practice, and relatively large ones can now be solved efficiently to optimality.

Heuristics

Heuristics are methods which cannot be guaranteed to produce optimal solutions, but (hopefully) fairly good ones. They are usually fast and easy to implement.



Nearest Neighbor Algorithm:

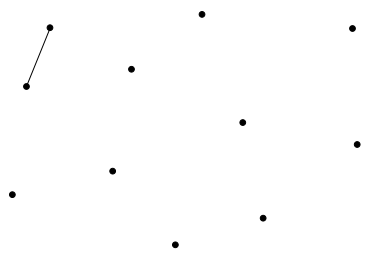
- Start at any node;
- While possible, go to the nearest neighbor not yet visited;
- Return to the starting node;

Average cost of NN-tours on TSPLIB-library are about 1,26 times the cost of corresponding optimal tours.

Exercise: NN can produce a tour that is arbitrarily bad for general TSP.

Heuristics

Heuristics are methods which cannot be guaranteed to produce optimal solutions, but (hopefully) fairly good ones. They are usually fast and easy to implement.



Nearest Neighbor Algorithm:

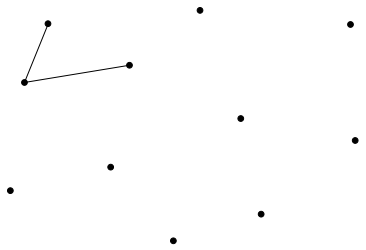
- Start at any node;
- While possible, go to the nearest neighbor not yet visited;
- Return to the starting node;

Average cost of NN-tours on TSPLIB-library are about 1,26 times the cost of corresponding optimal tours.

Exercise: NN can produce a tour that is arbitrarily bad for general TSP.

Heuristics

Heuristics are methods which cannot be guaranteed to produce optimal solutions, but (hopefully) fairly good ones. They are usually fast and easy to implement.



Nearest Neighbor Algorithm:

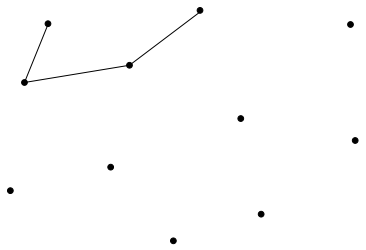
- Start at any node;
- While possible, go to the nearest neighbor not yet visited;
- Return to the starting node;

Average cost of NN-tours on TSPLIB-library are about 1,26 times the cost of corresponding optimal tours.

Exercise: NN can produce a tour that is arbitrarily bad for general TSP.

Heuristics

Heuristics are methods which cannot be guaranteed to produce optimal solutions, but (hopefully) fairly good ones. They are usually fast and easy to implement.



Nearest Neighbor Algorithm:

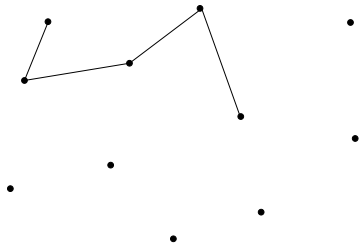
- Start at any node;
- While possible, go to the nearest neighbor not yet visited;
- Return to the starting node;

Average cost of NN-tours on TSPLIB-library are about 1,26 times the cost of corresponding optimal tours.

Exercise: NN can produce a tour that is arbitrarily bad for general TSP.

Heuristics

Heuristics are methods which cannot be guaranteed to produce optimal solutions, but (hopefully) fairly good ones. They are usually fast and easy to implement.



Nearest Neighbor Algorithm:

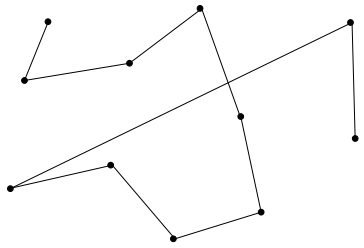
- Start at any node;
- While possible, go to the nearest neighbor not yet visited;
- Return to the starting node;

Average cost of NN-tours on TSPLIB-library are about 1,26 times the cost of corresponding optimal tours.

Exercise: NN can produce a tour that is arbitrarily bad for general TSP.

Heuristics

Heuristics are methods which cannot be guaranteed to produce optimal solutions, but (hopefully) fairly good ones. They are usually fast and easy to implement.



Nearest Neighbor Algorithm:

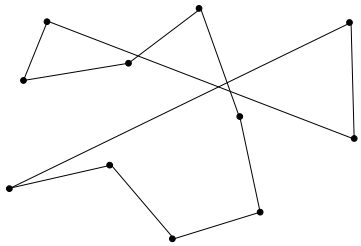
- Start at any node;
- While possible, go to the nearest neighbor not yet visited;
- Return to the starting node;

Average cost of NN-tours on TSPLIB-library are about 1,26 times the cost of corresponding optimal tours.

Exercise: NN can produce a tour that is arbitrarily bad for general TSP.

Heuristics

Heuristics are methods which cannot be guaranteed to produce optimal solutions, but (hopefully) fairly good ones. They are usually fast and easy to implement.



Nearest Neighbor Algorithm:

- Start at any node;
- While possible, go to the nearest neighbor not yet visited;
- Return to the starting node;

Average cost of NN-tours on TSPLIB-library are about 1,26 times the cost of corresponding optimal tours.

Exercise: NN can produce a tour that is arbitrarily bad for general TSP.

Triangle Inequality

The cost function c satisfies the Triangle-inequality (or Δ-inequality) if for all $u, v, w \in V$ holds $c_{uv} + c_{vw} \geq c_{uw}$.

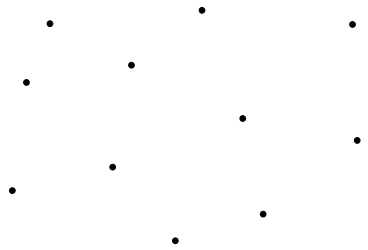
If the Δ-inequality holds, the TSP problem is called Euclidean TSP or metric TSP. (Still NP-hard).

We consider the symmetric TSP where $c_{uv} = c_{vu}$ for all $u, v \in V$. Otherwise, the problem is called assymetric TSP.

Remark: For the Euclidean TSP, the cost of a tour produced by the NN-heuristic is never more than $\frac{1}{2}\lceil \log_2 |V| \rceil + \frac{1}{2}$ times the cost of an optimal tour. This bound is tight.

Thus, the NN-heuristic is not of practical interest if we wish to give a good worst-case bound!

Insertion Methods

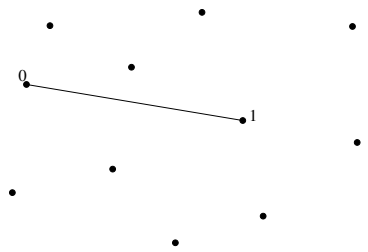


General form of an insertion method:

- Start with a tour consisting of two nodes;
- Add remaining nodes one by one, in such a way that the tour cost is increased by a minimum amount;

Question: What is a good order in which the nodes are inserted?

Insertion Methods

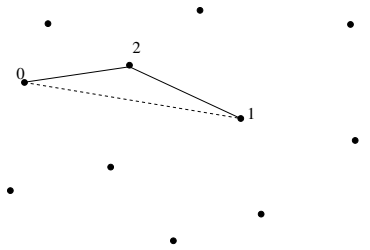


General form of an insertion method:

- Start with a tour consisting of two nodes;
- Add remaining nodes one by one, in such a way that the tour cost is increased by a minimum amount;

Question: What is a good order in which the nodes are inserted?

Insertion Methods

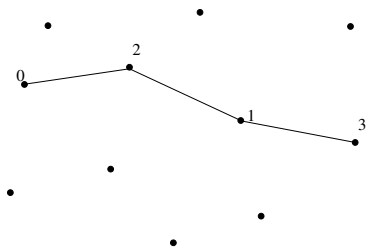


General form of an insertion method:

- Start with a tour consisting of two nodes;
- Add remaining nodes one by one, in such a way that the tour cost is increased by a minimum amount;

Question: What is a good order in which the nodes are inserted?

Insertion Methods

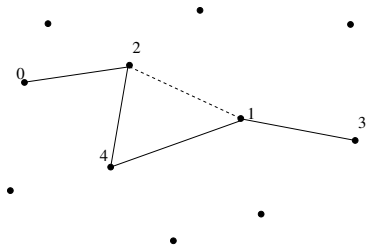


General form of an insertion method:

- Start with a tour consisting of two nodes;
- Add remaining nodes one by one, in such a way that the tour cost is increased by a minimum amount;

Question: What is a good order in which the nodes are inserted?

Insertion Methods

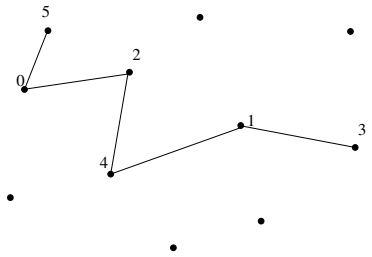


General form of an insertion method:

- Start with a tour consisting of two nodes;
- Add remaining nodes one by one, in such a way that the tour cost is increased by a minimum amount;

Question: What is a good order in which the nodes are inserted?

Insertion Methods

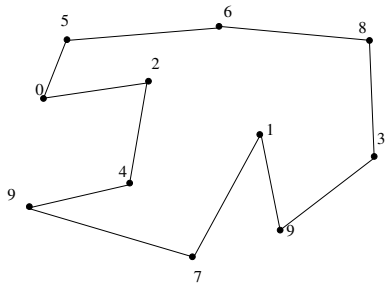


General form of an insertion method:

- Start with a tour consisting of two nodes;
- Add remaining nodes one by one, in such a way that the tour cost is increased by a minimum amount;

Question: What is a good order in which the nodes are inserted?

Insertion Methods



General form of an insertion method:

- Start with a tour consisting of two nodes;
- Add remaining nodes one by one, in such a way that the tour cost is increased by a minimum amount;

Question: What is a good order in which the nodes are inserted?

Farthest Insertion

Farthest Insertion Method:

- Start with two nodes that are endnodes of some high-cost edge;
- for each uninserted node v, compute the minimum cost d_v between v and a node in the current tour;
- choose as the next node to be inserted the one *maximizing* d_v;

Seems a bit counterintuitive to choose the node maximizing d_v.
However, farthest insertion works well in practice:

Remark: Farthest Insertion is usually the best insertion method in practice.

On TSPLIB-problems, the length of Farthest Insertion tours are, on average, about 1,16 times the length of the corresponding optimal tours.

Nearest and Cheapest Insertion

Nearest Insertion: choose as the next node to be inserted one for which the cost to any node in the tour is minimum.

Cheapest Insertion: choose as the next node to be inserted one for which the cost of the tour is increased the least.

Ususally, Farthest Insertion produces better tours than Cheapest or Nearest Insertion.

Remarks on worst-case analysis:

- On an Euclidean TSP instance, any insertion heuristic produces a tour of length at most $\lceil \log_2 |V| \rceil + 1$ times the length of the optimum.
- Nearest and Cheapest Insertion tours are at most twice the length of the corresponding optimal tours.
- Interestingly, although Farthest Insertion Methods are the best insertion methods in practice, no better worst-case bound than the one for general insertion methods could be found.

Christofides' Heuristic

The Christofides Heuristic always produces a tour of cost at most $\frac{3}{2}$ times the optimum (assuming $c \geq 0$ and Δ-inequality.)

Christofides' Heuristic (1976):

- Compute a MST $T = (V, E)$ in the complete graph $G(V)$;
- Let $W \subseteq V$ denote the nodes having odd degree in T. Compute a min-cost perfect matching M of the subgraph $G[W]$ induced by W;
- Let $J = E(T) \cup M$ (with two copies for each $e \in E(T) \cap M$.)Then $H = (V, J)$ is connected and each node in H has even degree.
- If $\delta_H(v) = 2$ for all $v \in V$, then J is the edge set of a tour and we terminate;
- Otherwise, choose a node v of degree at least 4.
- Choose two edges $\{u, v\}$ and $\{v, w\}$ in J such that $H' = (V, J \setminus \{\{u, v\}, \{v, w\}\} \cup \{u, w\})$ remains connected and has even degree at each node.
- Repeat this process until all nodes are incident with two edges in J.