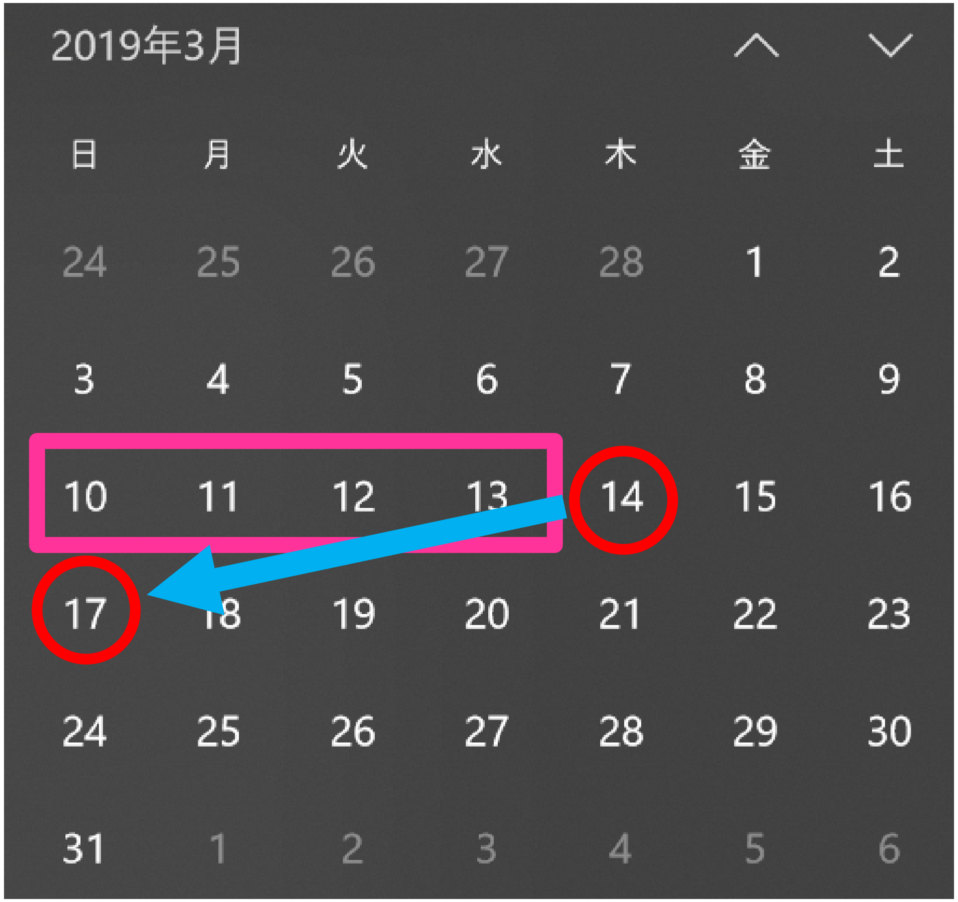
Schedule Change:
From March 14 to March 17 (Sunday)

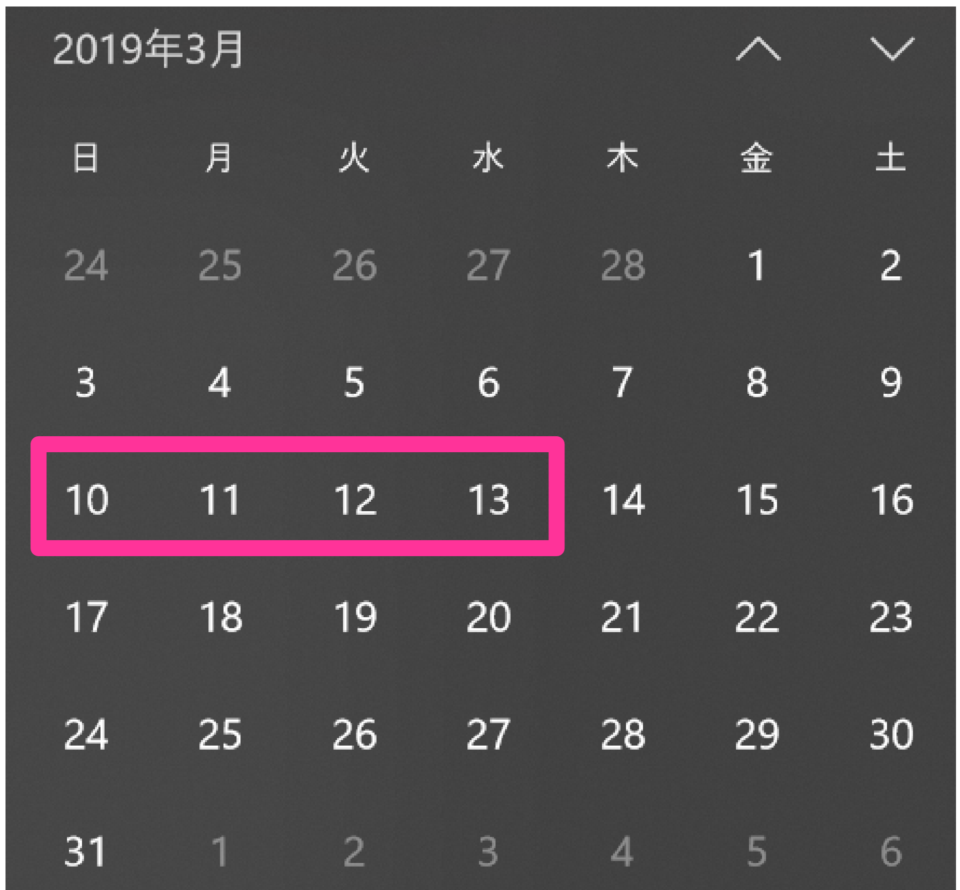


EMO-2019 Conference

10-13 March 2019

Kellogg Hotel and Convention Center

Michigan State University, East Lansing, Michigan, USA



EMO-2021: 28-31 MARCH 2021

Southern University of Science and Technology, Shenzhen, China

CALL FOR PAPERS

WEBSITE:
To be confirmed

ORGANIZERS:

General Chairs:
Hisao Ishibuchi
Qingfu Zhang

Program Chairs:
Hui Li
Handing Wang

Publication Chairs:
Aimin Zhou
Ke Li

Organizing chair:
Ran Cheng

About EMO:

EMO 2021 is the 11th Edition of International Conference Series on **Evolutionary Multi-Criterion Optimization (EMO)**, aiming to continue the success of previous EMO conferences. We will bring together both the EMO, Multiple Criteria Decision-Making (MCDM) communities, and other related fields and, moreover, focusing on solving real-world problems in government, business and industry.

Proceedings:

Full papers (12 pages) will be published by Springer Under LNCS series

Planned Sessions:

Keynotes, Tutorials, EMO Sessions, MCDM Sessions, Industry Sessions, Industry Booths for Software/Application Demonstration, Poster Session, Panel Session (to be finalized)

Best Paper Awards:

Best Student Paper Award and Best General Paper Award

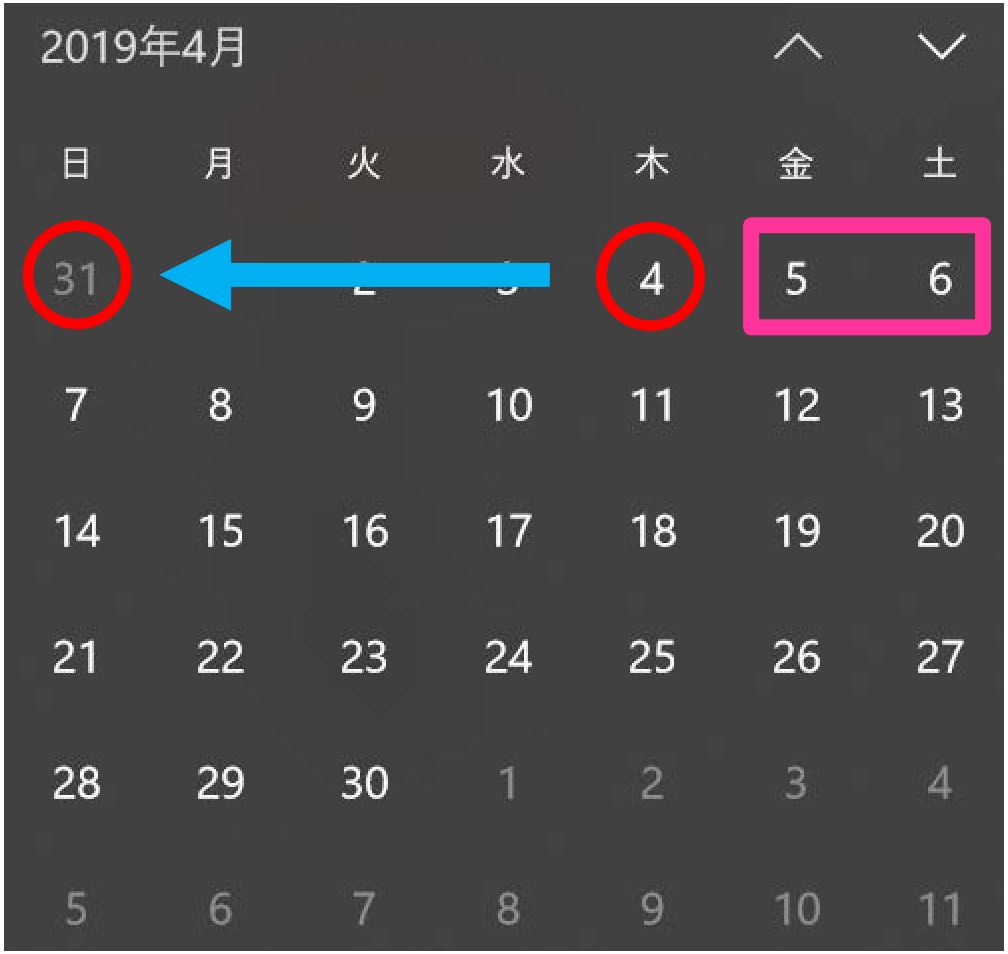
Topics of interests:

We aim to discuss all aspects of EMO development and deployment, including (but not limited to):

- EMO for handling of continuous, combinatorial and/or mixed-integer problems
- Constraint handling approaches
- Uncertainty and noise handling approaches, robust optimization
- Many-objective optimization
- Multiobjective swarm optimization

Schedule Change:

From April 4 to March 31 (Sunday)



IEEE Panel of Editors Meeting 2019

Dear Hisao

We invite you to attend the IEEE Panel of Technical Editors program on 5 & 6 April 2019, which will take place at the Renaissance Chicago Downtown Hotel, Chicago, IL USA.

We hope that you will join us this year to participate in discussions regarding our publications business, learn more about our services and products and the status of projects currently underway, exchange ideas and information, and interact with colleagues and staff.

Exercise TSP-0 (Presentation: Next Week)

Our TA will create a 100-city TSP problem. Using your own method (any method is OK), try to find the best solution. In your presentation file, please clearly explain your optimization algorithm. Please also clearly show the obtained tour and its total tour length.

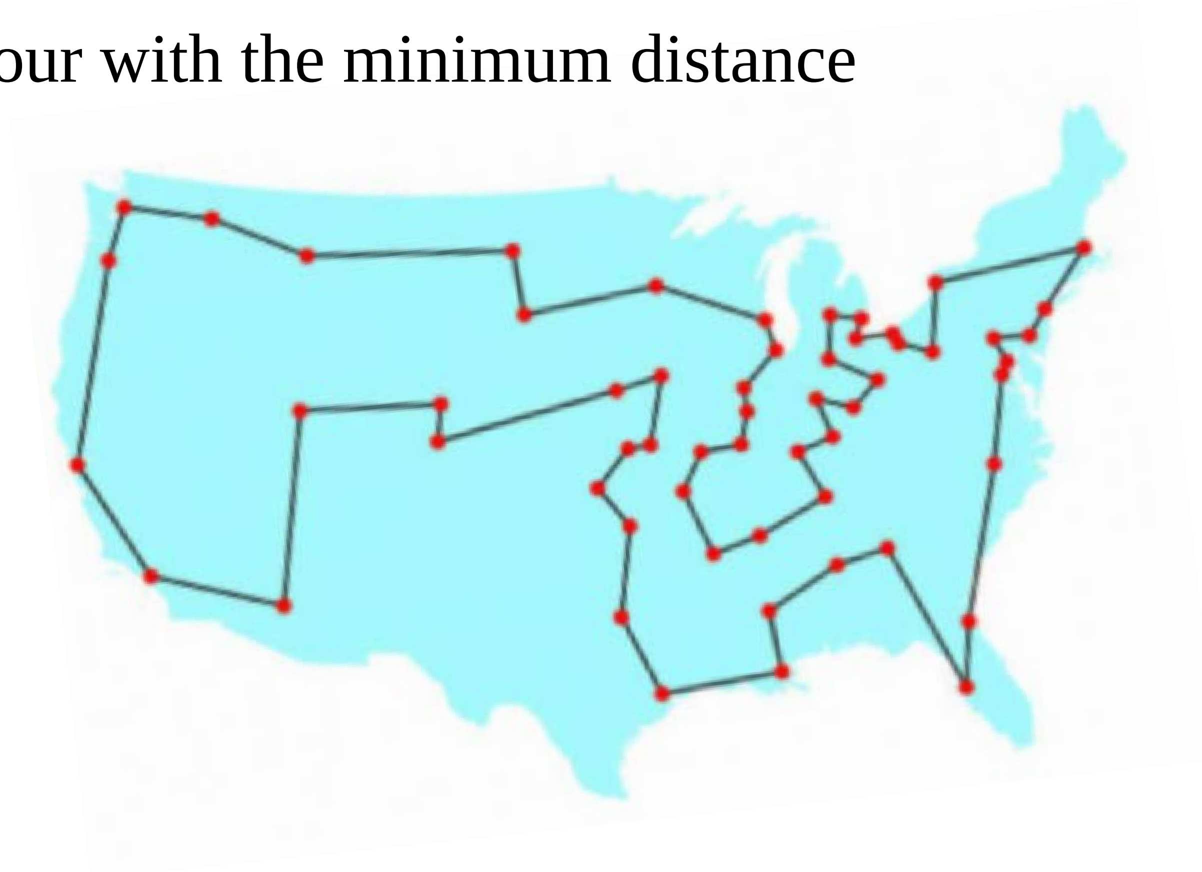
Travelling Salesman Problem (TSP)

Input: City set: n cities ($i = 1, 2, ..., n$)

Distance between each pair of cities (i, j): $d_{ij} = d_{ji}$

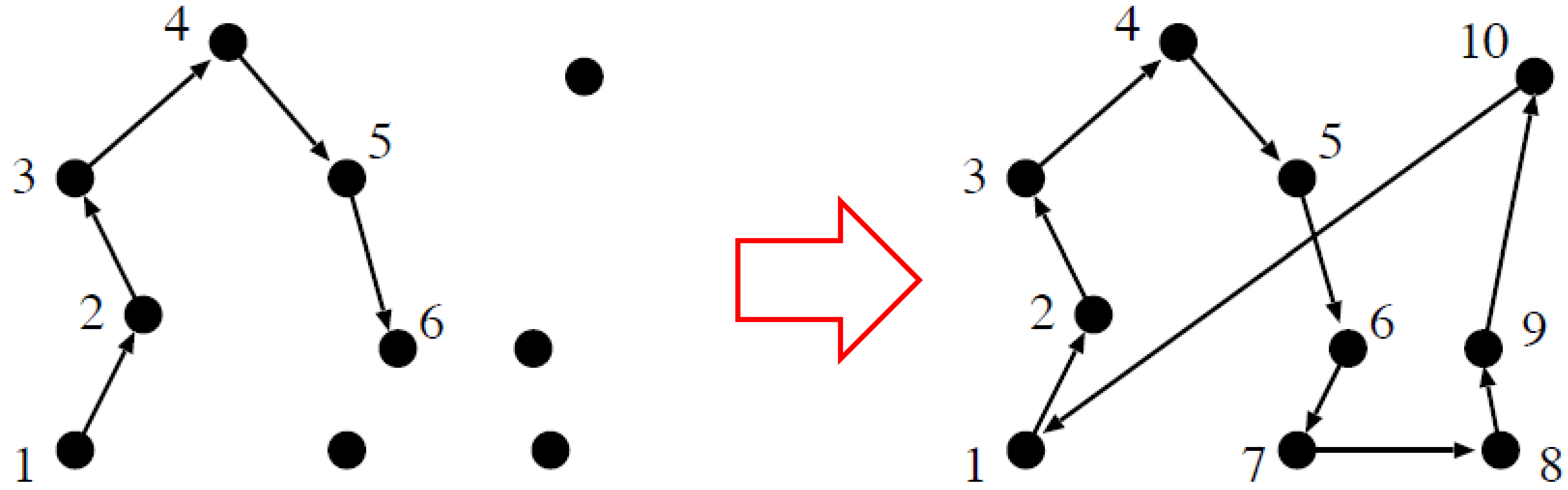
Objective: Minimization of a tour length starting from a city, visiting all cities and returning to the start city.

Output: Tour with the minimum distance

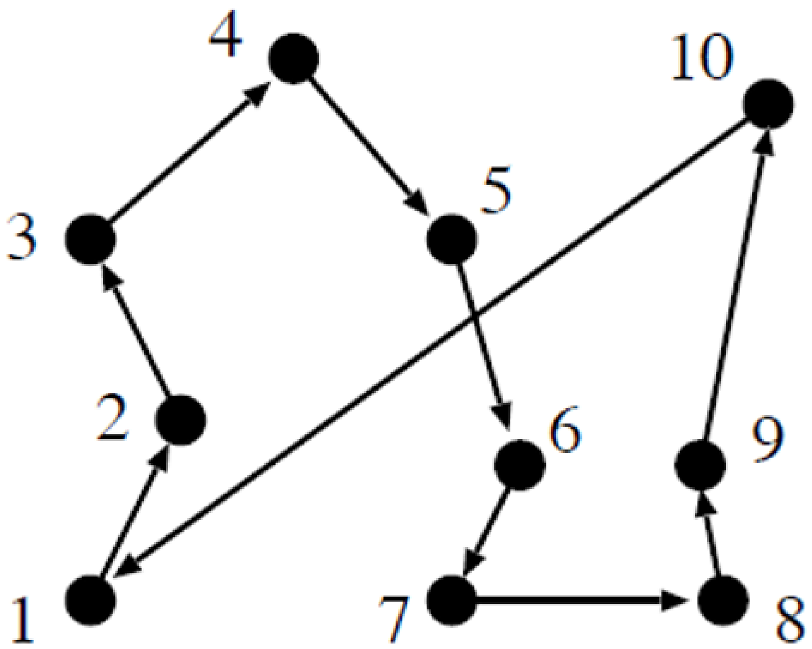


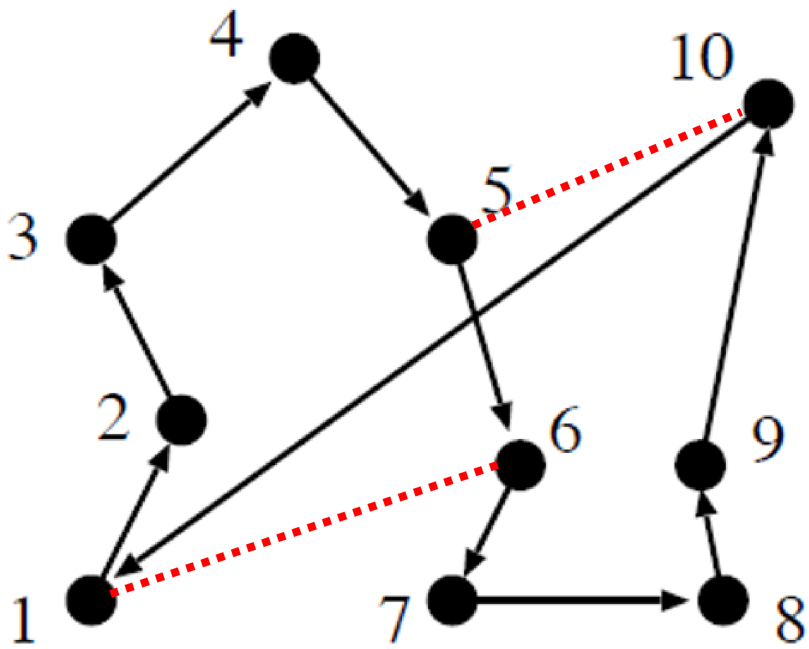
Nearest Neighbor Greedy Method

1. Arbitrarily select a starting city.
2. Move to the nearest city from the current city among the remaining cities.



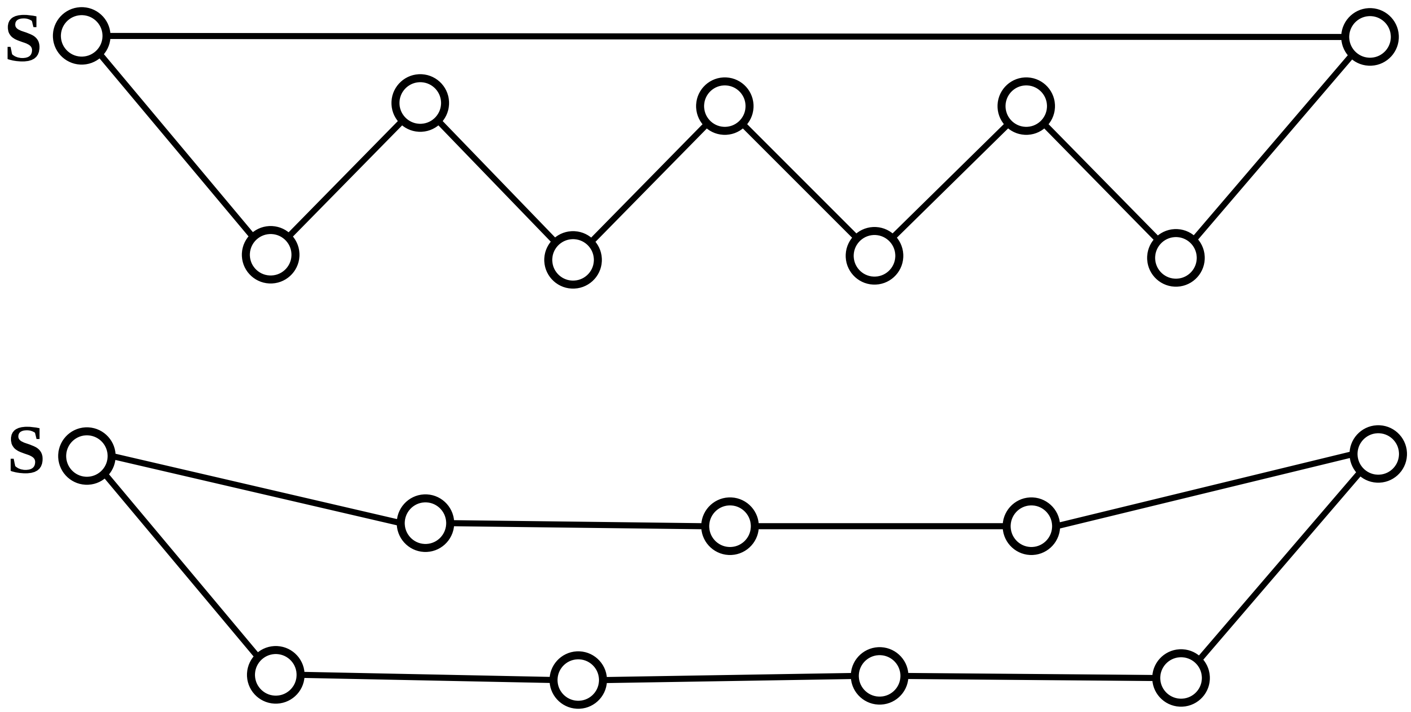
If two edges are crossing, the tour is not optimal





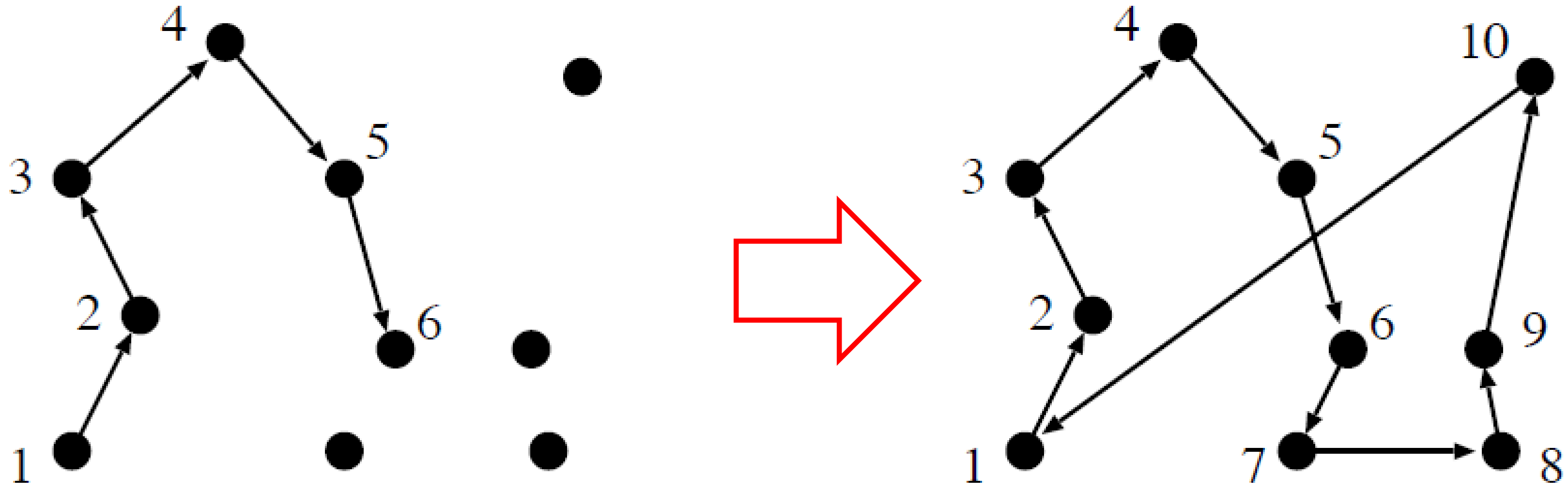
Exercise TSP-1

Create an example (problem instance) where the optimal solution is obtained by the nearest neighbor greedy method. Create another example where the obtained tour by the nearest neighbor greedy method is much longer than the optimal tour.

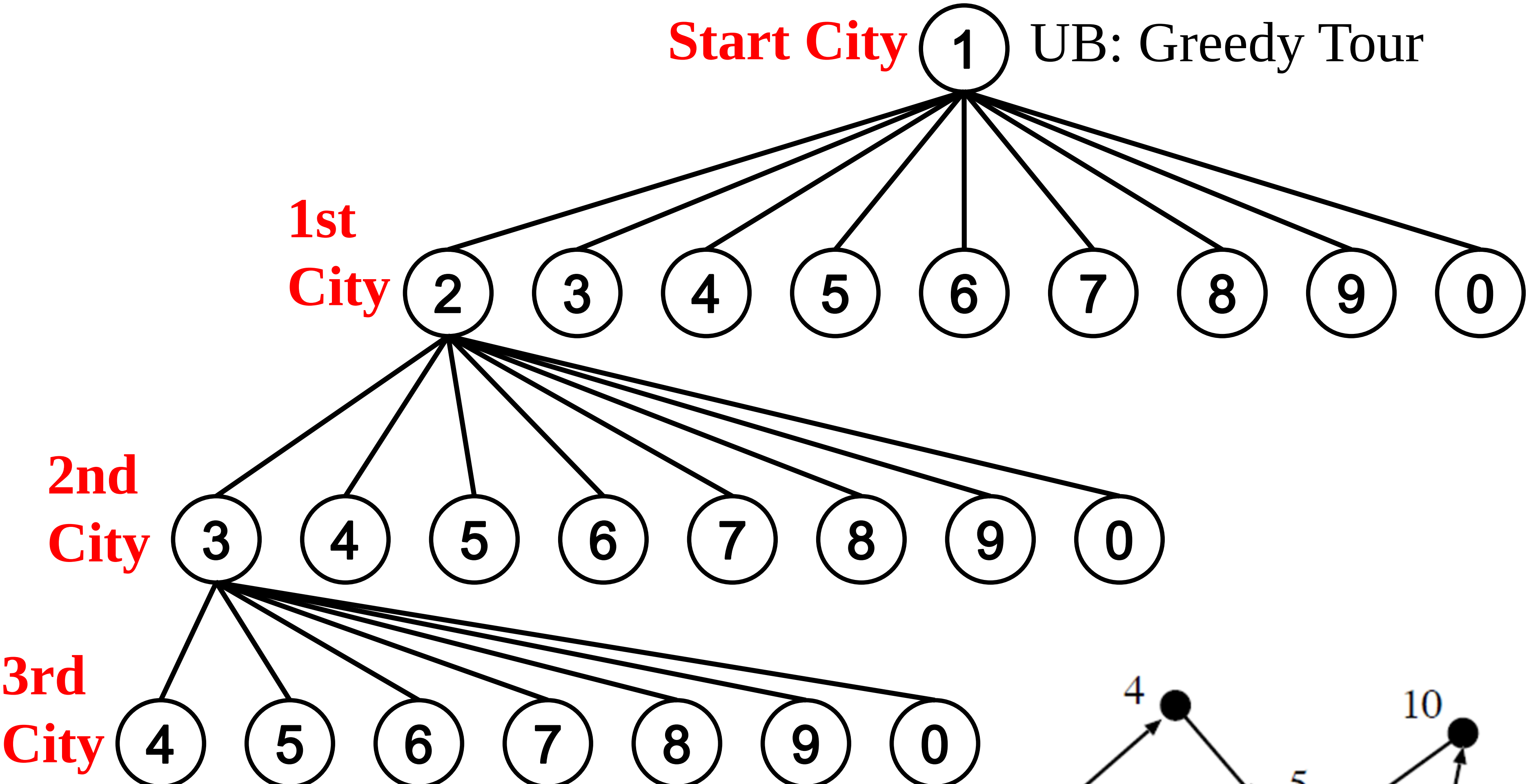


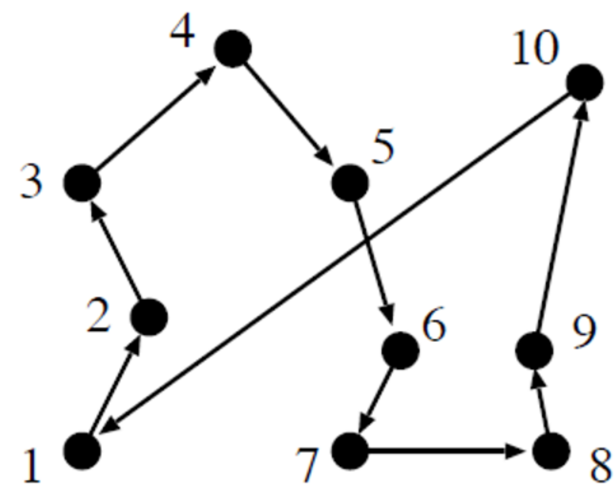
Exercise TSP-2

Examine the dependency of the obtained tour length (i.e., solution quality) on the choice of a start city in the nearest neighbor greedy method.



Branch and Bound



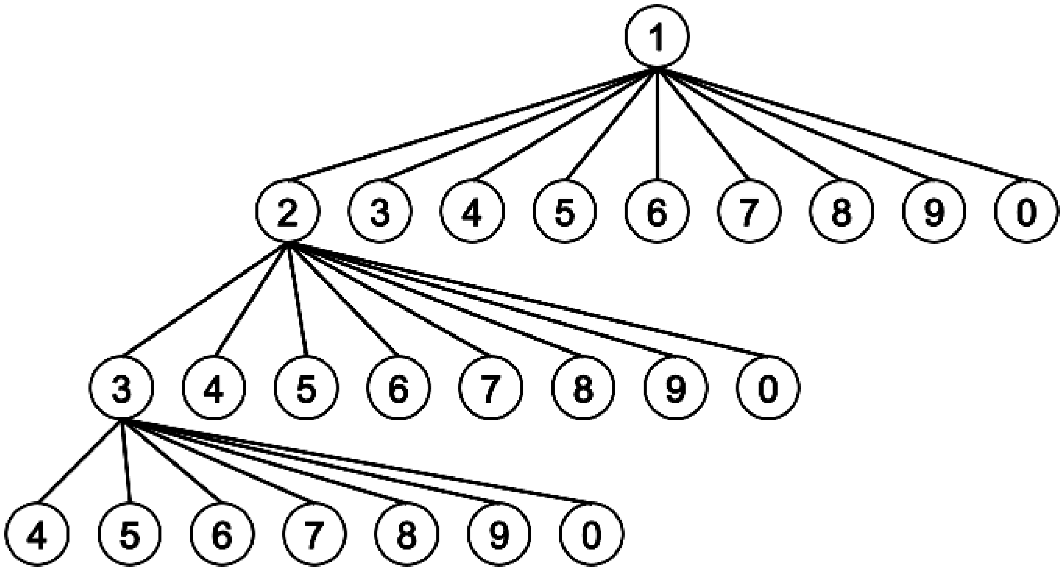


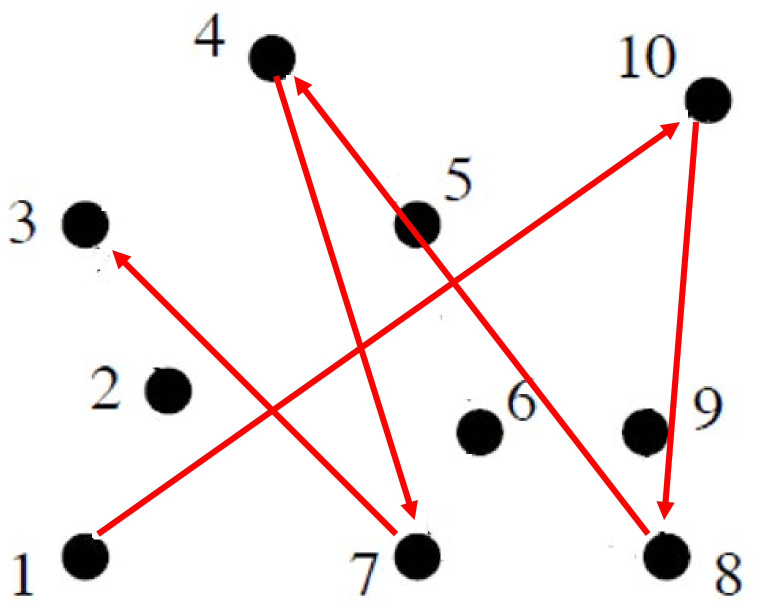
Implementation Issues.

(1) How to calculate the lower bound of each node.
Some tours may be easily identified as bad tours by calculating a part of them (e.g., tours starting with “1 10 8 4 7 3”).

(2) How to choose the node for the next branching.

(3) How to check the same tours for efficient search.
For example, “1 2 ... 9 10 1” is the same as “1 10 9 ... 2 1”.





Local Search

1. Generate an initial solution ***x***.
2. Iterate the following steps:

(i) Generate a neighbor solution ***y*** of the current solution ***x***.

(ii) If ***y*** is better than ***x***, replace the current solution ***x*** with ***y***.

(iii) If no better solution exists in the neighborhood of ***x***, terminate the execution of the algorithm.

Local search is simple. However, it has a lot of implementation issues such as

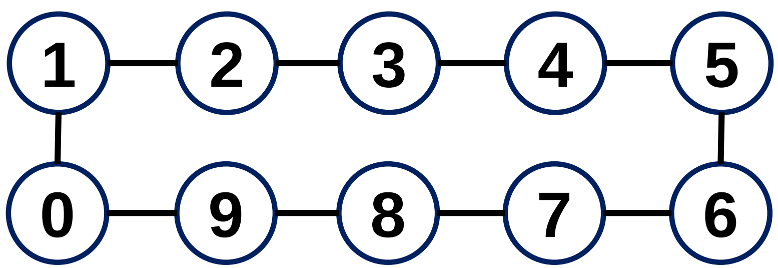
- How to generate an initial solution ***x*** ==> **Greedy Solution**
- How to specify the neighborhood of the current solution ***x***,
- How to specify the order of neighbors to be examined ==> **Random Order**

How to specify the neighborhood of the current solution *x*

(Specification of a neighborhood structure)

Current solution ***x*** (Order of *n* cities)

0	1	2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	0	1	2
9	8	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0	9	8

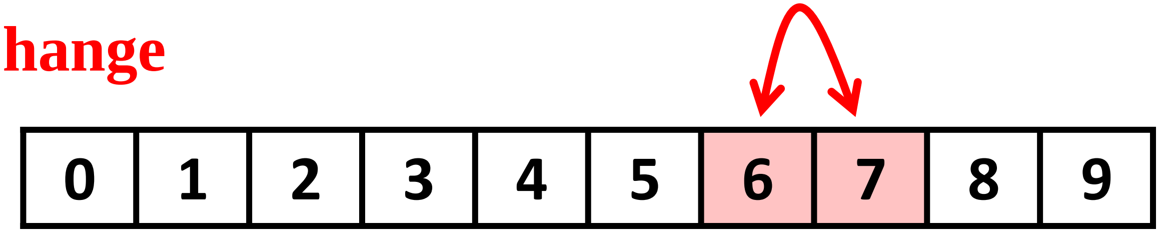


Number of different permutations: $n! = n \times (n - 1) \times (n - 2) \times \ldots$

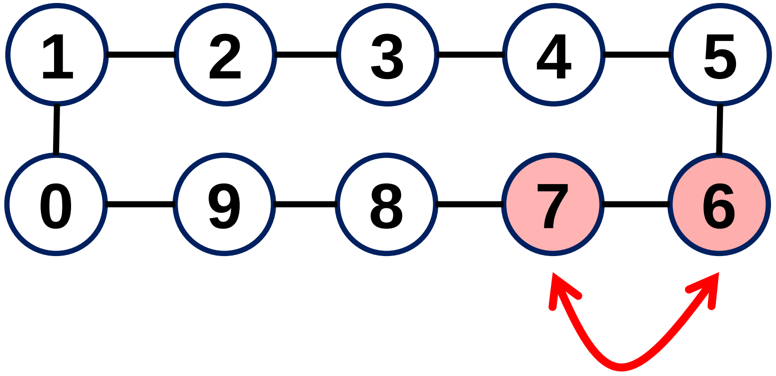
Number of different solutions:

Neighborhood Structures

Adjacent Two-City Change

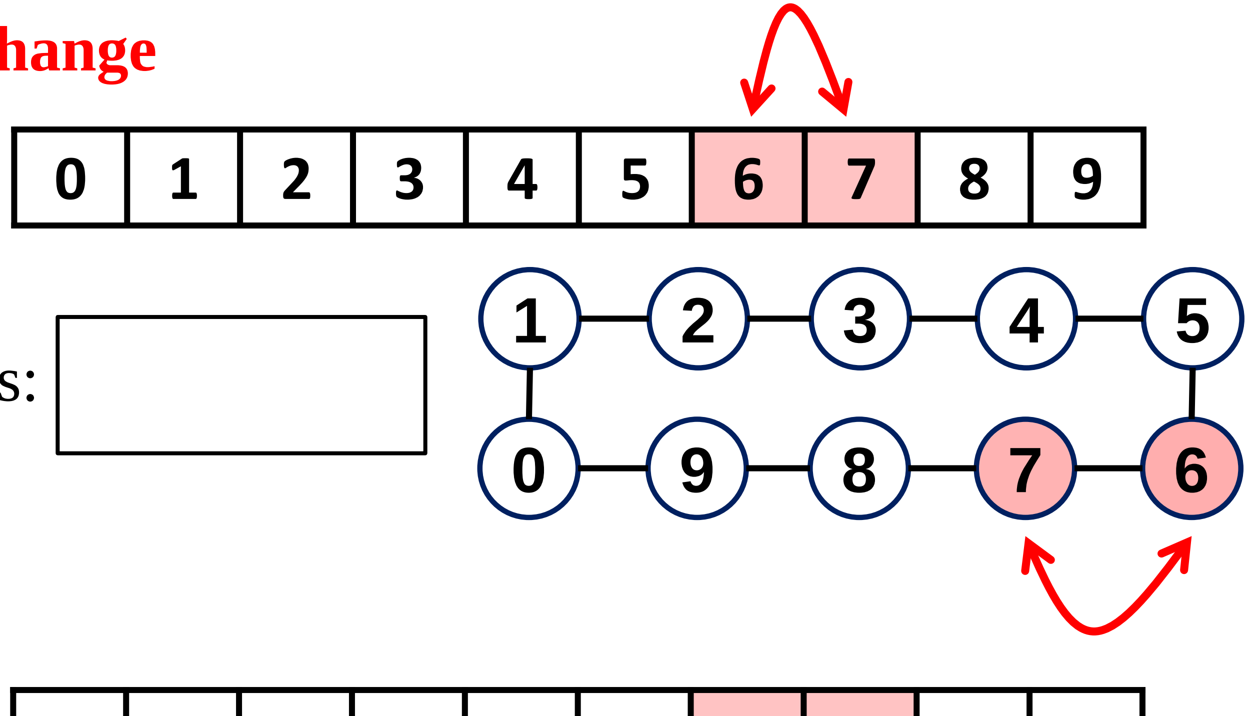


Number of neighbors:

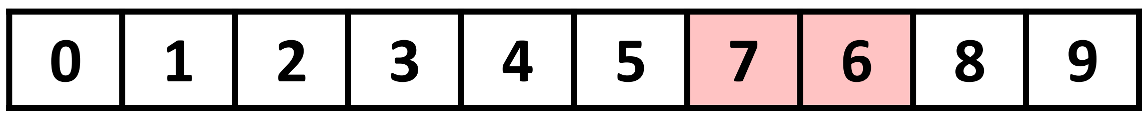


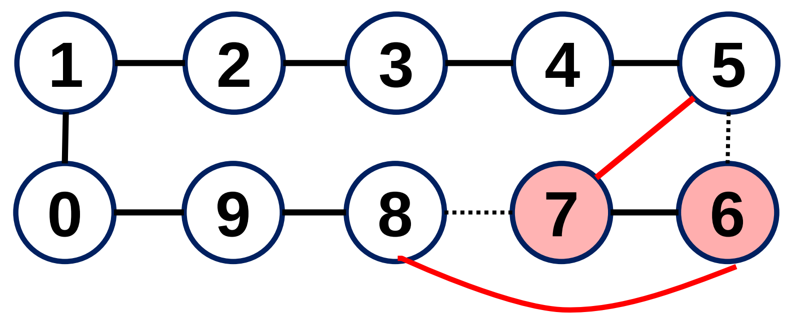
Neighborhood Structures

Adjacent Two-City Change



Number of neighbors:





Calculation of a new tour:

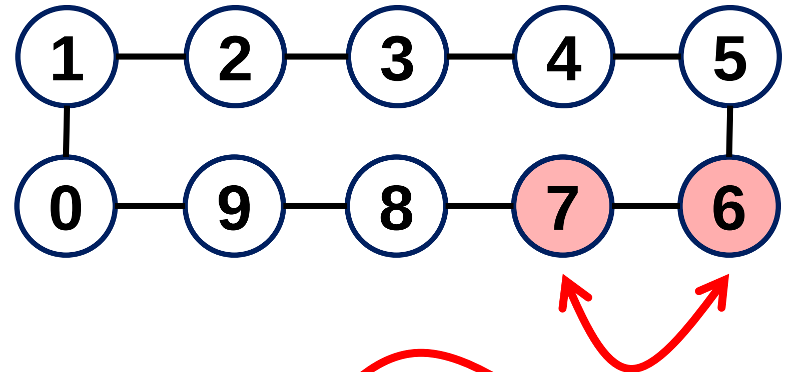
Addition of two edges and subtraction of two edges

Neighborhood Structures

Adjacent Two-City Change

0	1	2	3	4	5	6	7	8	9

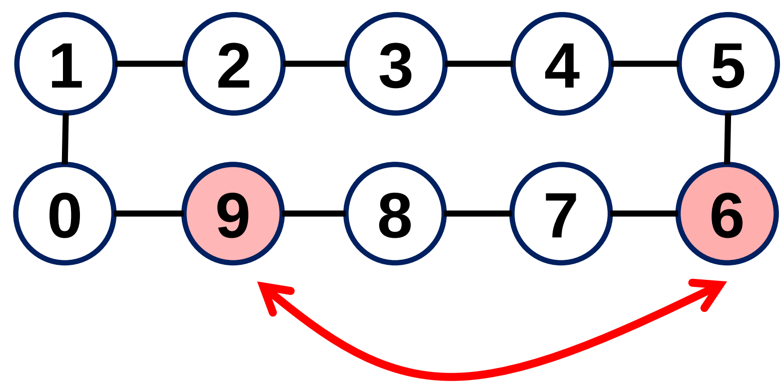
Number of neighbors:



Arbitrary Two-City Change

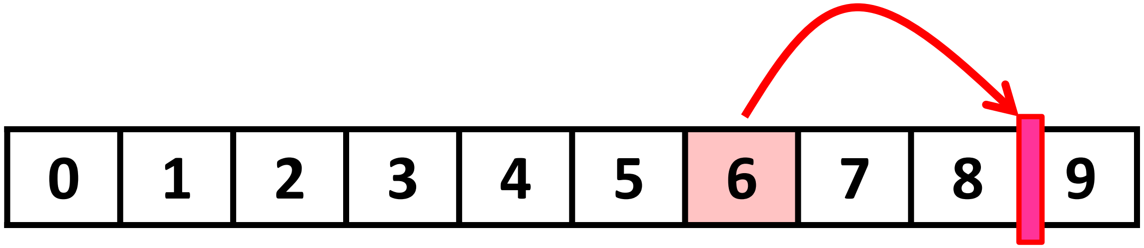
0	1	2	3	4	5	6	7	8	9

Number of neighbors:

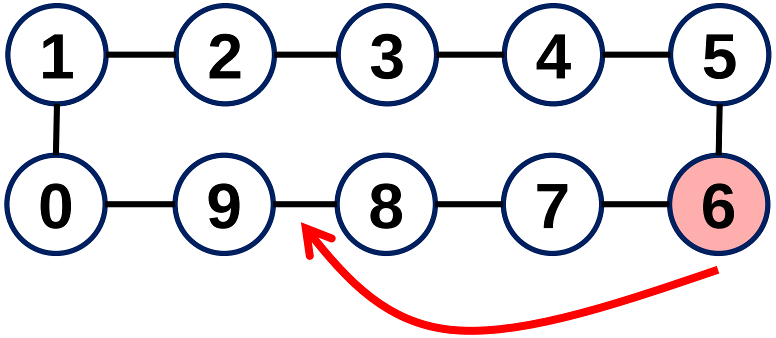


Neighborhood Structures

Insertion (Shift)

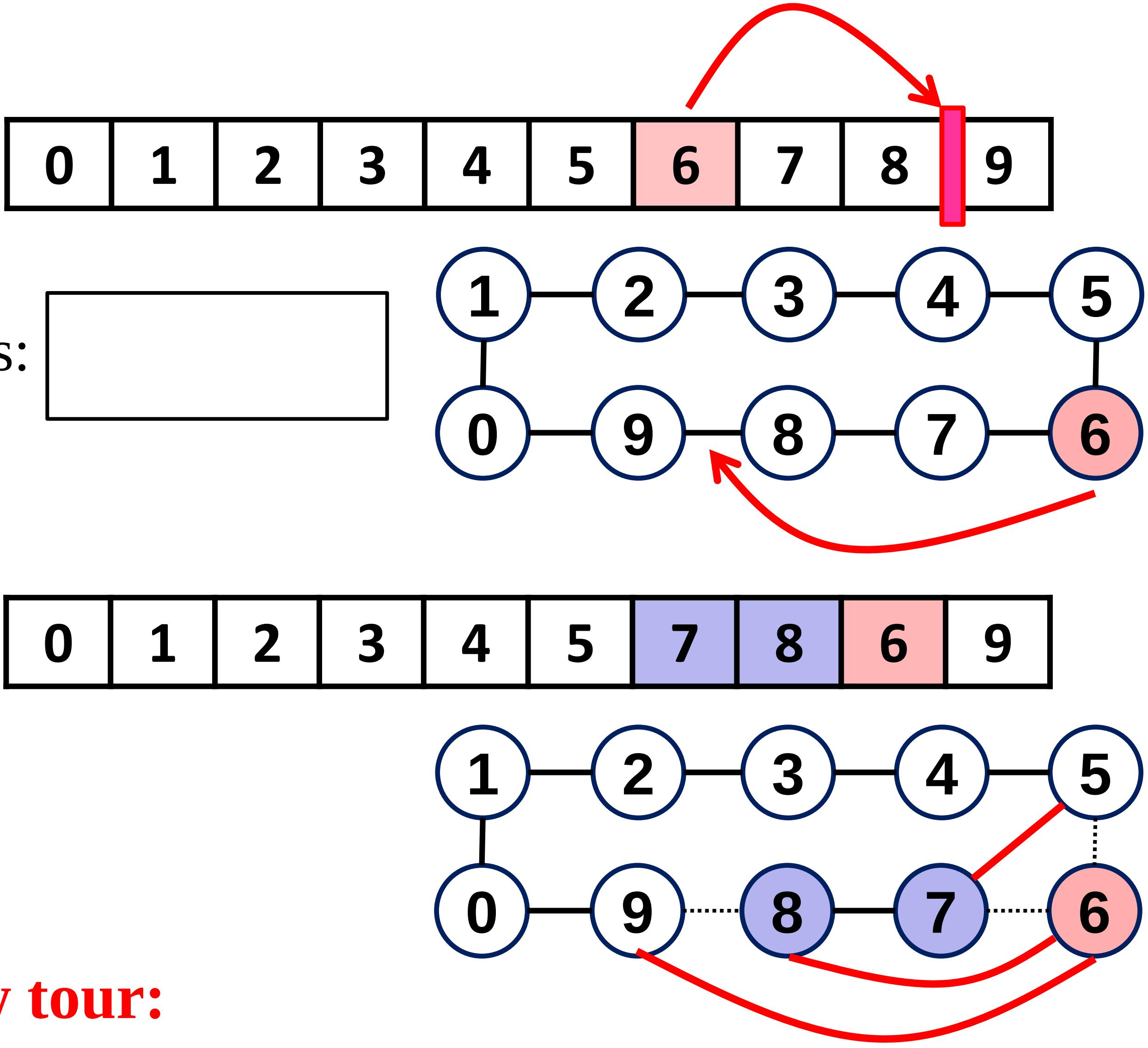


Number of neighbors:



Neighborhood Structures

Insertion (Shift)



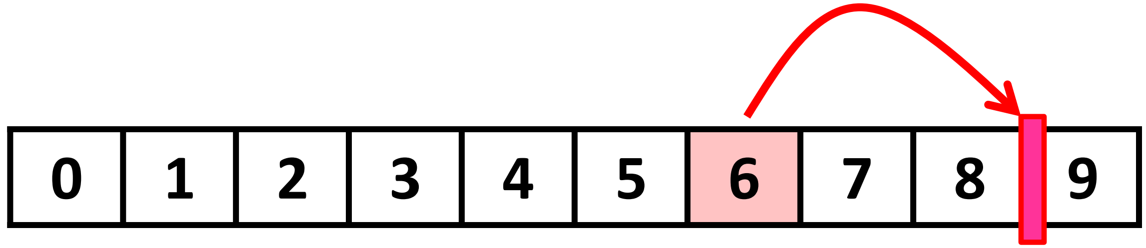
Number of neighbors:

Calculation of a new tour:

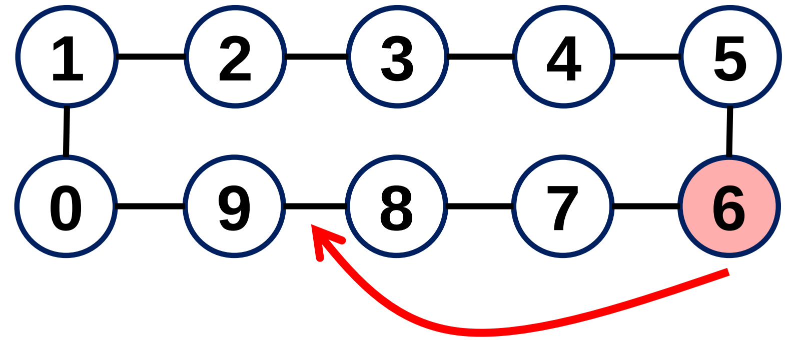
Addition of three edges and subtraction of three edges

Neighborhood Structures

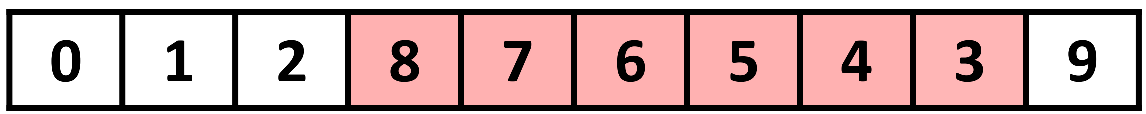
Insertion (Shift)



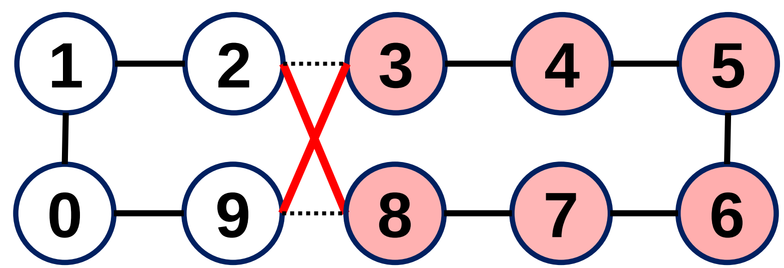
Number of neighbors:



Inversion (Arbitrary Two-Edge Change)



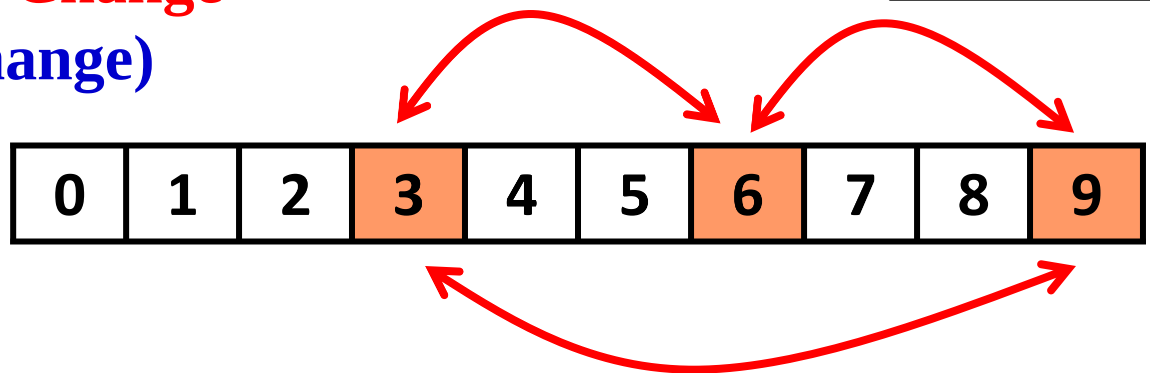
Number of neighbors:



Neighborhood Structures

Number of neighbors:

Arbitrary Three-City Change
(including two-city change)



0	1	2	3	4	5	6	7	8	9

0	1	2	3	4	5	9	7	8	6
0	1	2	6	4	5	3	7	8	9
0	1	2	6	4	5	9	7	8	3
0	1	2	9	4	5	3	7	8	6
0	1	2	9	4	5	6	7	8	3

Discussion

Which is the best and the worst neighborhood structure among the five choices for TSP?

Adjacent two-city change	0	1	2	3	4	5	7	6	8	9
Arbitrary two-city change	0	1	2	3	4	5	9	7	8	6
Insertion (Shift)	0	1	2	3	4	5	7	8	6	9
Inversion (Two-edge change)	0	1	2	8	7	6	5	4	3	9
Arbitrary three-city change	0	1	2	6	4	5	9	7	8	3