Advanced Algorithms

Reference

[1] Jon Kleinberg and Éva Tardos: Algorithm Design,
Pearson Education, Inc.
ISBN 0-321-29535-8

Evaluation

(1) Weekly Homework & Presentation: 40%

I will give a small homework every week. Your report should be in a form of a presentation file, which should be submitted before the midnight of Wednesday. Some students will be asked to present their reports during the morning or afternoon class. The evaluation is based on both the submitted files and the presentations.

(2) Final Week Presentation: 10%

In the last week, all students will give a short presentation (i.e., 2-4 minute presentation) on “Algorithms”.

(3) Final Examination: 50%

This is in the same style as the standard final examinations of other courses.

Two Goals: To learn the following

(1) How to examine the behavior of heuristic algorithms

I will show some simple heuristic algorithms. It is easy for all students to understand them (since they are very simple). The main focus of this course is how to examine the behavior of each algorithm. There are two directions. One is theoretical analysis about the worst case behavior. We will try to derive the upper/lower bounds on the quality of the obtained solution. The other is to numerically examine/explain the algorithm behavior using examples (i.e., test problems). Students will learn the importance of using good examples by creating simple examples by themselves.

(2) How to perform good presentations

Each student will give a presentation at least three times in the next 16 weeks. Student will learn a lot from other students' presentations as well as their own presentations.

Examined Algorithms

1. Greedy load balancing algorithm
2. Sorted greedy load balancing algorithm
3. Center selection algorithm
4. k-means clustering algorithms & fuzzy c-means clustering algorithm
5. Greedy set cover algorithm
6. Pricing method for vertex cover problem
7. LP (Linear Programming)
8. LP-based vertex cover method
9. LP-based generalized load balancing method
10. Greedy disjoint path algorithm (c = 1)
11. Greedy disjoint path algorithm (c > 1)
12. Greedy knapsack algorithm
13. Meta-heuristic algorithms for knapsack problem

Example of Theoretical Analysis

Next week using the greedy load balancing algorithm.

Example of Example-based Analysis

This week using TSP (Travelling Salesman Problem)

Example: Travelling Salesman Problem (TSP)

To find the shortest tour to visit all cities and return to the starting city.

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

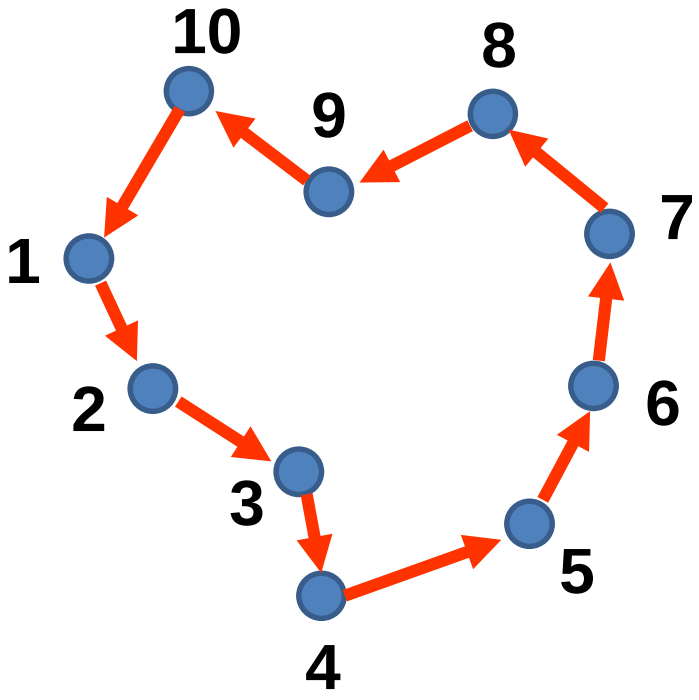
Distance between each pair of cities (i, j): $d_{ij} = d_{ji}$

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

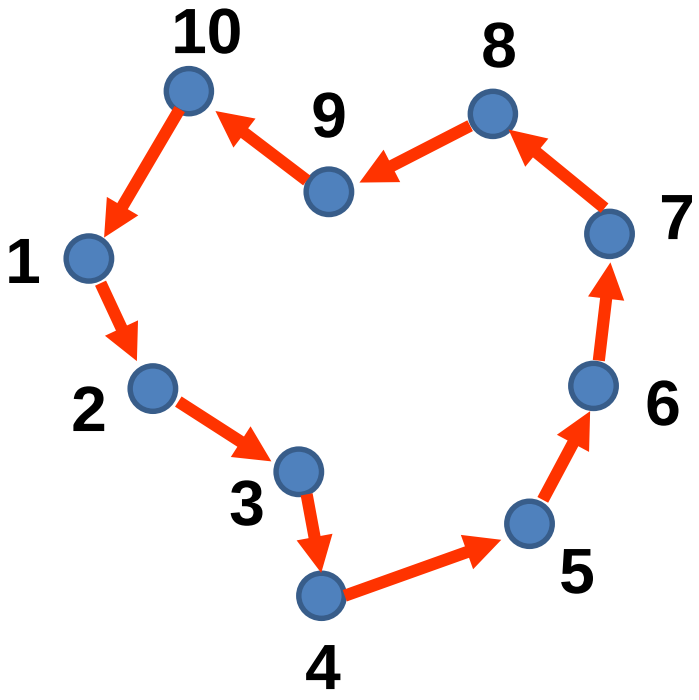
Nearest Neighbor Greedy Method

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



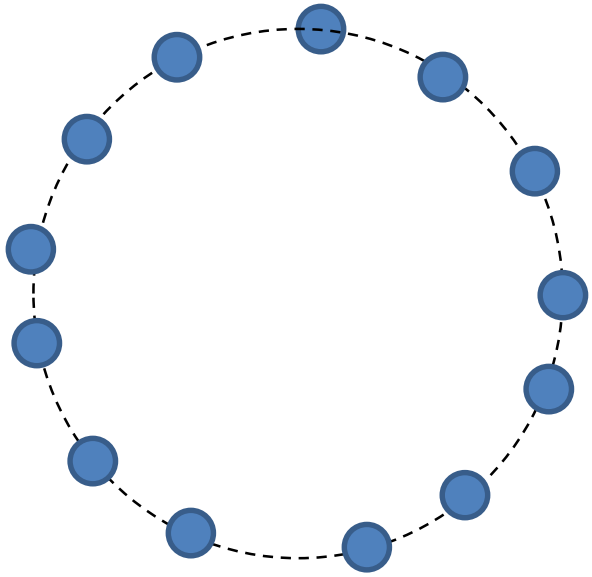
Use of Simple Examples

Examples are helpful to understand the greedy algorithm (and any algorithm). Create an example (problem instance) where the optimal solution is always obtained by the nearest neighbor greedy algorithm independent of the choice of a starting city.

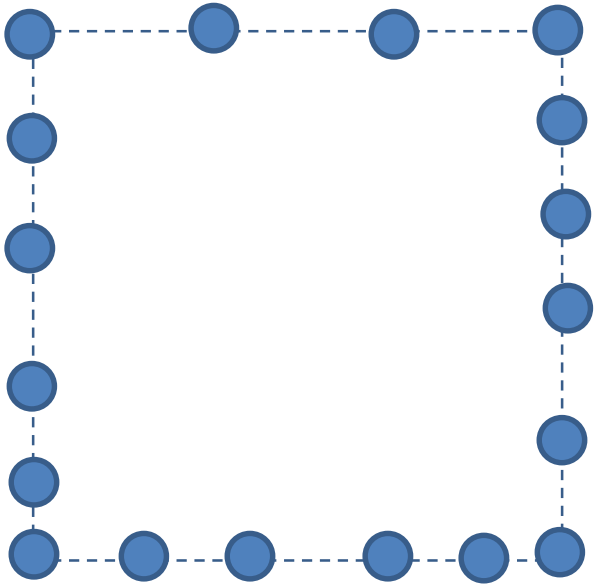


Examples of Example-Based Explanations

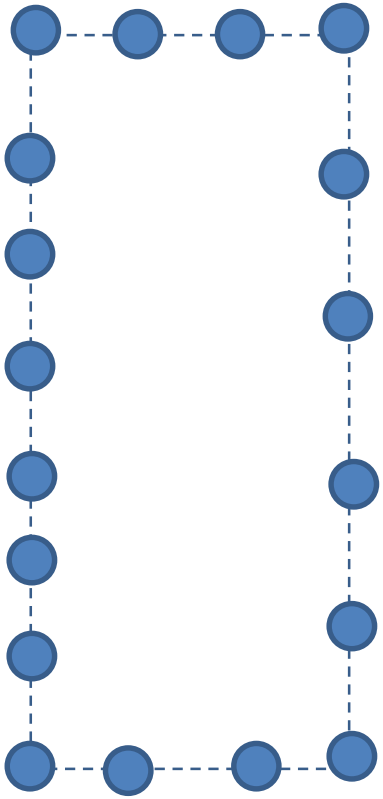
We can obtain the optimal solutions for various problems independent of the choice of the first city:



Example 1



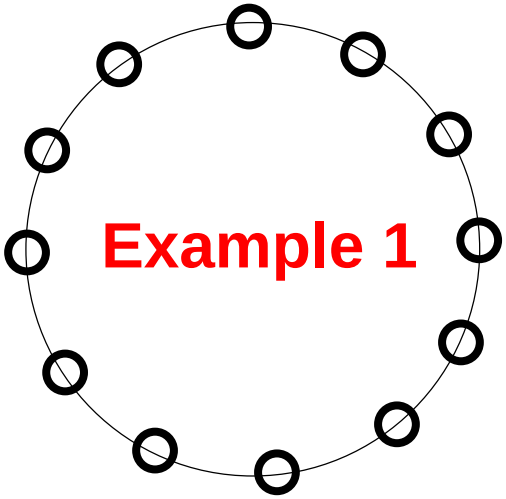
Example 2

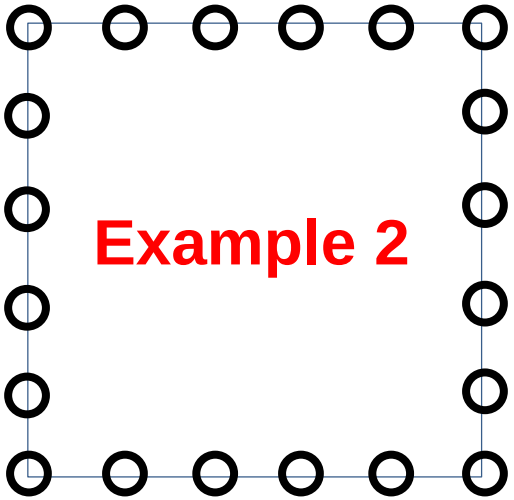


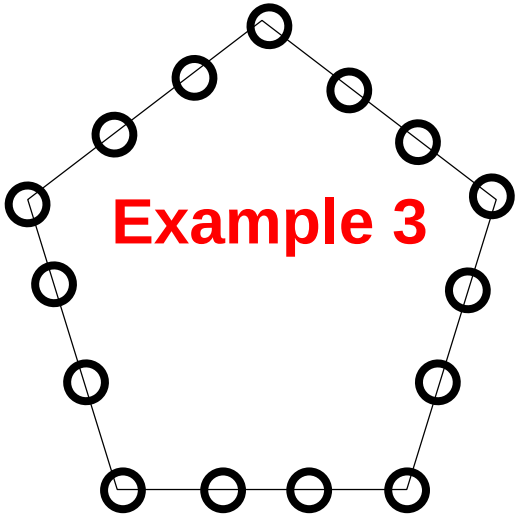
Example 3

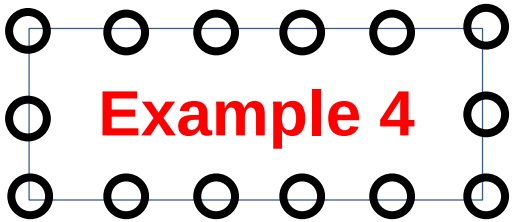
Examples of Example-Based Explanations

We can show that the optimal solution can be obtained for various problems independent of the choice of the first city (i.e., the choice of the first city is not important):

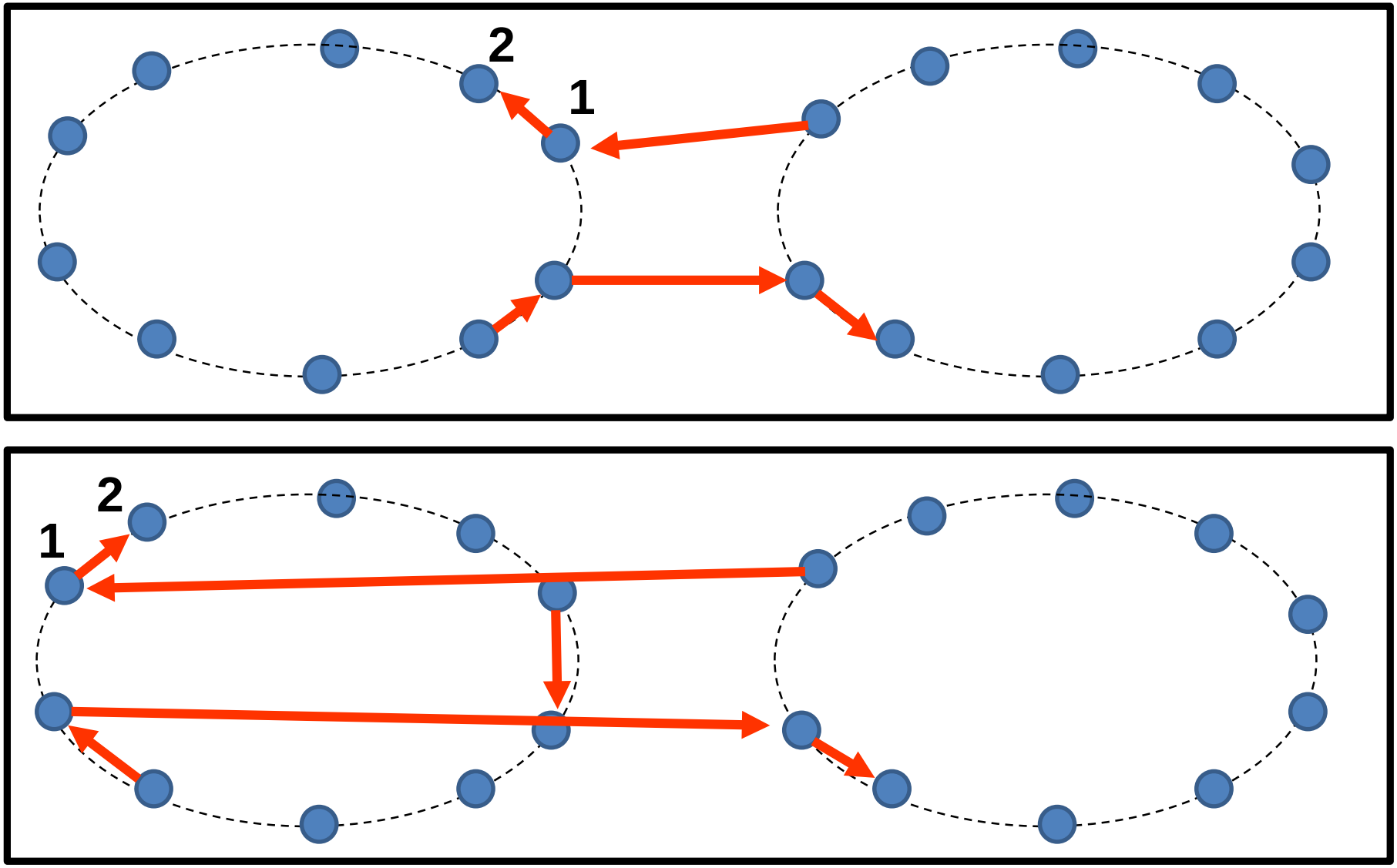




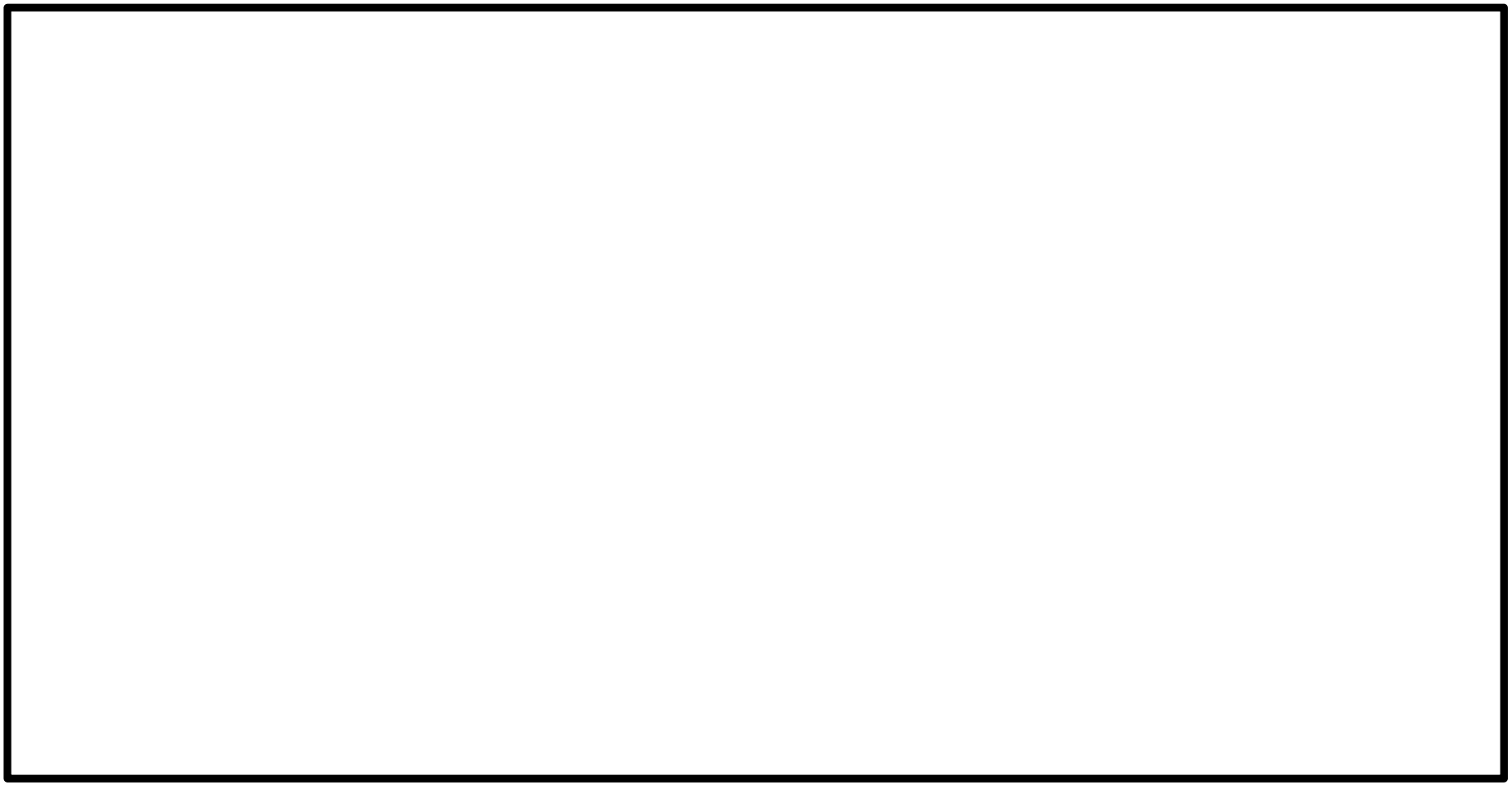




Using a different example, we can show that the choice of the first city is very important:



We may be able to show that good solutions cannot be obtained independent of the choice of the first solution (i.e., Bad solutions are always obtained independent of the choice of the first solutions).



Homework 1:

Create some examples (i.e., test problems) with 10 cities to clearly demonstrate the search behavior of the nearest neighbor greedy algorithm. The quality of the obtained solution (obtained tour) is evaluated by the following quality measure (in this formulation, the larger values show the worse solutions: 1 is the optimal solution).

$$\textbf{Solution Quality} = \frac{\textbf{Obtained Tour Length}}{\textbf{Optimal Tour Length}}$$

Choice of test problems is very important in algorithm design.

This can be applicable to almost all research fields.

e.g., Machine Learning