The Traveling Salesman Problem

Professor Z. Max Shen

IEOR 151

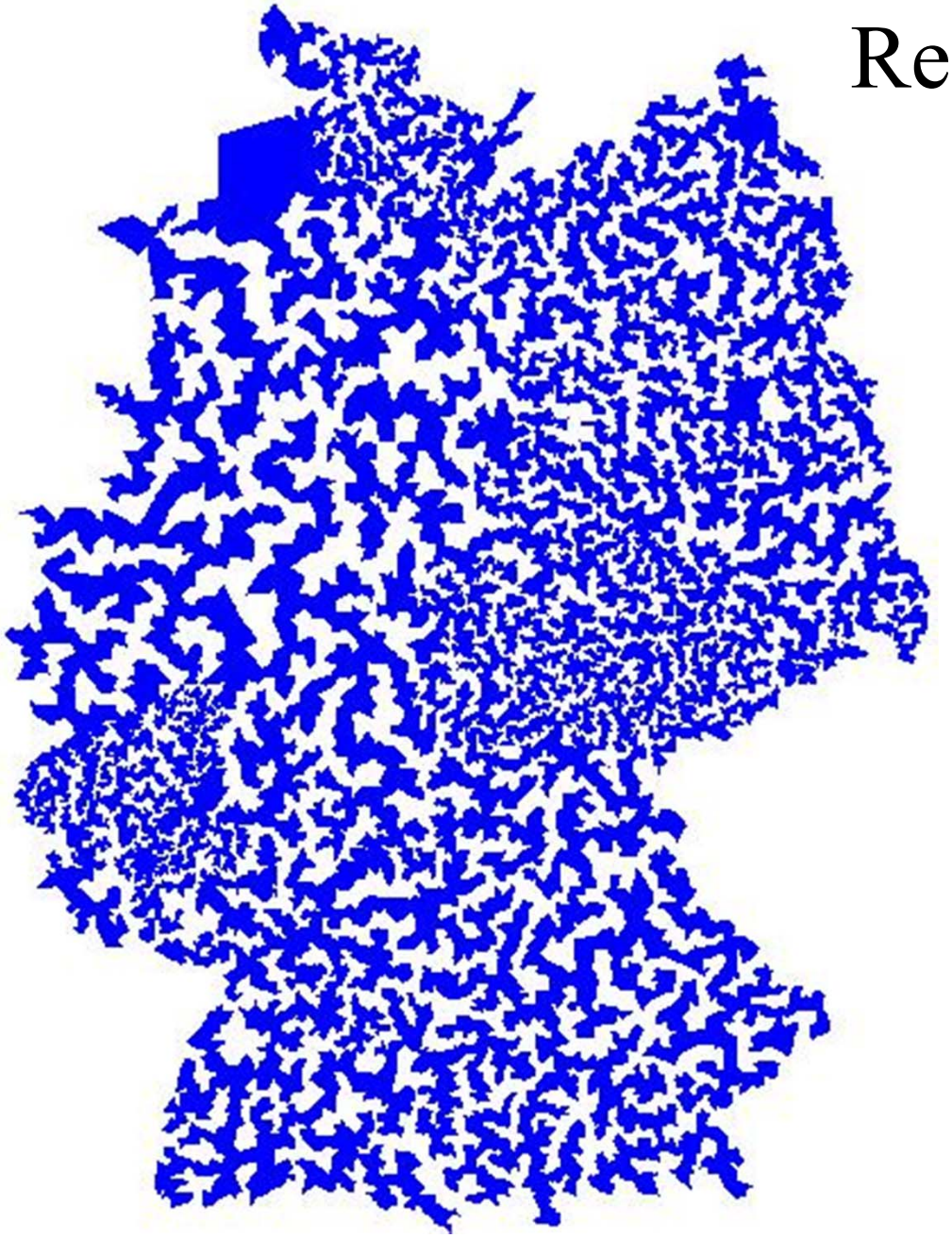
TSP Model

Let $G=(V, E)$ be a complete undirected graph with vertices V, $|V|= n$, and the edges E and let d_{ij} be the *length* of edge (i, j).

The objective is to find a tour that visits each vertex exactly once and whose total length is as small as possible.

Applications of TSP

- The TSP naturally arises as a subproblem in many transportation and logistics applications, for example the problem of arranging school bus routes to pick up the children in a school district (Vehicle Routine Problem).
- Another classic example is the scheduling of a machine to drill holes in a circuit board or other object, the cost of travel is the time it takes to move the drill head from one hole to the next.



Recent Achievement

An optimal tour through 15,112 cities in Germany, it was proved to be the optimal solution in April, 2001

CPU Time 22.6 years

Preliminary: Minimum Spanning Tree

- Given an undirected graph $G=(V, E)$ with arc lengths d_{ij}'s.
- A spanning tree T is a subgraph of G that is
  - a tree (a connected acyclic graph), and
  - spans (touches) all nodes.
- Every spanning tree has (n-1) arcs.
- Length of a spanning tree T is $\Sigma_{(i,j)\in T} d_{ij}$.
- The minimum spanning tree problem is to find a spanning tree of minimum length.

An algorithm for MST

- **Step 1:** find the shortest arc in the network. If there are more than one, pick any one randomly. Highlight this arc and the nodes connected.

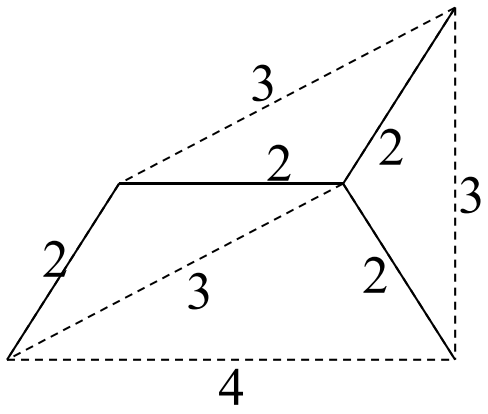
- **Step 2:** Pick the next shortest arc, unless it forms a cycle with the arcs already highlighted before. Highlight the arc and the nodes connected.

- **Step 3:** If all arcs are connected, then we are done. Otherwise, repeat Step 2.

A Minimum Spanning Tree Based Heuristic

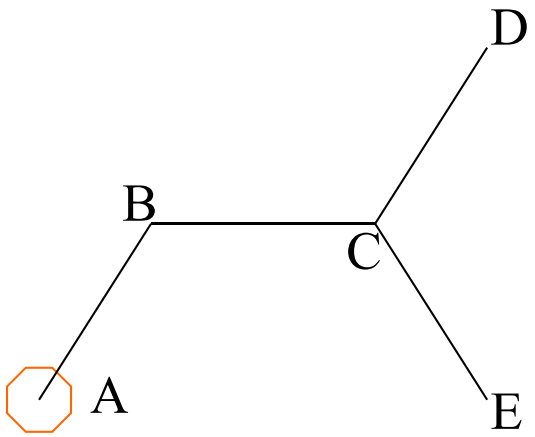
- *Step 1*: Construct a minimum spanning tree
- *Step 2*: Let the *root* be an arbitrary vertex
- *Step 3*: Traverse all the vertices by depth-first search, record the sequence of vertices (both visited and unvisited)
- *Step 4*: Use shortcut strategy to generate a feasible tour

Step 1: Construct a minimum spanning tree

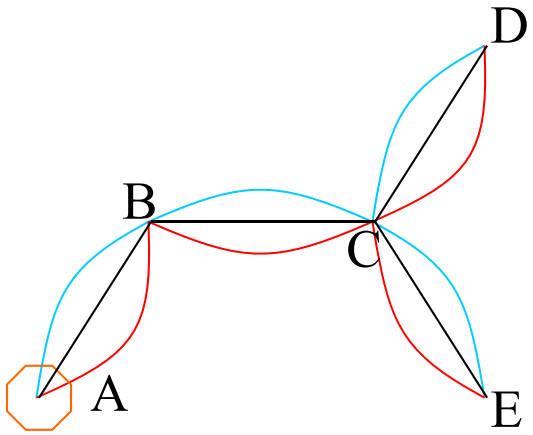


MST could be solved in $O(n^2)$, .which is also a Lower bound for TSP, $W^* \leq L^*$.

Step 2: Let the *root* be an arbitrary vertex



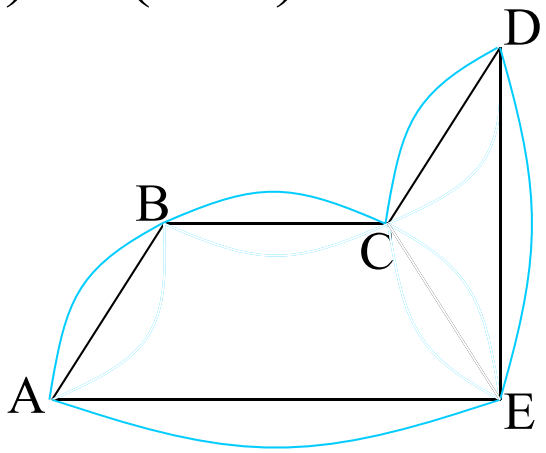
Step 3: Traverse all the vertices



The sequence is A-B-C-D-C-E-C-B-A, the length of this tour is $2W^*$.

Step 4: Use shortcut to generate the MST tour

A-B-C-D-(C)-E-(C-B)-A



The MST tour is A-B-C-D-E-A, the length of this TSP tour is less or equal to $2W^*$.

Worst-case Analysis

Let L^{MST} denotes the length of the tour generated by above strategy, then we have

$$L^{MST} \leq 2W^* \leq 2L^*$$

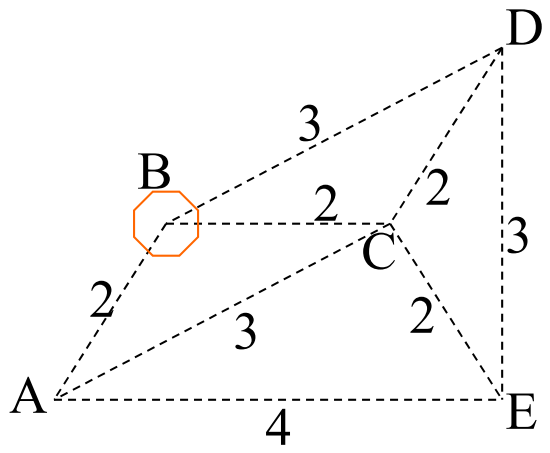
Where W^* denotes the length of the minimum spanning tree.

And this bound is tight.

Nearest Insertion Heuristic

- *Step 1*: Choose an arbitrary node v and let the cycle C consist of only v.
- *Step 2*: Find a node outside C closest to a node in C; call it k.
- *Step 3*: Find an edge $\{i, j\}$ in C such that $d_{ik}+d_{kj}-d_{ij}$ is minimal.
- *Step 4*: Construct a new cycle C by replacing $\{i, j\}$ with $\{i, k\}$ and $\{k, j\}$.
- *Step 5*: If the current cycle C contains all the vertices, stop. Otherwise, go to *Step 2*.

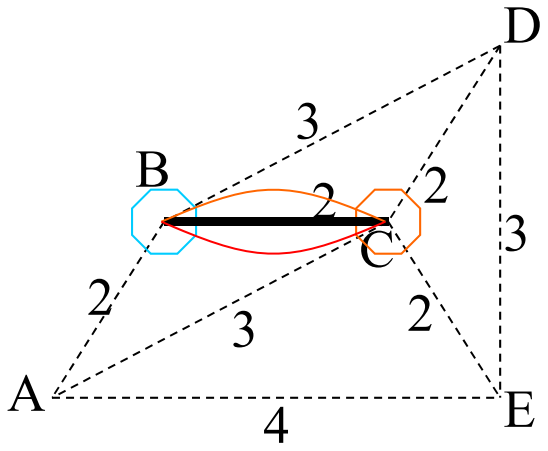
Step 1: Choose an arbitrary node v and let the cycle C consist of only v



The length of current tour C is zero.

Step 2-3-4: one iteration:

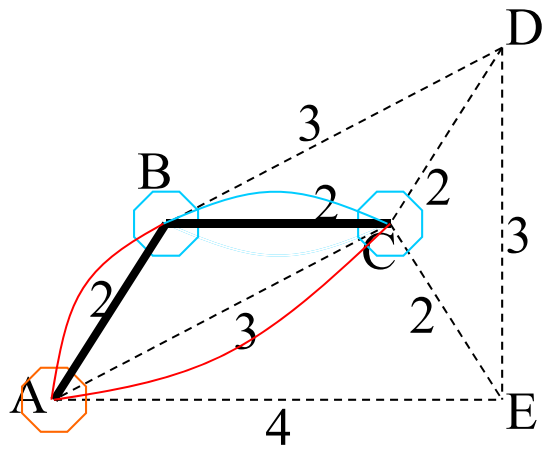
find nearest node, find shortest insertion, insert



The length of *C* is no more than the twice of the dark line length (equal in this iteration).

Step 2-3-4: second iteration:

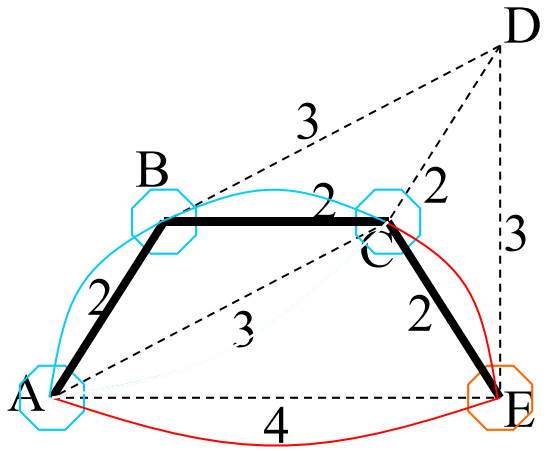
find nearest node, find shortest insertion, insert



The length of current tour *C* is no more than the twice of the dark line length.

Step 2-3-4: third iteration:

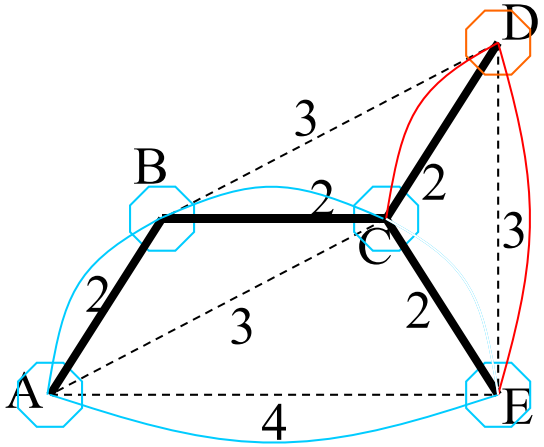
find nearest node, find shortest insertion, insert



The length of current tour *C* is no more than the twice of the dark line length.

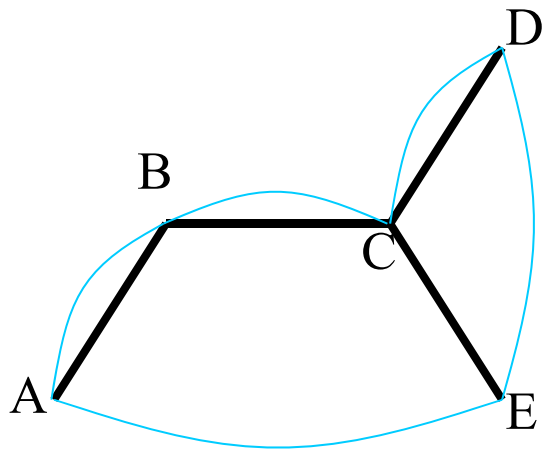
Step 2-3-4: fourth iteration:

find nearest node, find shortest insertion, insert



The length of current tour *C* is no more than the twice of the dark line length.

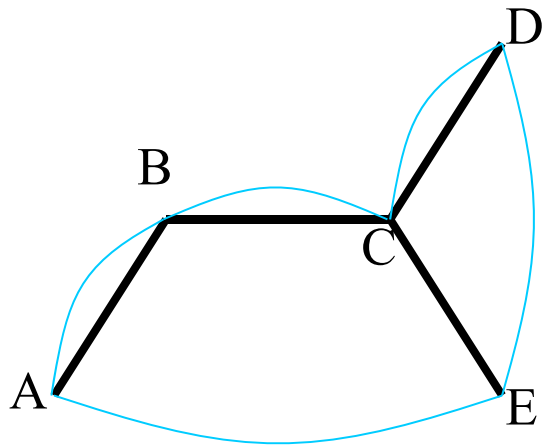
Step 5: Final result A-B-C-D-E-A



The length of current tour *C* is no more than the twice of the dark line length.

Worst Case Analysis

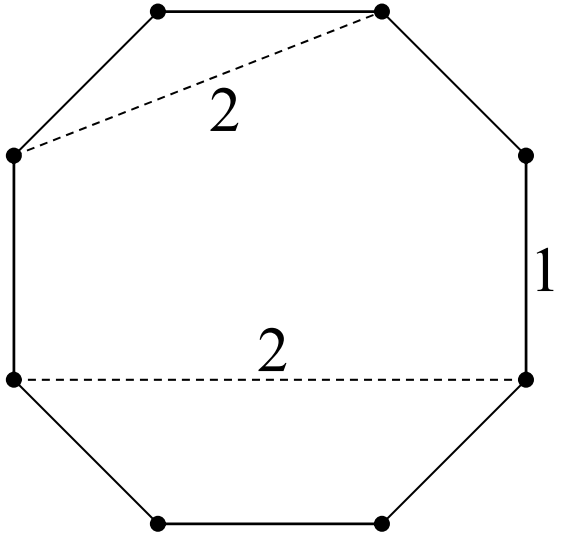
The length of current tour C is no more than the twice of the dark line length.



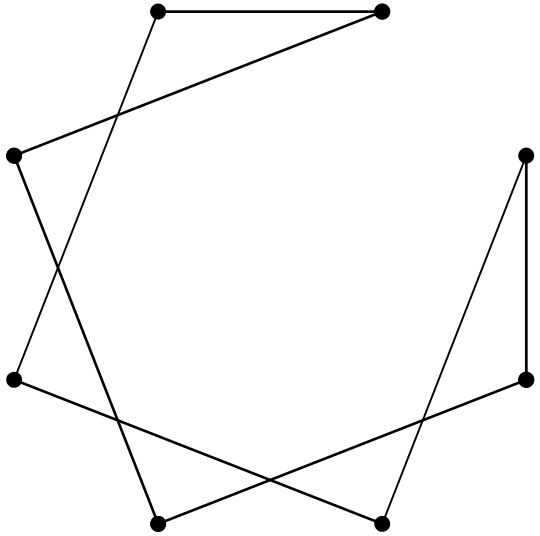
The dark line is nothing, but a minimum spanning tree (exactly same as Prim's Algorithm), thus

$$L^{MST} \leq 2W^* \leq 2L^*$$

An example for nearest insertion algorithm



The original network



The tour generated by the Nearest Insertion Algorithm

Christofides' Heuristic

- 1. Find the minimum spanning tree
- 2. Find the odd-degree nodes of the MST
- 3. Find the optimal matching of the odd-degree nodes and add links of the matching to the MST
- 4. Draw an Euler tour on the resulting graph

K-opt Heuristic

K-opt procedures ($k \geq 2$) belong to the class of neighborhood search algorithm (alternative called local search algorithm).

They are some of the oldest and the most extensively used heuristics developed for the Travel Salesman Problem.

Definition: *l-exchange* is a procedure that replace l edges currently in the tour by l new edges so that the result is again a traveling salesman tour.

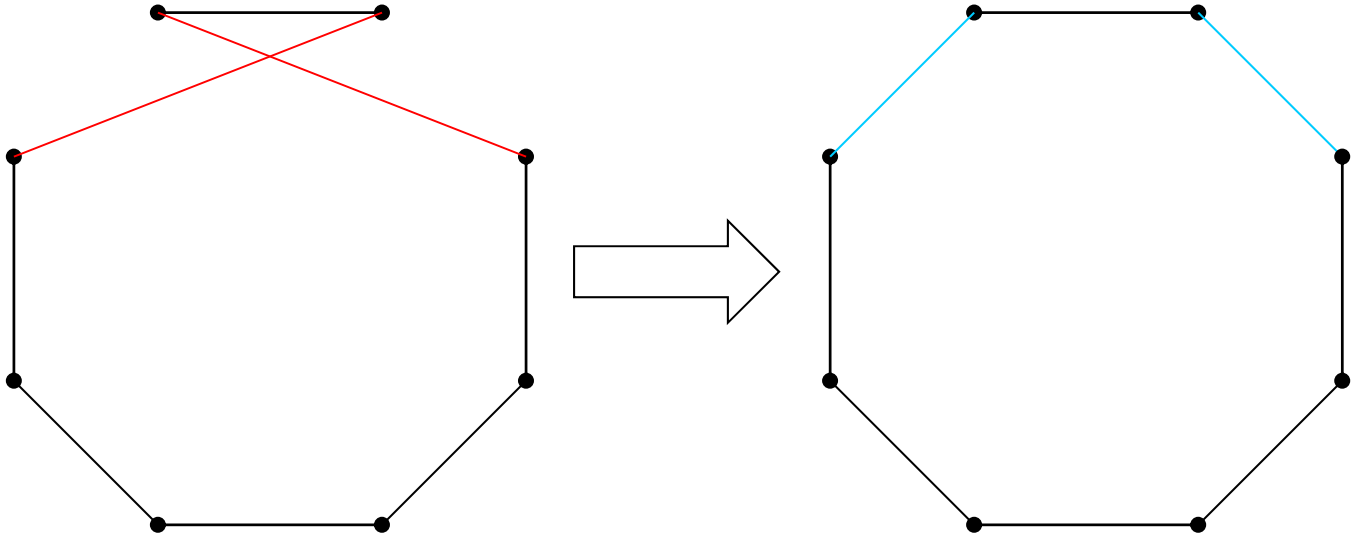
For instance, given a tour through set V, say

$$\{i_1, i_2, \ldots, i_u, i_{u+1}, \ldots, i_v, i_{v+1}, \ldots, i_n\}$$

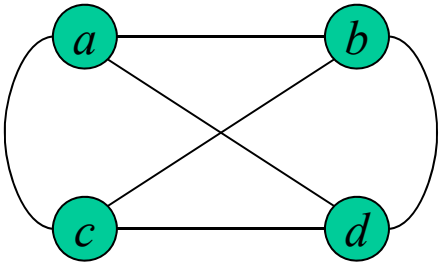
a 2-exchange procedure could replaces edges $\{i_u, i_{u+1}\}$ and $\{i_v, i_{v+1}\}$ with $\{i_u, i_v\}$ and $\{i_{u+1}, i_{v+1}\}$ and resulted in a new tour

$$\{i_1, i_2, \ldots, i_u, i_v, i_{v-1}, \ldots, i_{u+1}, i_{v+1}, \ldots, i_n\}$$

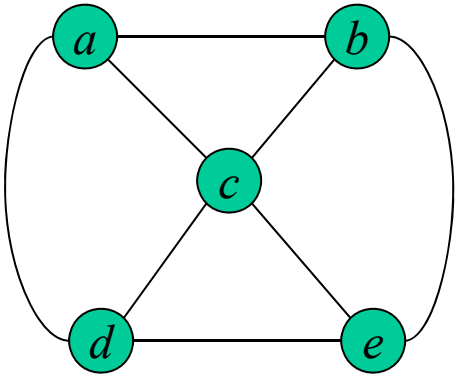
Example for 2-exchange



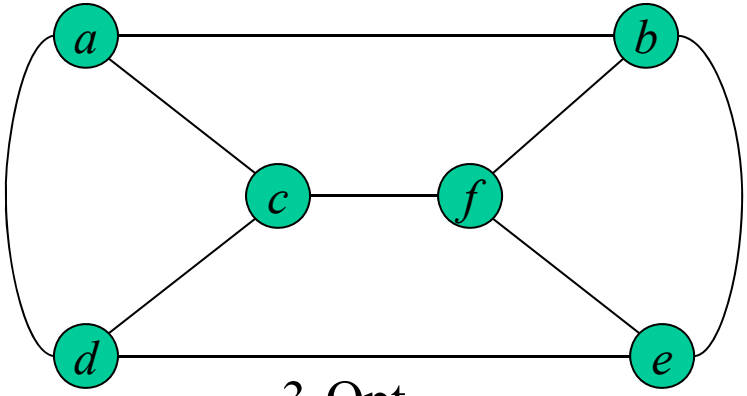
Improvement Heuristics



2-Opt



2H-Opt



3-Opt