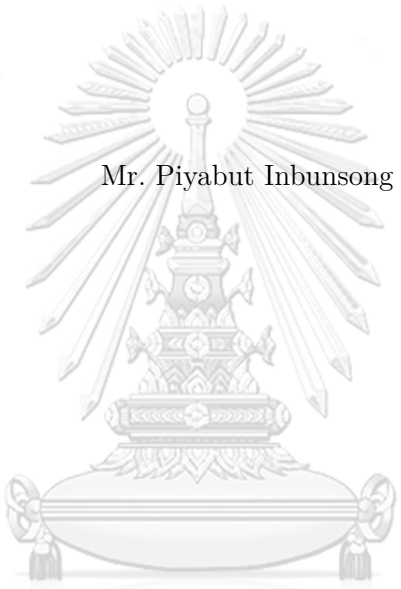
การระบุลักษณะเฉพาะที่เด่นที่สุดของคำตอบที่ดีสำหรับปัญหาการจัดเส้นทางเดินรถที่มีความจุจำกัดแบบไม่ยูคลิดโดยใช้ตัวแบบการเรียนรู้เชิงสถิติ

นายปิยะบุตร อินบุญส่ง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

IDENTIFYING THE MOST DISTINCTIVE CHARACTERISTICS OF A GOOD SOLUTION FOR NON-EUCLIDEAN CVRP USING STATISTICAL LEARNING MODEL

Mr. Piyabut Inbunsong



A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Applied Mathematics and Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2022

Thesis Title	IDENTIFYING THE MOST DISTINCTIVE CHARACTERISTICS OF A GOOD SOLUTION FOR NON-EUCLIDEAN CVRP USING STATISTICAL LEARNING MODEL
By	Mr. Piyabut Inbunsong
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Assistant Professor Boonyarit Intiyot, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

..... Dean of the Faculty of Science
(Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

..... Chairman
(Associate Professor Dr. Krung Sinapiromsaran, Ph.D.)

..... Thesis Advisor
(Assistant Professor Boonyarit Intiyot, Ph.D.)

..... Examiner
(Associate Professor Dr. Phantipa Thipwiwatpotjana, Ph.D.)

..... External Examiner
(Associate Professor Dr. Chawalit Jeenanunta, Ph.D.)

ปิยะบุตร อินบุญส่ง : การระบุลักษณะเฉพาะที่เด่นที่สุดของคำตอบที่ดีสำหรับปัญหาการจัดเส้นทางเดินรถที่มีความจุจำกัดแบบไม่ยูคลิดโดยใช้ตัวแบบการเรียนรู้เชิงสถิติ. (IDENTIFYING THE MOST DISTINCTIVE CHARACTERISTICS OF A GOOD SOLUTION FOR NON-EUCLIDEAN CVRP USING STATISTICAL LEARNING MODEL) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.บุญฤทธิ์ อินทิยศ, 69 หน้า.

ปัญหาการจัดเส้นทางเดินรถที่มีความจุจำกัดหรือ CVRP เป็นปัญหาเอ็นพีแบบยากของการหาค่าเหมาะที่สุดเชิงการจัดที่เป็นที่รู้จักกันดี ดังนั้นฮิวริสติกจึงเป็นวิธีการที่นิยมใช้ในการหาคำตอบที่ดี อัลกอริทึมดังกล่าวจะทำงานได้ดีขึ้นถ้ารู้จักลักษณะเฉพาะของคำตอบที่ดีของปัญหา ทั้งนี้มีงานวิจัยที่ศึกษาลักษณะเฉพาะของคำตอบของปัญหา CVRP แบบยูคลิด และภายหลังได้นำความรู้ที่ได้ไปประยุกต์ใช้กับเมธาฮิวริสติก งานวิจัยดังกล่าวได้กลายเป็นแรงบันดาลใจของเราในการศึกษาลักษณะเฉพาะของคำตอบของปัญหา CVRP แบบไม่ยูคลิด ดังนั้นเพื่อให้บรรลุเป้าหมายดังกล่าว เราพิจารณาคำตอบของปัญหา CVRP แบบไม่ยูคลิดในปริภูมิใหม่แบบยูคลิดที่มีมิติเดียวกันหรือสูงกว่า โดยใช้ multi-dimensional scaling ที่ซึ่งลักษณะเฉพาะของคำตอบสามารถถูกนิยามภายใต้คุณสมบัติของยูคลิด นอกจากนี้ตัวแบบการเรียนรู้เชิงสถิติได้ถูกใช้เพื่อระบุลักษณะเฉพาะที่โดดเด่นที่สุด ที่ให้ความแม่นยำสูงสุดจากการทำนายการเป็นคำตอบที่ดีในปริภูมิใหม่นี้ ยิ่งไปกว่านั้นยังมีการระบุกฎการตัดสินใจสำหรับใช้ตีความลักษณะเฉพาะของคำตอบที่ดีและไม่ดีสำหรับปัญหา CVRP แบบไม่ยูคลิด

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต .....
		ลายมือชื่อ อ.ที่ปรึกษาหลัก .....
สาขาวิชา	คณิตศาสตร์ประยุกต์และวิทยาการคณนา	ลายมือชื่อ อ.ที่ปรึกษาร่วม .....
ปีการศึกษา	2565	

6270066923 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
KEYWORDS : VEHICLE ROUTING PROBLEM/ STATISTICAL LEARNING MODEL / MULTI-DIMENSIONAL SCALING

PIYABUT INBUNSONG : IDENTIFYING THE MOST DISTINCTIVE CHARACTERISTICS OF A GOOD SOLUTION FOR NON-EUCLIDEAN CVRP USING STATISTICAL LEARNING MODEL. ADVISOR : ASST. PROF. BOONYARIT INTIYOT, Ph.D., 69 pp.

A capacitated vehicle routing problem (CVRP) is a well-known NP-hard combinatorial optimization. Therefore, heuristics are the common methods used to search for a good solution. The algorithms will perform better if characteristics of good solutions of the problem are known. There was a research study in the characteristics of Euclidean CVRP solutions and the knowledge was later applied in a metaheuristic. That study becomes our motivation to study the characteristics of non-Euclidean CVRP solutions. To that end, we considered the solutions of non-Euclidean CVRP in the new Euclidean space with the same or higher dimensions using multi-dimensional scaling in which the characteristics of the solutions can be defined under Euclidean properties. In addition, the statistical learning models were employed to identify the most distinctive characteristic which yields the highest accuracy of prediction of a good solution in the new space. Moreover, decision rules were also determined for interpreting characteristics of good and bad solutions of non-Euclidean CVRP.

Department	: Mathematics and Computer Science	Student's Signature	.....
		Advisor's Signature	.....
Field of Study	: Applied Mathematics and Computational Science	Co-advisor's Signature	.....
Academic Year	: 2022		

ACKNOWLEDGEMENTS

วิทยานิพนธ์นี้สำเร็จได้ด้วยดีเพราะได้รับคำแนะนำในด้านความรู้ เทคนิค วิธีการ และความเอาใจใส่ที่ดียิ่งจากอาจารย์ ผศ. ดร. บุญฤทธิ์ อินทิยศ ผู้เป็นอาจารย์ที่ปรึกษา ทำให้ผ่านพ้นช่วงเวลาที่เจอปัญหาและอุปสรรค และไม่ใช่เพียงด้านวิชาการเพียงเท่านั้น ผมขอขอบคุณที่ให้คำปรึกษาเรื่องแนวทางการใช้ชีวิตด้วย สิ่งเหล่านี้เป็นประโยชน์มาก และทำให้งานวิทยานิพนธ์นี้ออกมาเสร็จสมบูรณ์ ผมขอขอบพระคุณอาจารย์เป็นอย่างสูงครับ และผมขอขอบพระคุณอาจารย์ภาควิชาคณิตศาสตร์ประยุกต์และวิทยาการทุกท่านที่ประสิทธิ์ประสาทวิชา ให้ได้รับความรู้ ความเข้าใจและสามารถนำไปประยุกต์ใช้ในอนาคตได้

ขอบคุณพี่ตี๋ อำพล ดวงแป้น ที่ให้คำปรึกษาในเรื่องของประสบการณ์ชีวิต และธุรการต่าง ๆในมหาวิทยาลัย รวมถึงทุกช่วงเวลาที่ได้ทำกิจกรรมร่วมกัน เป็นช่วงเวลาที่ดีมาก และขอบคุณ หลี กร เจ เฟิร์ส หมิว แพรว เปิ้ล และพี่ชานนเพื่อน ๆในสาขาวิชาที่คอยแนะนำสิ่งที่เป็นประโยชน์ และอยู่ด้วยกันเสมอมา รวมถึงรุ่นพี่ในสาขาวิชาคนอื่น ๆด้วย

ขอบพระคุณ คุณพ่อและคุณแม่ รวมถึงญาติพี่น้อง ที่คอยให้กำลังใจ เป็นแรงผลักดันและเบื้องหลังความสำเร็จ

ขอบพระคุณทุนพสวท.สำหรับทุนค่าใช้จ่ายเล่าเรียนเพื่อให้มาถึงจุดนี้

ผู้วิจัยมีความซาบซึ้งในความกรุณาของทุกท่านที่ได้กล่าวถึงซึ่งได้มีส่วนในการช่วยเหลือและให้กำลังใจเสมอมา จึงขอขอบพระคุณด้วยความจริงใจ

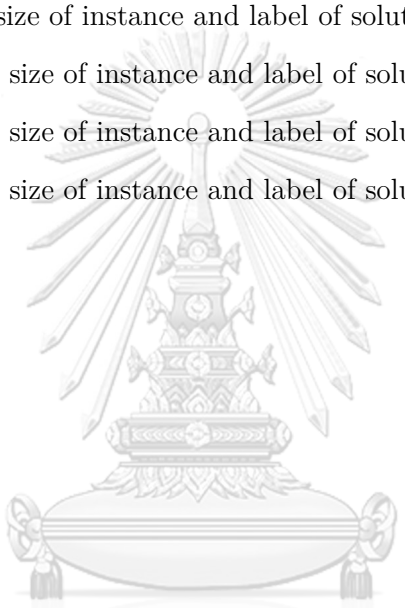
CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page



CHAPTER I

INTRODUCTION

This thesis presents a research study on a real-world-based Capacitated Vehicle Routing Problem (CVRP). The CVRP is a generalization of the Traveling Salesman Problem(TSP) which attempts to find optimal routes for a fleet of vehicles that minimize the total cost of delivering products to customers under the condition that is, every vehicle has limited capacity. Since TSP is NP-hard, so is CVRP, and this implies that the problem takes impracticable time to solve for an optimal solution in a large-sized problem. Therefore, solving for an exact solution is not satisfying for running a business that needs to complete a daily plan of routing on time. Then, (meta)heuristic algorithms are employed for handling this kind of problem. The algorithms are very fast even in large-sized problems. However, designing them to be efficient in terms of speed and quality is difficult. There are many research studies that proposed heuristic algorithms for CVRP and many of them can obtain the remarkable solution of benchmark problems. However, there is scarce research that deeply studies why heuristics work well. Recently, Arnold and Sörensen [1] presented problem-specific knowledge for heuristics in CVRP. They defined the structural characteristics of CVRP solutions. The characteristics, then, are adapted in the guiding part of a metaheuristic algorithm [2] which can reach a near-optimal solution more quickly than a metaheuristic using only cost to lead the search. Although the characteristics are practical, it needs to work in Euclidean space which is a necessary condition. However, this is not acceptable in a real-world problem because the distance is rarely Euclidean. Therefore, we are interested in extending Arnold and Sörensen's ideas about the characteristics of CVRP solutions in the scope of the non-Euclidean CVRP. In non-Euclidean space, we cannot determine the characteristics following their definitions. Therefore, we decide to consider the problem in a new abstract space in which the components of non-Euclidean CVRP such as the coordinate of the nodes are treated as if they have Euclidean properties. To that end, multi-dimensional scaling

is our answer to determine approximated locations of customer nodes in the abstract space we mentioned. This will allow us to extend Arnold and Sörensen's ideas to non-Euclidean CVRP solutions. Finally, the most distinctive characteristic is identified using a statistical learning framework. This also returns the specific knowledge related to the quality of non-Euclidean CVRP solutions.

The remainder of this thesis is organized as follows. In Chapter 2, we introduce background knowledge and related works. Chapter 3 explains the methodology of the whole study, and then the result will be shown in Chapter 4 and reveal the most distinctive characteristics of the non-Euclidean CVRP solution. Next, we discuss our results and conclude the study in Chapter 5.

CHAPTER II

BACKGROUND KNOWLEDGE

In this chapter, the essential definitions and theories are described. The chapter begins with an explanation of the CVRP and the algorithms to solve such a problem. Then the main techniques and methods we use in the study come next.

2.1 Capacitated Vehicle Routing Problem

The family of vehicle routing problems (VRPs) is combinatorial optimization problems which generally have the same purpose [3],[4]. Given a fleet of vehicles and transportation requirements, the task is to determine a set of vehicle routes that suit transportation requests at the minimum cost. The CVRP is the simplest and the most studied research topic in the family of VRPs. The problem description is given below.

Given a set of capacitated vehicles and a set of clients with the number of orders and their locations including a depot, each client must receive his order from the depot, which is delivered by a vehicle. Each vehicle must start from the depot to serve several clients and return to the depot after completing the delivery. Each client must be done exactly once i.e. each order cannot be split. The products that each vehicle carries must not exceed its capacity. Then, the task is to find optimal routes for the set of vehicles to complete all clients' orders.

2.1.1 Components of CVRP

In general, CVRP consists of the following components:

1. **Set of nodes** : Nodes are a set of customers and a depot. There can be multiple depots in the problem which is considered as multi-depot CVRP. Each node must be labeled by an amount of demand. In addition, the demand of the depot is set to

zero.

2. **Fleet of vehicles** : Vehicles are used in delivery to serve all customers' demands. Each vehicle must have limited capacity and the number of vehicles in a fleet must be large enough to satisfy the total demand in CVRP.

3. **Cost** : There are various costs in CVRP depending on factors such as truck rental cost, fuel cost, and traveling distance. The cost can be designed to make CVRP the simplest by only considering it as a distance between two nodes where the location of each node is given. In addition, the distance is typically assumed to be Euclidean distance, actual road distance, or it can be considered as the actual cost calculated by multiple factors such as distance, fuel cost, and rental cost.

2.1.2 Notation

All transportation requirements above can be written in mathematical notation. The CVRP is usually represented by an undirected complete graph $G = (V, E)$, where $V = \{0, 1, 2, \ldots, n\}$ is a set of nodes with 0 being the depot and the set of customers being $V' = \{1, 2, \ldots, n\}$. Every node is labeled by a demand $q_i \leq Q$, where $i \in V'$, and Q is a truck capacity, except for the depot whose demand is zero. The CVRP becomes the TSP when only one vehicle can serve all products to all customers, that is $\sum_{i \in V'} q_i \leq Q$. The set of edges is denoted by $E = \{(i, j) \,|\, i, j \in V \text{and } i \neq j\}$ and similarly, every edge is labeled by a cost d_{ij}, where d_{ij} is the cost of traveling from i to j. A route is a set $\mathbf{r} = \{n_1, n_2, \ldots, n_r\} \subset V'$ and all connecting segments between each node are $\{(0, n_1), (n_1, n_2), \ldots, (n_r, 0)\}$. Let m be the number of trucks in the fleet and a set of vehicles $K = \{1, 2, \ldots, m\}$. Each truck in K is assumed to be homogeneous, where every vehicle has the same capacity $Q > 0$. Therefore, the number of vehicles, $m = \text{ceil}\left(\sum_{n \in V} q_n / Q\right)$ is enough for all customers' demand. The route is feasible for CVRP if and only if the total demand in the route does not exceed the vehicle limit capacity Q and the same customer is not repeated in the route i.e. $\sum_{n \in \mathbf{r}} q_n \leq Q$ and $n_i \neq n_j$ for all $n_i, n_j \in \mathbf{r}$. Similarly, a solution $R = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_m\}$ is feasible if and only if all routes $\mathbf{r}_i \in R$ are feasible and $\bigcup_{i=1}^{m} (\mathbf{r}_i) = V'$, which means all routes are the

partitions of the set of customer nodes. In this thesis, we study Euclidean CVRP and non-Euclidean CVRP. The Euclidean CVRP is the CVRP where the cost d_{ij} is Euclidean i.e. $d_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2$ for all $i, j \in V$ where (x_i, y_i) and (x_j, y_j) are coordinates of nodes i and j in V, respectively. Otherwise, The non-Euclidean CVRP has non-Euclidean CVRP distance such that $d_{ij}^2 \neq (x_i - x_j)^2 + (y_i - y_j)^2$ for some $i, j \in V$.

Define a distance matrix D, the $(n+1) \times (n+1)$ matrix, contains a cost d_{ij} for all $i, j \in V$. Note that a Euclidean distance matrix is a distance matrix that contains only Euclidean distance. Otherwise, a non-Euclidean distance matrix is a distance matrix that contains non-Euclidean costs for some $i, j \in V$.

2.1.3 Exact solution of CVRP

The CVRP can be formulated as a mixed integer linear program (MILP) problem. The most straightforward formulation is the following.

$$\text{min} \quad \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{2.1}$$

$$\text{subject to} \quad \sum_{i \in V} x_{ij} = 1 \qquad \text{for all} \quad j \in V \setminus \{0\} \tag{2.2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \text{for all} \quad i \in V \setminus \{0\} \tag{2.3}$$

$$\sum_{i \in V} x_{i0} = |K| \tag{2.4}$$

$$\sum_{j \in V} x_{0j} = |K| \tag{2.5}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \gamma(S) \qquad \text{for all} \quad S \subset V \setminus \{0\}, |S| \geq 2 \tag{2.6}$$

$$x_{ij} \in \{0, 1\} \qquad \text{for all} \quad i, j \in V \tag{2.7}$$

Binary variables x_{ij} for $i, j \in V$ indicate the inclusiveness of arc (i, j) in the solution. If the arc (i, j) is in the route, then $x_{ij} = 1$, otherwise $x_{ij} = 0$. The value $|K|$ is the

minimum number of vehicles used for the delivery and $\gamma(S)$ is the minimum number of vehicles needed for serving customers in the subset $S \subset V \setminus \{0\}$. The objective function (2.1) is to minimize the total traveling cost from nodes by summing all cost d_{ij} where arc (i, j) is used. Constraints (2.2) and (2.3) guarantee that all clients are visited exactly once. Constraints (2.4) and (2.5) confirm that all vehicles leave the depot at the beginning of delivery and return to the depot. Constraints (2.6) are the general subtour elimination constraints. An example of a subtour in CVRP is illustrated in Figure 2.1.

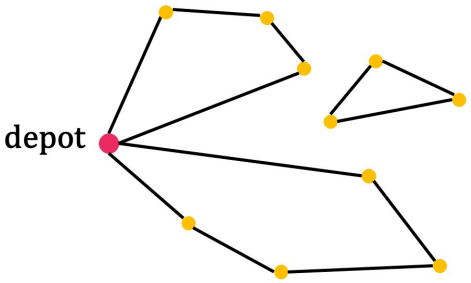


Figure 2.1: Example of a subtour in CVRP

Although the above formulation is the most straightforward, it generates many constraints, especially constraint (2.6). The total number of constraints is $O(2^n)$ which is equal to a number of subset S. To solve the above formulation, families of MILP-based branch-and-cut algorithms are widely used [5, 6, 7]. Other formulations are proposed for CVRP to reduce the number of constraints or to obtain a good bound from its LP relaxation, such as the Two-Commodity Flow Formulation proposed by Baldacci et al. [8]

2.1.4 Heuristic algorithms for CVRP

Heuristic algorithms are popular for solving CVRP because of its speed in problem solving. Although all heuristic algorithms have to trade off between time complexity and the quality of solution, heuristic is still faster than the exact algorithms and the quality of solution is acceptable. There are many heuristic algorithms that are purposed since CVRP has been explored and they are classified into three categories [9] as follows.

2.1.4.1 Constructive heuristic

Constructive heuristics have been proposed for years; however, only some are commonly used. Although the algorithms are very fast for solving CVRP, their solution quality is not great. This is why they are usually included in metaheuristics (explained in the next section) for providing an initial solution. The following are examples of constructive heuristics which are currently in use :

1. **Sequential insertion algorithm**

The subsequential insertion algorithm constructs a route one by one. At the first step, the algorithm randomly chooses unrouted node i to begin forming the route with depot i.e. $(0, i, 0)$. Next, another unrouted node is inserted into the route if the node gives the least cost and the route is still feasible. The route is constructed this way until no more node can be inserted, in which case the algorithm repeats with a new route. The algorithm continues until all nodes are picked. A modified version of this algorithm is called Improved Parallel Insertion Algorithm. This algorithm first forms as many routes as the total number of vehicles and then inserts unrouted nodes simultaneously.

2. **Clarke and Wright Savings heuristic**

Clark and Wright Savings algorithm starts by forming routes $(0, i, 0)$ for all $i \neq 0$. Secondly, it calculates the saving of two nodes i and j, say $s(i, j) = d_{i0} + d_{0j} - d_{ij}$, which infers to the saving after two routes are combined by linking i and j. The set of savings, $F = \{s(i, j) \,|\, i \neq j \,\text{and}\, i, j \neq 0\}$ remains the same throughout the algorithm and it is sorted in decreasing order. Then, two routes, say $\mathbf{r}_1$ and $\mathbf{r}_2$ are combined following the order of savings $s(i, j)$ in F under two conditions: (1) an edge $(i, 0)$ is in $\mathbf{r}_1$ and an edge $(0, j)$ is in $\mathbf{r}_2$ and (2) the total demand does not exceed the truck capacity. The algorithm repeats until no more routes can be combined.

3. **Cluster-First-Route-Second Heuristic**

Cluster-First-Route-Second heuristic [10] has two process stages as its name implies.

The algorithm begins with partitioning a set of nodes into different clusters and then determines routes on each cluster. The sweep algorithm is one type of the Cluster-First-Route-Second heuristic. The algorithm first normalizes all nodes by shifting the depot to the center of the coordinates $(0, 0)$ and other nodes in the same direction through the translation process. Then the coordinate of nodes are transformed into the polar coordinates and the nodes are sorted in increasing order by the angle parameter. In the next step, the first cluster is formed from the initial node n_0 and then the nodes are inserted into the cluster one by one following order in the sorted set until the truck capacity is full and the clustering process stops when all nodes are clustered. Finally, all nodes in each cluster are routed by the sequential insertion algorithm for TSP. The sweep algorithm is restricted to the non-Cartesian coordinate. In addition, another algorithm called Fisher and Jakumar algorithm can be applied for non-Euclidean CVRP instances. The algorithm is also in the cluster-first-route-second algorithm and it works based on a Generalized Assignment Problem (GAP).

2.1.4.2 Classical improvement heuristic

Classical improvement heuristic performs a local search (LS) on an existing solution of CVRP which makes minor changes to the solution that decreases objective function value. There are many different operations in LS, but they all work on the same process. Let S be a solution space of CVRP and its neighborhood $N(S)$. LS operator makes a new move $s' \in N(S)$ which provides the minimum value for objective function i.e. $f(s') \leq f(s)\ \forall s \in N(S)$, where $f : S \to \mathbb{R}$ is an objective function. There are two main operations of move in LS. First, the intra-route operator makes changes within a route only. For example, the 2-OPT operator seeks minimum cost yielded by removing two edges and replacing two edges within the route as shown by Figure 1 in Appendix A. In general, it is possible to perform a local search on multiple edges which is known as the $\lambda-$OPT operator, where λ is the number of edges used to perform. Second, the inter-route operator such as RELOCATE, SWAP, and 2-OPT* make changes between two routes. The RELOCATE removes k consecutive customers from their current route and reinserts them elsewhere in another route. The SWAP operator swaps customers between two

different routes. Finally, the 2-OPT* removes an edge from each route and reconnects them differently. The illustration of RELOCATE, SWAP, and 2-OPT* are shown in Figure 4 to Figure 2 in Appendix A, respectively. In addition, the intra-route operators are performed before the inter-route operators because the size of the neighborhood when performing the intra-route operator is smaller.

2.1.4.3 Metaheuristics

Metaheuristic algorithms mix multiple ideas of the above methods to explore more parts of the search space. The algorithms have a mechanic to escape from the local optimum after performing a local search on the neighborhood of the solution. Examples of metaheuristic algorithms are Simulated Annealing, Tabu Search, Guided Local Search (GLS), and Population-based algorithms. In addition, the concept of different metaheuristic algorithms can be combined to create state-of-the-art methods called hybridization algorithms. However, many hybridization algorithms have high computational complexity, so it needs to trade off between computation time and the quality of the solution. Among many metaheuristic algorithms are proposed in recent years, GLS is the one we use in this study.

GLS is a penalty-based metaheuristic. The algorithm employs a set of solution features which can be any characteristics of the solution for guiding the search. Each defined solution feature has its own cost which represents the information related to the problem. The cost of the feature is used to strengthen the objective function called the augmented objective function. For example in the traveling salesman problem, the feature can simply be an edge between any two nodes and the cost of the feature is the distance of the edge. To explain the GLS in detail, some notations need to be declared. Define an augmented objective function as follows:

$$O'(s) = O(s) + \lambda \sum_{i \in F} \mathrm{Ind}_i(s) \cdot p_i \cdot c_i \tag{2.8}$$

where s is a candidate solution, $O(s)$ is the original objective value of the solution s. The number i ranges over the set of features F and the value p_i is the penalty parameter for a feature i with the corresponding cost c_i, and Ind_i is an indicator function of whether feature i exists in solution s i.e. $\text{Ind}_i(s) = 1$ if feature i exists in the solution s, otherwise $\text{Ind}_i(s) = 0$. The λ is a hyperparameter in GLS for scaling the product of the penalty cost term. The GLS repeatedly calls the local search to minimize the augmented objective function. One may be tempted to penalize only the high-cost features. However, doing so is biased against the features with expensive costs, which is not necessarily a good thing. For example, we know that long edges are a bad characteristic of solutions in TSP since they have a high cost. Therefore, for each iteration, the algorithm will keep penalizing only the same long edges over and over making the search space that the algorithm explores too limited. To remedy this, the utility function is defined to be a criterion for selecting the feature to penalize. The utility of feature i, util_i is defined as follows:

$$\text{util}_i(s) = \text{Ind}_i(s) \times \frac{c_i}{1 + p_i} \tag{2.9}$$

The utility function is a key to selecting unfavorable features that need to be penalized by selecting a feature that maximizes the utility function. First, the utility function of feature i has a positive value when feature i exists in the solution. The higher the cost c_i is, the greater the value of the utility function of feature i will be. Moreover, the larger in p_i or the number of times feature i is penalized, then the lower the utility function will be. This makes the GLS change the penalized target on other features. GLS can be simply implemented as follows:

Algorithm 1 Simple implementation of guided local search algorithm

```
//Initialize//
k ← 0
s_0 ← initial solution inS
s* ← s_0 //s* is expected output of the algorithm//
for each feature i do
    p_i ← 0 //set all penalties to 0//
O' ← O + λ · ∑ Ind_i × p_i × c_i //Set an augmented objective function//
while not StoppingCondition() do
    s_{k+1} ← LocalSearch(s_k, O') //call a local search operator on O'//
    for each feature i do
        util_i(s_{k+1}) ← Ind_i(s_{k+1}) × c_i/(1+p_i)
    if a then feature i makes util_i maximum
        p_i ← p_i + 1
    if O(s_{k+1}) < O(s_k) then
        s* ← s_{k+1} //update the improved solution//
        k ← k + 1
return s*
```

At first, the algorithm initializes a solution, all penalty parameters, and an augmented objective function. After that, the algorithm repeats the penalization process. It starts to find a new solution s_{k+1} by the local search on the augmented objective function of the solution s_k. Then determines a penalized feature i that gives the highest value of util_i of the solution s_{k+1} and also increments the parameter p_i. This process repeats until the expected solution s^* is a new local optimum. Finally, the desired solution s^* is returned when the stop criterion is reached.

2.1.5 Google OR-tools

Google OR-tools [11] is an open-source software suite developed by the Google Optimization team. The suite contains many solvers for combinatorial optimization problems such as assignment problems, routing problems, bin packing problems, network flows problems, and scheduling problems. Google OR-Tools won four gold medals in the 2021 MiniZinc Challenge which is the international constraint programming competition. This will guarantee the efficiency of the Google OR-Tools suite. The solvers are written in C++ and provide wrappers in Python, C# and Java. For routing problems, Google

OR-Tools has packages to solve TSP and varieties of VRP, such as CVRP and VRP with time windows (VRPTW).

There are two phases for using Google OR-tools to solve CVRP. The first phase is called the first solution strategy which contains some operations shown in Table 2.1. Details of each operation are already explained in Section 2.1.4.1.

Operation	**Description**
`PATH_CHEAPEST_ARC`	Sequential Insertion algorithm
`PARALLEL_CHEAPEST_INSERTION`	Improved Parallel Insertion algorithm
`SAVINGS`	Clarke&Wright Savings algorithm
`SWEEP`	Sweep algorithm

Table 2.1: Operations in First Solution Strategy of Google OR-tools

The second phase is called the local search. It employs metaheuristics which involve classical improvement heuristics to improve the solution from the first phase. There are some of the local search options provided and one of them is `GUIDED_LOCAL_SEARCH` which is GLS in **Section 2.1.4.3**.

2.2 Statistical learning model

Let Y be a response variable in the experiment with p independent variables or features denoted by $X_1, X_2, \ldots, X_p$. Assume that there is a relationship between Y and $X = (X_1, X_2, \ldots, X_p)$ which can be written in a form:

$$Y = f(X) + \epsilon \tag{2.10}$$

where f is an unknown function and ϵ is a random error term. A statistical learning model for supervised learning can be referred to as a set of approaches for estimating the function f. There are two reasons one would want to estimate f, [12]. The first

one is for prediction. The features $x = (x_1, \ldots, x_p)$ are treated as inputs and fed into the function f to obtain the estimated value of observation $\hat{y} = f(x)$. The second is for interference, in other words, to understand the association between observation Y and its features X. This is useful because understanding a relationship between the response variable and its features provides specific or deep knowledge of a problem. Sometimes it is easy to identify some important features among a large set of features X. If we expect the function f as a predictor, the estimated value y should be precise. Otherwise, f will be interpretable and can tell the association between Y and features X when f has less complexity. However, the results obtained might be low in precision. A trade-off between precision and interpretability must be considered before employing a statistical learning model. There are many practical learning models in real-world problems and in this study, we only introduced two well-known models in detail: decision tree and support vector machine.

2.2.1 Decision tree

The decision tree model is good for making an inference between observation values and features because it represents their relationship by a top-down tree chart illustrated by Figure 2.2(a). The example shows decisions based on the observed data which has two features sex and age to get the rules telling the personal status of people. The tree keeps splitting into branches following the logical condition of the features and then stops when a decision is made. The branch in which the tree stops splitting is called the leaf.

A decision tree divides feature space X into some non-overlapping regions and these regions are referred as conditions of features used for classifying observation value. The regions are shown in Figure 2.2(b). A strategy for dividing regions depends on how much accuracy we will get compared to the observed data. The accuracy we mentioned corresponds to the cost function we want to minimize. If the cost is very low, then the tree fits the observed data or the accuracy is high. An example of the cost is a Gini index which indicates an impurity from classification. The Gini index is calculated from the amount of probability of a specific feature that is classified incorrectly, $\text{Gini} = 1 - \sum_i p_i^2$.

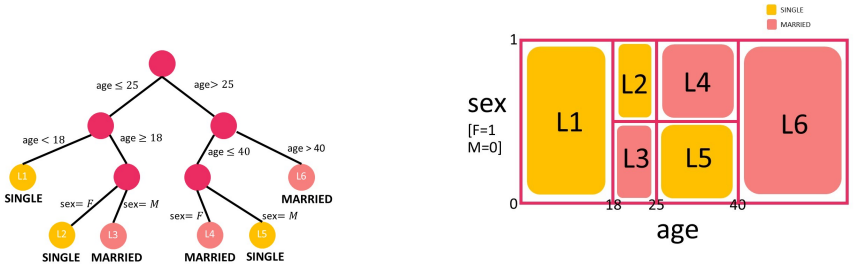
A value p_i is the probability of an element being classified for a distinct class. The Gini index ranges between 0 to 0.5 for two-label classification. It is zero when a node is split and no other classes are classified incorrectly. Otherwise, The index is highest at 0.5 when the actual class and incorrect class are distributed equally. In addition, there are some hyperparameters to control the tree to stop growing and reduce overfitting such as the maximum depth, and the minimum number of samples in each leaf. This can help reduce the complexity of the model and makes it less overfitting.

2.2.2 Support vector machine

Support vector machine or SVM performs well in a classification task. The model attempts to determine two hyperplanes on the feature space X, so that it can separate observed data into their classes in the condition that the data is linearly separable. The region between two hyperplanes is called a margin. In fact, many hyperplanes can achieve the task and the one we are seeking has the maximum margin i.e. the distance between two hyperplanes is maximized as shown in Figure 2.3(a). The maximum margin ensures to some extent that the hyperplane will also classify new data which is never observed. The margin which has no data points lie in is called the hard margin. Unfortunately, the data in a real-world problem may not be separated clearly by linear hyperplanes. The margin which allows some misclassification lie in is called the soft margin for handling such a problem and making the result as good as possible. The example of a soft margin is shown in Figure 2.3(b). In addition, if there are p features of observations, the model will seek two hyperplanes which are $p-1$ dimensional subspace.

We can see that the SVM is unlike the decision tree in interpretability. It is easy to interpret how the decision tree works by the tree chart even though we are dealing with many features. However, the SVM works like a black box when there are more than three features because we cannot draw the illustration and see the relation between features. Nevertheless, the advantage of the SVM is that it can handle the non-linear problem smoother than the decision tree (since the decision tree divides feature space into some partitions) by a kernel method. The method maps the original features not

linearly separable to the higher dimensional space and determines hyperplanes to separate data that lies in the higher dimension.



(a) The tree chart

(b) The partitioning of the decision tree

Figure 2.2: The illustration of the decision tree

(a) An example of a hard margin

(b) An example of a soft margin

Figure 2.3: An example of SVM

2.2.3 Feature importance

A feature importance is a score that represents the impact of each feature with respect to a learning model. In the decision tree, we split any nodes by selecting a feature yielding the lowest impurity of the classification task to divide the feature space. This implies that the selected feature once has an impact on the model. Therefore, feature importance of each feature is determined by the frequency they are being used to split a node in the tree. Note that, the feature importance does not indicate predictive power by itself but indicates how important this feature is for a particular model. In addition, it is difficult to calculate feature importance in some models such as non-linear models. A permutation feature importance is a method that can generate a feature importance for those models, see [13].

2.3 Multi-dimensional Scaling

Multi-dimensional scaling (MDS) is a tool mainly to represent similarity or dissimilarity among pairs of objects as a distance in the low-dimensional space. The main purposes of MDS are in the scope of visualization [14], and in the thesis, it is a necessary measurement to bridge between the Euclidean and the non-Euclidean space. The MDS can determine the new position of points in the preferred k-dimensional Euclidean space related to information in a distance matrix of the problem using the idea of principle component analysis (PCA) [15]. Given a distance matrix D and let $X_{n\times k}$ be a matrix of the coordinate of n points in k-dimensional space. Many applications of MDS are purposed, and one of them is feature extraction in CVRP. Jussi Raku et al. [16] presented hundreds of features related to CVRP instances used for improving the automatic algorithm configuration i.e. the algorithms to help automatically configure hyperparameters of the heuristic algorithm for the CVRP being considered. One category of features requires nodes information. Unfortunately, some benchmark instances they used provide only a distance matrix. Then, the MDS is employed to determine the coordinate matrix X:

Step 1 Let $B = XX^\top$ be Gram matrix.

Step 2 Rewrite B in terms of a distance matrix D: $B = J(-D^2/2)J\top$, where $D^2 = \left[d_{ij}^2\right]$ and $J = I - \mathbf{1}\mathbf{1}^\top/n$

Step 3 By the idea of PCA, $M = -D^2/2 \approx Q\Lambda Q^\top$, where $\Lambda_{k\times k}$ is a diagonal matrix formed by positive eigenvalues of D sorted in decreasing order and $Q_{n\times k}$ are formed by the corresponding eigenvectors.

Step 4 $XX^\top = JMJ^\top \approx (JQ\Lambda^{1/2})(JQ\Lambda^{1/2})^\top$, where $\Lambda^{1/2} = \left[\lambda_{ij}^{1/2}\right]$ and λ_{ij} is the elements of Λ.

Step 5 $X = JQ\Lambda^{1/2}$

The coordinate of points contained in X lie in k-dimensional Cartesian coordinate and their centroid (average coordinate of points in each axis) is located at the origin.

However, the new Euclidean distance matrix D' yielded by MDS only approximates the original distance matrix D. Therefore, measurement of similarity between the two distance matrices is necessary to measure how close they are.

2.4 Similarity between distance matrices

In this section, we introduce two different matrix similarity measurements, namely, matrix norm and Mantel test.

2.4.1 Matrix norm

A classic tool for measuring distance between two matrices is the matrix norm. The distance is determined by calculating a norm of the difference of both matrices. There are some well-known matrix norms, for example:

Let $A = [a_{ij}]_{n\times n}$.

1. Frobenius norm

$$||A||_2 = \left(\sum_{j=1}^{m} \sum_{i=1}^{m} |a_{ij}^2| \right)^{1/2} \tag{2.11}$$

2. L_p norm

$$||A||_p = \left(\sum_{j=1}^{m} \sum_{i=1}^{m} |a_{ij}^p| \right)^{1/p} \tag{2.12}$$

3. Max norm

$$||A||_{\max} = \max_{i,j=1,\ldots,m} |a_{ij}| \tag{2.13}$$

Let A and B be $n \times n$ matrices, then the distance between matrices A and B is $||A - B||$, where $|| \cdot ||$ can be one of the norms defined above.

2.4.2 Mantel test

The previous measurement represents the numerical closeness between two matrices. Next, we introduce a statistical approach to measure a correlation between two distance matrices: the Mantel test [17]. The test is commonly used in Ecology. For example, to get a correlation between two distance matrices corresponding to their metrics in the ecology field. Let us say the first metric calculates the genetic distance between species and the second calculates distance data from geographical information. Then, the Mantel test is employed to test a correlation between the two distance matrices generated before. The procedure of the test is based on a randomization test. To implement the test, first let two distance matrices from different metrics be D^x and D^y, where entries of D^x are a distance between objects x_i and x_j or d^x_{ij} and also similar to D^y which contains d^y_{ij}. The two distance matrices must be symmetric and have the same dimension. Next, compute a correlation as follows:

$$r = \frac{\sum_i \sum_j (d^x_{ij} - \bar{D^x}) \times (d^y_{ij} - \bar{D^y})}{\sqrt{\text{var}(D^x) \times \text{var}(D^y)}} \tag{2.14}$$

where $\bar{D}$ and $\text{var}(D)$ are the mean and variance of all entries in the distance matrix, respectively. This standardized version of the Mantel test is the Pearson correlation efficient. The value of r ranges between -1 (perfect negative correlated) to 1 (perfect positive correlated). However, if $r = 0$, it indicates no relationship between the matrices. Simultaneously, the collection of r's is computed by randomly permuting rows of D^y. The idea of the test is, r should be higher than r' since the value of r is not disturbed by randomizing the rows that will make a change in the relationship. Next, the empirical test is performed to get a probability of observing r's that are higher than r, then return the probability which is a p-value under the null hypothesis that there is no relationship between two distance matrices. For example, if 999 of r's are lower than r, then the probability is $1/(999+1)$ where 1 is also observed which means to r, so that the p-value = 0.001.

2.4.3 non-Euclidean vs. Euclidean Distance

There is a numerical value indicating the difference between two distance matrices [18] called deviation factor $\mathrm{DF}(D^x, D^y) = \binom{|V|}{2}^{-1} \sum_{i,j \in V} \frac{d^x_{ij}}{d^y_{ij}}$ which is the average of the ratio between distances of each pair of nodes. The study [18] computes DF of an actual road distance and a Euclidean distance between cities in well-known countries. The number of nodes is different for each country corresponding to the availability of the geographical data. The result is that DF is ranging between 1.12 and 2.10. The value of DF depends on many factors such as node sample size, road density and connectivity, transportation rules, and geographical obstacles like mountains, rivers, and hilliness. If two distance matrices are close, then DF is close to 1. Otherwise, DF will be much different from 1.

In addition, there is the study of outcomes from solving TSP in real road networks and Euclidean TSP [19]. They collected the road network distance within big cities such as Paris, London, and Seoul. Then, two distance matrices were generated to be used in the TSPs of each city. Note that the non-Euclidean distance they used was the shortest path based on the real road network. The DF of two distance matrices for each city ranged between 1.1 to 1.4 and the Pearson correlation coefficients were around 9.7 which is very high. They solved both TSPs to find an optimal solution and the result shows that the ratio between the optimal solution from TSP with a non-Euclidean distance matrix and the Euclidean one was ranging around 1.2 to 1.5 which reflected a large gap between them.

2.5 Characteristics of CVRP solutions

Arnold and Sörensen proposed a framework to obtain problem-specific knowledge that can distinguish characteristics of good CVRP solutions from bad ones. The authors defined ten solution metrics representing the appearance of the solutions. Then they used these features to classify high-quality and low-quality solutions through a statistical learning model to see which ones perform well in classifying near-optimal and non-optimal solutions. They also applied the knowledge for guiding a search in their modified GLS and their result showed improvement in search compared with the original GLS. Arnold

and Sörensen's work is engaging with promising applications.

To explain Arnold and Sörensen's work, let us expand the CVRP notations as follows. Let $G = (V, E)$ be a graph representing CVRP, where V is a set of nodes and all nodes lie in two-dimensional space and V' be a set of customer nodes. Let R be a solution of CVRP contains all routes $\mathbf{r} = \{n_1^{\mathbf{r}}, n_2^{\mathbf{r}}, \ldots, n_{|\mathbf{r}|}^{\mathbf{r}}\}$, where $n_i^{\mathbf{r}}$ is the i^{th} customer in route $\mathbf{r}$ and all connecting segments between each node are in $\{(0, n_1^{\mathbf{r}}), (n_1^{\mathbf{r}}, n_2^{\mathbf{r}}), \ldots, (n_{|\mathbf{r}|}^{\mathbf{r}}, 0)\}$. Let $G_{\mathbf{r}}$ be the centroid of nodes in route and $L_{G_{\mathbf{r}}}$ be the line that passes through the depot and the centroid. A Euclidean distance between node n_1 and n_2 is denoted by $\mathrm{d}(n_1, n_2)$ and a Euclidean distance between a node n and the line $L_{G_{\mathbf{r}}}$ is $\mathrm{d}(L_{G_{\mathbf{r}}}, n)$ which can be positive and negative depending whether the node is on the right and the left side of the line respectively. An angle between node n_1 and n_2 with respect to the depot is denoted by $\mathrm{rad}(n_1, n_2)$. Finally, $\mathrm{I}(\mathbf{r}_i, \mathbf{r}_j)$ represents the total number of times the segments of route $\mathbf{r}_i$ intersect segments of route $\mathbf{r}_j$. Arnold and Sörensen's solution metrics considered in this paper are defined below.

S1 - Average number of intersections per customer

$$\frac{\sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} \mathrm{I}(\mathbf{r}_i, \mathbf{r}_j)}{N} \tag{2.15}$$

S2 - Average longest distance between two connected customers per route

$$\frac{\sum_{\mathbf{r} \in R} \max_{i \in \{1, \ldots, |\mathbf{r}|-1\}} \mathrm{d}(n_i^{\mathbf{r}}, n_{i+1}^{\mathbf{r}})}{|R|} \tag{2.16}$$

S3 - Average distance between depot to directly-connected

$$\frac{\sum_{\mathbf{r} \in R} \left(\mathrm{d}(0, n_1^{\mathbf{r}}) + \mathrm{d}(n_{|\mathbf{r}|}^{\mathbf{r}}, 0) \right)}{2|R|} \tag{2.17}$$

S4 - Average distance between routes (their centers of gravity)

$$\frac{\sum_{\mathbf{r}_1 \in R} \sum_{\mathbf{r}_2 \in R \setminus \mathbf{r}_1} \mathrm{d}(G_{\mathbf{r}_1}, G_{\mathbf{r}_2})}{2|R|} \tag{2.18}$$

S5 - Average width per route

$$\frac{\sum_{\mathbf{r} \in R} \left(\max_{i \in 1,\ldots,|\mathbf{r}|} \mathrm{d}(L_{G_\mathbf{r}}, n_i) - \min_{i \in 1,\ldots,|\mathbf{r}|} \mathrm{d}(L_{G_\mathbf{r}}, n_i) \right)}{|R|} \tag{2.19}$$

S6 - Average span in radian per route

$$\frac{\sum_{\mathbf{r} \in R} \max_{i,j \in \{1,\ldots,|\mathbf{r}|\}} \mathrm{rad}(n_i^\mathbf{r}, n_j^\mathbf{r})}{|R|} \tag{2.20}$$

S7 - Average compactness per route, measured by width

$$\frac{\sum_{\mathbf{r} \in R} \sum_{i=1}^{|\mathbf{r}|} |\, \mathrm{d}(L_{G_\mathbf{r}}, n_i)|}{N} \tag{2.21}$$

S8 - Average compactness per route, measured by radian

$$\frac{\sum_{\mathbf{r} \in R} \sum_{i=1}^{|\mathbf{r}|} \mathrm{rad}(G_\mathbf{r}, n_i)}{N} \tag{2.22}$$

S9 - Average depth per route

$$\frac{\sum_{\mathbf{r} \in R} \max_{i \in \{1,\ldots,|\mathbf{r}|\}} \mathrm{d}(n_i^\mathbf{r}, 0)}{|R|} \tag{2.23}$$

S10 - Standard deviation of the number of customers per route

$$\sqrt{\frac{\sum_{\mathbf{r}\in R}(|\mathbf{r}| - \frac{N}{|R|})^2}{|R|}} \tag{2.24}$$

These solution metrics are defined under the hypothesis that they can be the features which are able to be a separating criterion of solution quality. Behind the idea of design, the solution metrics are expected to be a criterion for the guided local search mentioned in Section 2.1.4.3. To roughly explain, the design is based on hypothesis that good solutions should contain routes that rarely overlap and this can be indicated using the solution metric (S1). The overlapping of segments of different routes refers to the frequency that different trucks travelling to the near region. The good CVRP solutions should also have short edges (S2) and the edges connected to the depot should also be short (S3). The solution metric (S4) measures how well the routes separated as well and good solutions should ideally have clearly separated non-overlapping routes because otherwise the routes that are too close to one another or overlapping may be combined and yield a better solution. Moreover, to reduce overlapping, the average width per route (S5) should be small. The solution metric (S6) measures the average route span around the line $L_{G_\mathbf{r}}$ by looking at the average maximum angle between two customers with respect to the depot per route. Having a small (S6) value indicates most routes are in slender tapered shape. The solution metrics (S7) and (S8) give the average route compactness measured by the width and the radian, respectively. They can also be interpreted as the average broadness per route. If these values are small, the segments of most routes are close to $L_{G_\mathbf{r}}$. The metric (S9) gives the average depth of the route measured by the average distance between the farthest customer and the depot per route. The metric (S9) should be small for a good solution. Lastly, the balance of the number of customers in each route is measured by the solution metric (S10). The visualization of some solution metrics is shown in Figure 5 in Appendix B. We also give three solutions from the same Euclidean CVRP which have three different appearances as shown in Figure 6 with values of each solution metric and costs of the solutions in Table 1 in Appendix B.

In addition, Arnold and Sörensen also defined features called instance metrics. The metrics are characteristics of CVRP instance which have a purpose to make CVRP solutions different at the instance level while solution metrics make solutions different at the solution level. The instance metrics are calculated using parameters from a corresponding CVRP instance and the solution metrics are calculated using parameters from a corresponding CVRP solution as defined above. Each instance metric is defined as follows:

I1 - Number of customers

$$N \tag{2.25}$$

I2 - Number of routes (trucks)

$$m \tag{2.26}$$

I3 - Degree of capacity utilization

$$\frac{\sum_{i \in V'} q_i}{m \times C} \tag{2.27}$$

I4 - Average distance between each pair of customers

$$\text{mean}(\{d(i,j) \mid i,j \in V', i \neq j\}) \tag{2.28}$$

I5 - Standard deviation of the pair-wise distance between customers

$$\text{std}(\{d(i,j) \mid i,j \in V', i \neq j\}) \tag{2.29}$$

I6 - Average distance from customers to the depot

$$\text{mean}(\{d(i,0) \mid i \in V'\}) \tag{2.30}$$

I7 - Standard deviation of the distance from customers to the depot

$$\text{std}(\{d(i,0) \mid i \in V'\}) \tag{2.31}$$

I8 - Standard deviation of the radians of customers towards the depot

$$\text{std}(\{\text{rad}(i,0) \mid i,j \in V', i \neq j\}) \tag{2.32}$$

CHAPTER III

METHODOLOGY

In this chapter, we explain our methodology in detail from the beginning until we reach our goal of identifying the most distinctive characteristics of non-Euclidean CVRP solutions. The process is summarized by the diagram in Figure 3.1. First of all, we generated our non-Euclidean CVRP datasets. Then MDS is employed to provide a local requirement for calculating solution metrics and instance metrics of the non-Euclidean CVPR solution. In addition, the instances are simultaneously solved by using a routing solver package in Google OR-tools. After that, we had sufficient information to calculate the instance metrics and solution metrics. Then we used them as features to process in the statistical learning framework and the most distinctive characteristic was finally returned. Each step we mentioned will be clarified in the following sections.

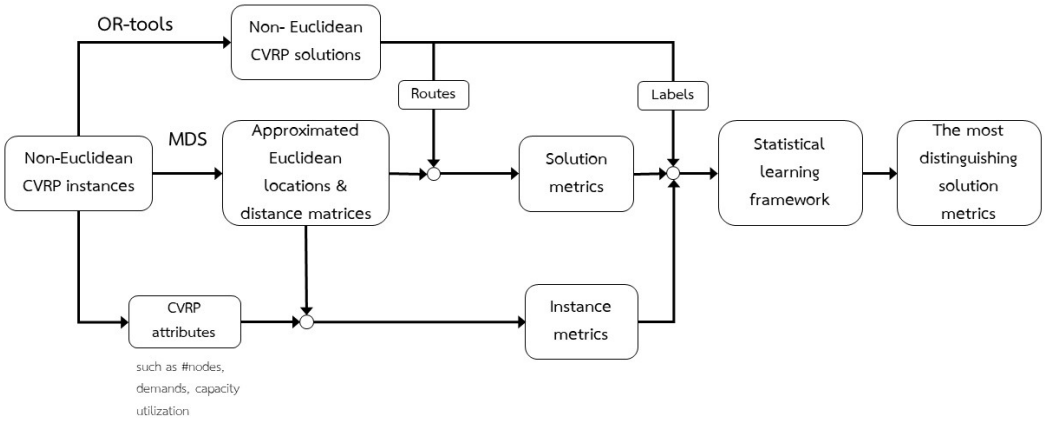


Figure 3.1: Overview of the methodology of the thesis

3.1 Generating non-Euclidean CVRP instances

In this study, we generated eight classes of non-Euclidean CVRP instances. Each class varies in the following attributes to complete all components of CVRP mentioned in Chapter 2.

1. **Node attributes**: We considered three attributes of nodes including the number of nodes, the location of a depot, and the customer's demand. There are two sizes of problem instances where the numbers of nodes are different. The small and big sizes are denoted by SMALL and BIG, respectively. The SMALL instances contain 20 to 50 customers and the BIG instances contain 70 to 100 customers. Each node is located on a 200×200 grid in the first quadrant. Likewise, the demand is either randomly generated from 1 to 5, or is set to be the same for all customers and the demand of a depot is zero. A depot location is either located at the center of the grid at position $(100, 100)$ or at the edge of grid $(0, 100)$. This will generate two types of instances: one with a depot at the center and another with a depot located at the middle of the left edge. The number of instances in each class and the overview is shown below in Table 3.1. We name each class by three characters to indicate variant in three attributes: (1) size is aliased by S=SMALL and B=BIG, (2) demand is aliased by F=FIXED and V=VARIED, and (3) depot location is aliased by C=CENTER and E=EDGE.

2. **Fleet attributes**: An amount of truck capacity is generated depending on the size of CVRP instances. The SMALL instances are served by trucks with 10-unit capacity and the BIG instances are served by trucks with 30-unit capacity. A truck capacity is assumed to be the same for all instances in the same class, but the number of routes will vary depending on the total demand of each generated instance.

Classes	Size	#Nodes	Demand	Truck Cap.	Depot pos.	#Instances
class1-SFC	SMALL	20-50	1	10	center	2000
class2-SVC	SMALL	20-50	1-5	10	center	2000
class3-SFE	SMALL	20-50	1	10	edge	2000
class4-SVE	SMALL	20-50	1-5	10	edge	2000
class5-BFC	BIG	70-100	1	30	center	2000
class6-BVC	BIG	70-100	1-5	30	center	2000
class7-BFE	BIG	70-100	1	30	edge	2000
class8-BVE	BIG	70-100	1-5	30	edge	2000

Table 3.1: Details of generated CVRP instances

In addition, the distance matrix of each instance is calculated using the Manhattan distance. This is assumed that all nodes take place in a block-shaped city like Manhattan, making the instances non-Euclidean. We choose the Manhattan norm because it has an example in the real world. Demographic information of the generated non-Euclidean CVRP classes are shown as boxplots in Figure 7 - 11 in Appendix C.

3.2 Mapping to Euclidean space using multi-dimensional scaling

Although we generate the Manhattan distance from the coordinates of each node of the problem, we cannot use both the distance and the coordinates to determine the solution metrics defined in Section 2.5. The reason is that the solution metrics are defined under the Euclidean space. Therefore, we attempt to consider the non-Euclidean CVRP in the new space in which components of the non-Euclidean problem can be treated as Euclidean. In other words, if we can find the coordinate of each node of the CVRP in that space and the corresponding distance matrix is the same as the original, then it is able to calculate the solution metrics under Euclidean space. Such a problem is called Euclidean distance geometry problem or Euclidean-DGP [20]. There are some methods for solving DGP. However, Euclidean-DGP is an NP-hard problem which implies that it consumes a lot of time to solve for an exact solution. Therefore, we choose MDS which provides an approximated solution to the problem. The MDS returns the new position of nodes

that lie in the preferred k-dimensional Euclidean space and a new corresponding distance matrix will become Euclidean. Note that we call the location which is the outcome from MDS an *approximated Euclidean location* and its corresponding distance matrix is called an *approximated Euclidean distance matrix*. We varied the number k from 1 to 5. The number k that makes the approximated distance matrix get close to the original one as much as possible is selected. The closeness between the new and the original matrices is quantified by the similarity measurement described in Chapter 2: Frobenius matrix norm. The number k is selected if it produces the least matrix norm. We also perform the mantel test for monitoring a correlation between the two distance matrices. The mantel correlation coefficient is very high, more than 0.9 and the p-value is less than 0.05 to reject the null hypothesis. From the experiment, k is unsurprisingly mostly equal to 2 because the instances are initially generated in two-dimensional space and Manhattan distance is close to Euclidean distance. Nevertheless, a value of 3 is more suitable for k in some instances. We show the overview of the process in Figure 3.2.

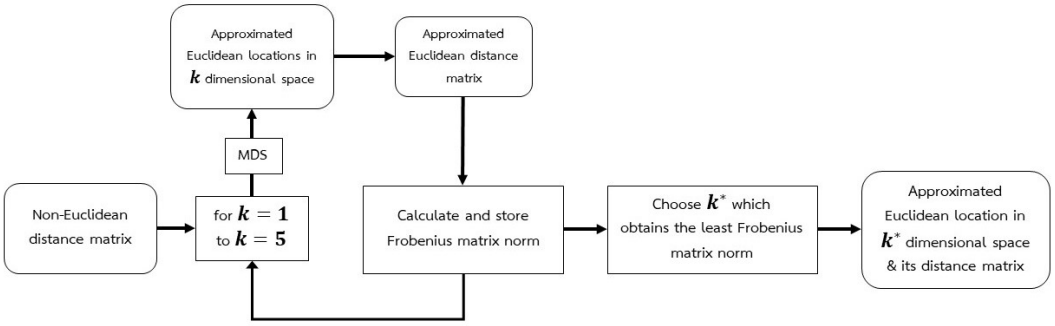


Figure 3.2: Overview of the methodology of applying MDS

The Frobenius matrix norm of the difference between the approximated Euclidean distance matrix and the original distance matrix is shown by a boxplot in Figure 13 and the correlation coefficient from Mantel test between both matrices is shown by a boxplot in Figure 14 in Appendix C. We also calculate the deviation factor between an approximated Euclidean matrix and the original one. The average value of DF is around 1.05 and the standard deviation of DF is around 0.005 for all classes. This implies that the elements of the original Euclidean distance matrix are greater than the approximated

Euclidean distance matrix around 5%.

3.3 Non-Euclidean CVRP solutions

As mentioned in the previous chapter, we use Google OR-tools routing packages to solve the non-Euclidean CVRP instances where the process can be divided into two phases. For the first phase, we obtain an initial solution of CVRP using First Solution Strategy and the `PATH_CHEAPEST_ARC` is selected. Next, we improve the initial solution using the local search operator `GUIDED_LOCAL-SEARCH`. In this study, we do not require a solver which provides excellent quality of the solution. We select this option for solving non-Euclidean CVRP because the option is consistent in performance. It can solve every generated CVRP instances and the design of its algorithm is deterministic.

For each non-Euclidean CVPR that is solved, two solutions are collected and labeled. The first solution is an initial solution provided by `PATH_CHEAPEST_ARC` and is labeled as non-optimal solution. The initial solution then is improved by using `GUIDED_LOCAL_SEARCH` for searching for another better solution in the neighborhood of the initial solution. The solution from the improvement process will be labeled as a near-optimal solution. Note that to clearly separate the difference between the non-optimal solution and the near-optimal solution, we specify a gap in cost between both solutions. Indeed, we set the gap of solution cost in SMALL instances and BIG instances to be 5%-10% and 3%-10%, respectively. We extend the range of the gap of BIG instances because it takes a long time to search for the improved solution which has a gap between 5%-10% due to the large neighborhood of an initial solution. As described before, we generated 2,000 instances in each class and we obtained two solutions for each instance, namely a non-optimal solution and a near-optimal solution, then our dataset contains 4000 data points for all data classes. In addition, We draw a boxplot to summarize a difference in cost between two solutions for each class as shown in Figure 10 in Appendix C.

3.4 Building a dataframe of features of CVRP solutions

The characteristic of CVRP solution or solution metrics S2 to S10 defined by Arnold and Sörensen in Chapter 2 was adopted. Note that the solution metric S1 does not perform well in the experiment because it requires a long computational time, so we drop it from our consideration. Because of the MDS, some of the approximated locations are determined to be in 3D Euclidean space. This makes the definition of one solution metric incompatible with the higher dimensional space of node locations. Therefore, we adjust the definition of the solution metric to fit the solutions in 3D Euclidean space. The solution metric which must be adjusted is the solution metric S5. Originally, S5 characterizes the width of a route in the solution by summing up the distance between the center line $L_{G_\mathbf{r}}$ and the two farthest points on both sides of the center line as shown in the Figure 5(a) in Appendix B. Therefore, we redesign the solution metric S5 in a similar way. The width of each route in a solution lies in the higher dimensional would be a diameter of a cylinder centered along $L_{G_\mathbf{r}}$ which wraps all customers in the route tightly. We also define our solution metrics called S11 and S12. S11 is the average radius per route, where the radius of a route is the maximum distance between its centroid and its nodes. S12 is another version of route compactness measured by the average distance of each node from its center of gravity. The figures of the three mentioned solution metrics are shown in Figures 3.3 to 3.5 and the definitions are as follows:

S5* - Redefined solution metric S5

$$\frac{\sum_{\mathbf{r}\in R}\left(\max_{i\in 1,\ldots,|\mathbf{r}|} 2\,\mathrm{d}(L_{G_\mathbf{r}}, n_i)\right)}{|R|} \tag{3.1}$$

S11 - Average radius per route

$$\frac{\sum_{\mathbf{r}\in R}\left(\max_{i\in 1,\ldots,|\mathbf{r}|} \mathrm{d}(G_\mathbf{r}, n_i)\right)}{|R|} \tag{3.2}$$

S12 - Average compactness per route, measured by average distance of each node from its centroid

$$\frac{\sum_{\mathbf{r}\in R}\sum_{i=1}^{|\mathbf{r}|}\mathrm{d}(G_{\mathbf{r}}, n_i)}{N} \tag{3.3}$$

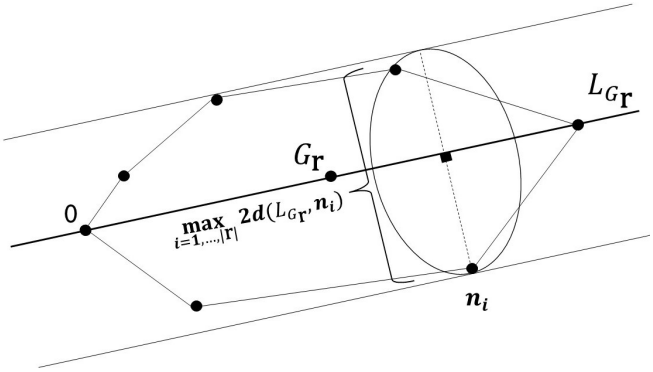


Figure 3.3: Visualization of the redefined solution metric S5

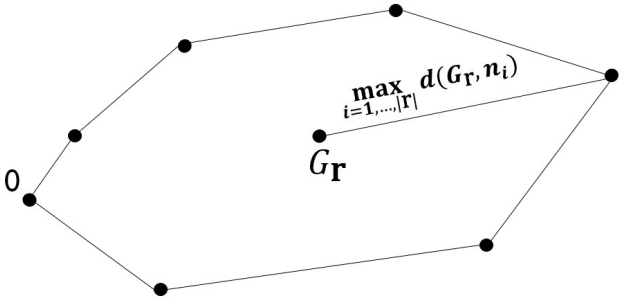


Figure 3.4: Visualization of the new solution metric S11

In each generated instance, a value of each instance metric I1 to I8 and each solution metric S2 to S10 is determined by the definitions using the approximated information obtained in Section 3.2 and Section 3.3. The overview of each solution metric is visualized by boxplots by the size of instances and label of solution in Appendix D. Note that we also call the instance metrics and the solution metrics as the features of non-Euclidean CVRP solution. The instance metrics and solution metrics are collected to build a dataframe of features of each non-Eucldean CVRP solution. In addition, the dataframe includes a label of the solution (near-optimal and non-optimal). We do the same process for all eight

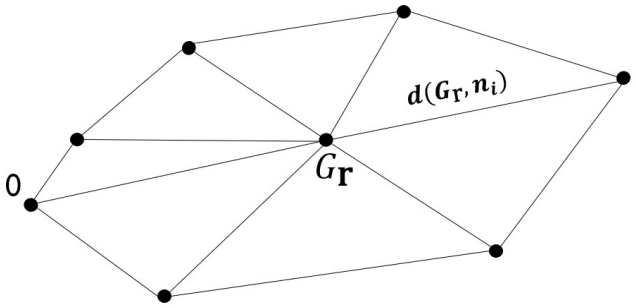


Figure 3.5: Visualization of the new solution metric S12

classes mentioned in Section 3.1 and we will obtain total eight dataframes. The dataframes will fuel a data-driven technique to determine the most distinctive characteristic of a non-Euclidean CVRP.

3.5 Statistical learning framework for measuring features performance

Statistical learning models such as SVM and decision tree are employed to be as a tool to measure the performance of each solution metric. First, we test the overall performance of features by feeding all of the features to SVM. After that we employ another model, a decision tree, to measure the performance of each solution metric. Instead of feeding the whole dataframe, we select only the instance metrics I1 to I8 and only one solution metric as features. Then, the accuracy of the prediction of each solution metric corresponding to the one we push to the model is collected. The most distinctive characteristic as we said before is the solution metric which yields the highest accuracy of prediction. We use a decision tree to extract some rules from the results; these rules might be practical for guiding the search. The example of the dataframe used for SVM and decision tree is shown in Table 3.2 and Table 3.3.

The implementation is written in Python on Google Colaboratory. We import SVM and decision tree from scikit-learn [21], a popular library package. In addition, every parameters we set for running SVM is the default. It is optional for us to adjust the SVM to get a better result because we only want to test if the given features have any predictive power. For the decision tree, We set the Gini index for the criterion of

splitting and maximum depth for splitting branch from the start node or `maxdepth` to be 9 which is equal to the total number of features feeding to the model.

Data	**Instance metrics**				**Solution metrics**				**Label**
points	I1	I2	$\cdots$	I8	S2	S3	$\cdots$	S12	
1									non-opt
2									near-opt
3									non-opt
$\vdots$									$\vdots$

Table 3.2: Input dataframe for SVM

Data	**Instance metrics**								**Solution**	**Label**
points	I1	I2	I3	I4	I5	I6	I7	I8	**metric**	
1										non-opt
2										near-opt
3										non-opt
$\vdots$										$\vdots$

Table 3.3: Input dataframe for decision tree

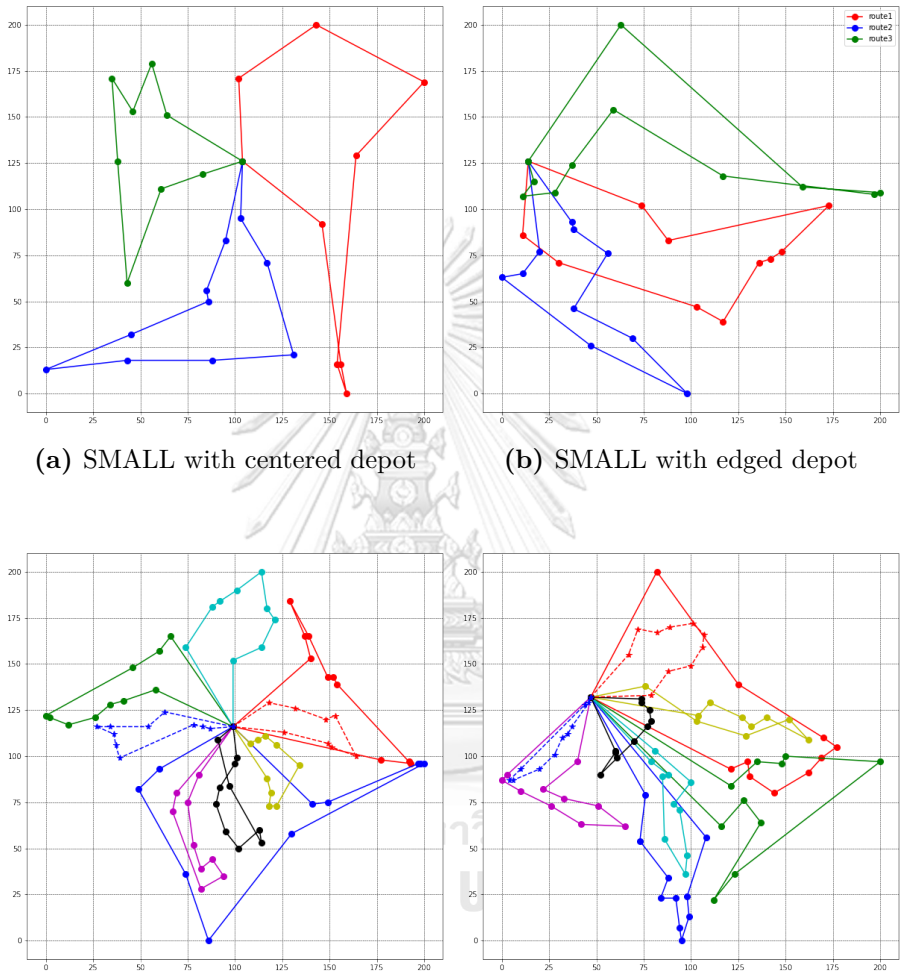
CHAPTER IV

EXPERIMENTS AND RESULTS

4.1 Predictive power of solution metrics

As mentioned in the previous chapter, we first test whether the given features have any predictive power by feeding all of them to the support vector machine model. The average accuracy of prediction from 5-fold cross-validation is shown in Table 4.1. All average accuracy obtained is more than 79% which is quite high. This implies the features have the potential for the model to learn patterns of the solutions and their features. These results can be discussed from two perspectives.

- **SMALL instances versus BIG instances**: The accuracy of prediction BIG instance datasets (classes 5-8) is generally higher than SMALL instances datasets (classes 1-4). Since the BIG instances are rich in information, so the characteristics of solutions stand out and the solution will become easier to classify by the hyperplanes of SVM.
- **depot at the center versus depot at the edge**: the datasets which have a depot at the center (classes 1,2,5,6) obtain higher accuracy than the datasets which have a depot at the edge (classes 3,4,7,8). The depot position makes the appearance of the solution different as shown in Figure 4.1. The figures below are obtained by plotting approximated locations of each node and then are scaled to range 0 to 200 which is the original range. Notice that the edged-depot instances have many overlapped routes and each route is narrow. This might make the appearance of near-optimal solution and non-optimal solution similar and the learning model cannot separate them very well.



(a) SMALL with centered depot **(b)** SMALL with edged depot

(c) BIG with centered depot **(d)** BIG with edged depot

Figure 4.1: Example of non-Euclidean SMALL and BIG CVRP solution after approximating location by MDS

Classes	#Data points	Average Prediction of Accuracy (%)
class1-SFC	4000	83.05
class2-SVC	4000	80.65
class3-SFE	4000	79.67
class4-SVE	4000	78.78
class5-BFC	4000	86.65
class6-BVC	4000	86.32
class7-BFE	4000	81.95
class8-BVE	4000	81.12

Table 4.1: Average prediction accuracy by SVM from 5-fold cross-validation using all features (I1-I8 and S2-S10)

4.2 Identifying the most distinctive characteristic

Next, to find the most distinctive characteristic, the decision tree is selected. We feed all instance metrics I1 to I8 and only one solution metric to the model to get an accuracy of prediction. Then, the solution metric that yields the highest accuracy of prediction will be the most distinctive characteristic of non-Euclidean CVRP solutions. The rounded average accuracy by the decision tree from 5-fold cross-validation which is higher than 55% is presented in Table 4.2

Classes	**Rounded average accuracy of prediction (%)**										
	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
class1-SFC	65	59	**69**	66	66	65	**69**	55		64	**69**
class2-SVC	64	58	64	66	64	65	66			66	**67**
class3-SFE	62		**66**	61	55	61	58	56	57	56	64
class4-SVE	61	55	**64**	62	55	60	57	57	56	58	60
class5-BFC	68	64	**71**	**71**	64	69	68		55	66	**71**
class6-BVC	67	67	62	**71**	64	68	67		55	68	67
class7-BFE	59	57	**65**	61	55	64	58	58	56		62
class8-BVE	55	**65**	58	63	52	61	56	58	56	58	56
average	62	59	64	**65**	59	64	62	55	55	60	64

Table 4.2: Rounded average prediction accuracy by the decision tree from 5-fold cross-validation of each solution metric

From Table 4.2, the solution metrics which have low predictive power (accuracy of prediction lower than 60) are S9 and S10. The solution metric which gives the highest accuracy on average is S5. This is confirmed by Table 4.3 which shows the one-tailed Wilcoxon signed-rank test between the accuracy obtained by solution metric S5 and others. The result shows that the accuracy obtained by the solution metric S5 is significantly higher than the accuracy obtained by another solution metric at 99% confidence level, except for solution metric S4 (average distance between routes) and S12 (average compactness by the distance between nodes and a centroid).

Our proposed solution metrics S11 and S12 have comparable performance with others although S11 does not perform better than S12 in most classes. We can interpret the meaning of the definition of S11 as the average radius of routes in solution, which is similar to S9 (average depth of routes). However, the accuracy of prediction obtained by S11 is higher than by S9. Moreover, the new solution metric S12 performs very well and better than other solution metrics which reflect the compactness of a route such as S7 (compactness by width) and S8 (compactness by radian). Therefore, it will be better if

we consider the compactness by the distance between nodes and the centroid instead.

pair of solution metrics to compare	**difference of average accuracy of prediction (%)**	**difference of accuracy of prediction tested by Wilcoxon Signed-Rank Test**
(S5,S2)	3.86	S5 gives significantly higher accuracy. (W=723.0, p<0.001)
(S5,S3)	8.95	S5 gives significantly higher accuracy. (W=810.0, p<0.001)
(S5,S4)	1.38	Not statistically different. (W=455.0, p=0.15)
(S5,S6)	9.03	S5 gives significantly higher accuracy. (W=806.0, p<0.001)
(S5,S7)	2.70	S5 gives significantly higher accuracy. (W=652.0, p<0.001)
(S5,S8)	5.52	S5 gives significantly higher accuracy. (W=728.5, p<0.001)
(S5,S9)	15.65	S5 gives significantly higher accuracy. (W=820.0, p<0.001)
(S5,S10)	16.17	S5 gives significantly higher accuracy. (W=820.0, p<0.001)
(S5,S11)	7.55	S5 gives significantly higher accuracy. (W=791.5, p<0.001)
(S5,S12)	1.34	Not statistically different. (W=482.0, p=0.17)

Table 4.3: Wilcoxon signed-rank test results at 99% confidence level for accuracy comparison between S5 and every other solution metrics

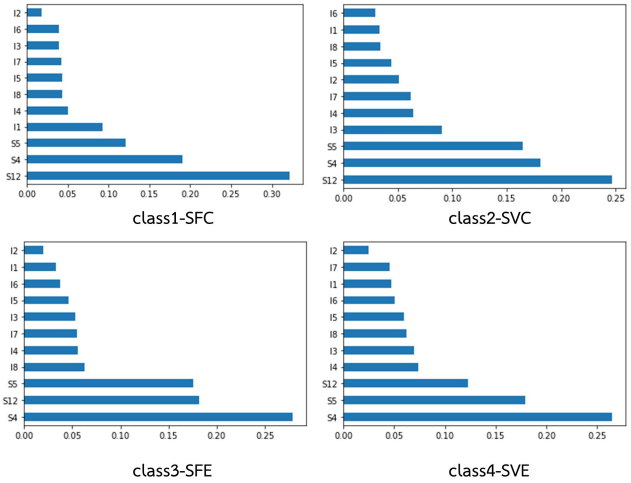
Note that we cannot and should not directly compare our results and Arnold and Sörensen's study because there are two major differences. Firstly, our solution metrics are generated from an approximated Euclidean locations and distance matrix, so the information which the solution metrics represent is approximated. The main disadvantage

of the approximated Euclidean distance matrix obtained from MDS is that the pair-distance ordering does not remain the same. In other words, if the elements d_{ij} and d_{kl} of a non-Euclidean distance matrix D are such that $d_{ij} < d_{kl}$, then it might be the case that the approximated Euclidean distance matrix D' given by MDS occurs $d'_{ij} > d'_{kl}$. Therefore, this makes the two problems inequivalent.

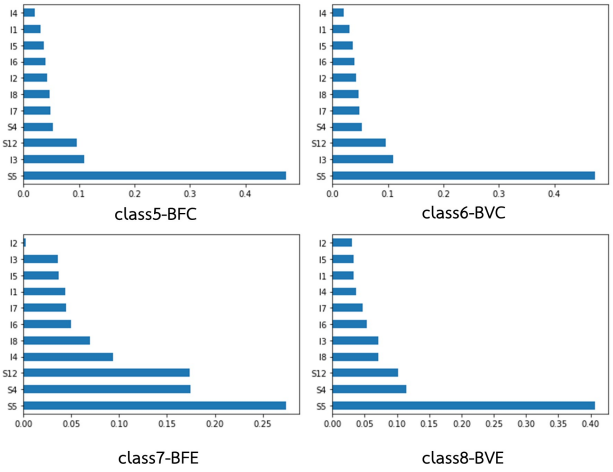
4.3 Decision rules

The decision rule is a set of logical conditions of features used to determine what class of a datapoint is. In the decision tree, a decision rule is obtained from conditions corresponding to a leaf node of a tree diagram as described in Section 2.2.1. At this time, we feed the model by only three features S4, S5, and S12 which yield a high accuracy of prediction for the classification task to help determine the decision rules. In addition, we do not use the instance metrics because of two reasons:(1) the feature importance of instance metrics compared to the solution metrics is very low for most classes, and (2) the small number of features makes it easy to determine decision rules. The feature importance of each instance metric and solution metric S4, S5, and S12 are shown in Figure 4.2. We set the Gini index for the criterion of splitting and `maxdepth` to be 11 which is equal to the number of features we use. The decision rules are shown in Tables 4.4 and 4.5. In the tables, we select the decision rules of each class that has the least Gini index in both non-optimal solution and near-optimal solution leaf nodes. For example, for the decision rules in class1-SFC in Table 4.4, the solution is non-optimal if S4 < 115 and S12 > 67. This rule has a Gini index of 0.368 which reflects the impurity of the leaf node that contains 779 non-optimal solutions (actual class) and 250 near-optimal solutions

For each label, the decision rules from each class are similar. On one hand, we interpret that the routes of each non-optimal solution might be close together since S4 is small. And routes are also wide and not compact (each node in a route is located far from the centroid) since the values of S5 and S12 are big. On the other hand, the routes of a near-optimal solution are not overlapped much (S4 is big) and each route tends to be narrow or compact (S5 and S12 are small). However, the rules only cover some solutions



(a) Feature importance in SMALL classes



(b) Feature importance in BIG classes

Figure 4.2: Feature importance of S4,S5, S12 and all instance metrics in all classes

in our dataset. They cover 25%-39% of total non-optimal solutions and 2%-34% of total near-optimal solutions for SMALL classes. Likewise, the rules cover 14%-37% of total non-optimal solutions and 6%-39% of total near-optimal solutions for BIG classes.

Classes	**Decision rules**			**Label**	**Score**
	S4	S5	S12		(Gini)
class1-SFC	< 115		> 67	non-optimal	0.368 [non=779,near=250] (non-opt 39% of 2000)
	> 126	< 124	< 67	near-optimal	0.172 [non=49,near=467] (near-opt 23% of 2000)
class2-SFE	< 113	> 158	> 77	non-optimal	0.259 [non=476,near=86] (non-opt 24% of 2000)
	< 113	< 126	< 77	near-optimal	0.193 [non=4,near=33] (near-opt 2% of 2000)
class3-SVC	< 129	> 162	> 70	non-optimal	0.349 [non=680,near=198] (non-opt 34% of 2000)
	> 123	< 123	< 70	near-optimal	0.218 [non=72,near=505] (near-opt 25% of 2000)
class4-SVE	< 104	> 132		non-optimal	0.276 [non=500,near=99] (non-opt 25% of 2000)
	> 114	< 141	< 84	near-optimal	0.304 [non=131,near=569] (near-opt 28% of 2000)

Table 4.4: Decision rules of SMALL classes

Classes	**Decision rules**			**Label**	**Score**
	S4	S5	S12		(Gini)
class5-BFC	< 124	> 103		non-optimal	0.159 [non=746,near=71] (non-opt 37% of 2000)
		< 78	< 58	near-optimal	0.135 [non=34,near=434] (near-opt 22% of 2000)
class6-BFE	< 123	> 123		non-optimal	0.214 [non=361,near=50] (non-opt 18% of 2000)
		< 100	< 71	near-optimal	0.193 [non=228,near=787] (near-opt 39% of 2000)
class7-BVC	< 117	> 116		non-optimal	0.099 [non=345,near=19] (non-opt 17% of 2000)
		< 79		near-optimal	0.19 [non=54,near=454] (near-opt 23% of 2000)
class8-BVE		> 137	>67	non-optimal	0.183 [non=282,near=32] (non-opt 14% of 2000)
		< 80		near-optimal	0.114 [non=8,near=124] (near-opt 6% of 2000)

Table 4.5: Decision rules of BIG classes

CHAPTER V

CONCLUSIONS AND FUTURE WORK

This thesis aims to find the most distinctive characteristic of the non-Euclidean CVRP solution to extend Arnold and Sörensen's work. We generated non-Euclidean CVRP for eight classes which were different in size, variation of demand, and depot location. We proposed the idea of using MDS to overcome the non-Euclidean condition. To that end, the MDS helped determine the k-dimensional coordinate of nodes in non-Euclidean CVRP. Therefore, we could treat the non-Euclidean CVRP as Euclidean by using approximated Euclidean locations and its distance matrix. This allowed us to calculate the solution metrics proposed by Arnold and Sörensen. We followed their definition of solution metrics S1-S10. However, we did not use S1 due to high computational time. In addition, we needed to adjust the definition of solution metric S5 to be compatible with approximated locations which sometimes lie in the higher dimensional space. We also defined new solution metrics S11 (an average radius per route) and S12 (average compactness per route by an average distance of each node from its centroid). We calculated the values of the solution metrics of non-Euclidean CVRP and used them for the statistical learning framework. For each solution metric S2 to S12, we fed them one by one with all instance metrics I1 to I8 to the decision tree and got an accuracy of prediction corresponding to the solution metric.

The results from the decision tree showed that S4, S5, and S12 are the top three solution metrics in terms of prediction accuracy with S5 being the highest. Although MDS gave us an approximated location and distance matrices, it provided us with results comparable to Arnold Sörensen's work in terms of the solution metric which gave the highest accuracy of prediction. Moreover, our solution metric S12 performed better than S7 and S8 which were also average compactness of routes. Although the solution metric S11 did not stand out in performance, it gave much higher accuracy of prediction com-

pared to S9 (average depth of routes) which was similar to S11. In addition, the statistical Wilcoxon signed rank test of the difference between the accuracy of prediction obtained by S5 and others suggested that S4 (average distance between routes) and S12 also gave a comparable performance to that of S5. Therefore, the most distinctive solution metrics could be S4, S5, and S12.

To obtain an insight into non-optimal and near-optimal characteristics using the influential solution metrics, we selected S4, S5, and S12 to make decision rules by feeding them to the decision tree again. At this time, we extracted a decision rule of non-Euclidean CVRP solutions from the leaf node with the minimum Gini index of non-optimal and near-optimal solutions. From the rules, non-optimal solutions generally have small S4 and high S5 and S12. This could be implied that non-optimal solutions have wide routes, are less compact (not compressed to the centroid), and there is a lot of overlap between routes. These were the opposite of the rules of near-optimal solutions. In addition, the rules covered up to 37% of total non-optimal solutions and 39% of total near-optimal solutions. These decision rules are specific knowledge that can be used to guide a local search to a promising area.

This study can still be improved in many aspects. In fact, the most important key of this study is MDS. The more we can improve the outcome of MDS, the better results for our work can be. In addition, if there exists a method to replace MDS which preserves the paired-distance ordering in the new distance matrix, our results will have a much more impact. Developing such a method could be interesting and challenging research work. Moreover, the distance matrix with non-Euclidean distance other than Manhattan distance should also be explored since Manhattan distance is not enough to represent real-world problems. It would be interesting to see if real-world datasets would yield the same results. Furthermore, the application of the outcome of this study can be further explored by applying our most solution metric to the guided local search similar to Arnold and Sörensen's other work [2]. The study adopted the most distinctive characteristic and applied it in their modified GLS which give remarkable results in their large-sized Euclidean CVRP (up to 10,000 nodes). Although our work is similar to theirs, we cannot

do as they did directly since we consider the problem based on the approximation. We should always realize that our current method is an approximation under the new space from MDS. To apply any knowledge from the work in a guided search, we first must transform them back to the original space. The transformation step is also challenging in its own right.

REFERENCES

[1] F. Arnold and K. Sörensen, "What makes a vrp solution good? the generation of problem-specific knowledge for heuristics," *Computers & Operations Research*, vol. 106, pp. 280–288, 2019.

[2] F. Arnold, M. Gendreau, and K. Sörensen, "Efficiently solving very large-scale routing problems," *Computers & operations research*, vol. 107, pp. 32–42, 2019.

[3] P. Toth and D. Vigo, *Vehicle Routing*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014.

[4] S. Irnich, P. Toth, and D. Vigo, *Chapter 1: The Family of Vehicle Routing Problems*, pp. 1–33. Society for Industrial and Applied Mathematics, 2014.

[5] F. Semet, P. Toth, and D. Vigo, *Chapter 2: Classical Exact Algorithms for the Capacitated Vehicle Routing Problem*, pp. 37–57. Society for Industrial and Applied Mathematics, 2014.

[6] M. Poggi and E. Uchoa, *Chapter 3: New Exact Algorithms for the Capacitated Vehicle Routing Problem*, pp. 59–89. Society for Industrial and Applied Mathematics, 2014.

[7] L. Gilbert, "What you should know about the vehicle routing problem," *Naval Research Logistics (NRL)*, vol. 54, pp. 811 – 819, 12 2007.

[8] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation," *Operations Research*, vol. 52, pp. 723–738, 10 2004.

[9] G. Laporte, S. Ropke, and T. Vidal, *Chapter 4: Heuristics for the Vehicle Routing Problem*, pp. 87–116. Society for Industrial and Applied Mathematics, 11 2014.

[10] S. Avdoshin and E. Beresneva, "Constructive heuristics for capacitated vehicle routing problem: a comparative study," *Proceedings of the Institute for System Programming of the RAS*, vol. 31, pp. 145–156, 09 2019.

[11] L. Perron and V. Furnon, "Or-tools (version 7.2)," 2019.

[12] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R.* Springer Texts in Statistics, Springer New York, 2014.

[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[14] I. Borg and P. J. F. Groenen, *Modern multidimensional scaling: Theory and applications, 2nd ed.* Springer Science + Business Media, Springer New York, 2005.

[15] J. C. Gower, "Principal coordinates analysis," *Wiley StatsRef: Statistics Reference Online*, pp. 1–7, 2014.

[16] R. Jussi, K. Tommi, and M. Nysret, "Feature extractors for describing vehicle routing problem instances," *OASICS; 50*, pp. 1–13, 2016.

[17] A. Diniz-Filho, T. Soares, V. L. Jacqueline Lima, Ricardo Dobrovolski, M. Telles, T. Rangel, and L. Bini, "Mantel test in population genetics," *Genetics and molecular biology*, vol. 36, pp. 475–485, 2013.

[18] R. Ballou, H. Rahardja, and N. Sakai, "Selected country circuity factors for road travel distance estimation," *Transportation Research Part A: Policy and Practice*, vol. 36, no. 9, pp. 843–848, 2002.

[19] B. Boyacı, T. H. Dang, and A. N. Letchford, "Vehicle routing on road networks: How good is euclidean approximation?," *Computers & Operations Research*, vol. 129, p. 105197, 2021.

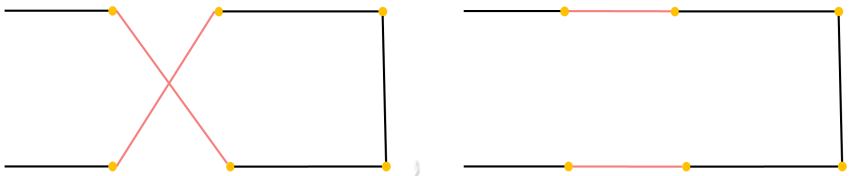
[20] L. Liberti and C. Lavor, *Euclidean distance geometry*, vol. 133. Springer, 2017.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

APPENDIX

APPENDIX A : Local search operators in CVRP

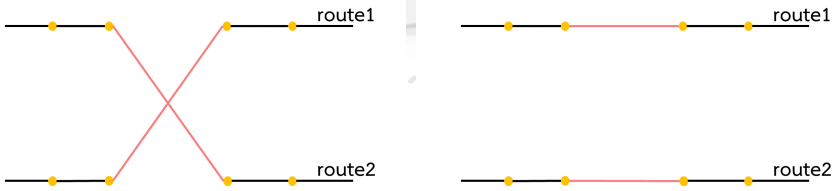
- Intra-route operator.
  - 2-OPT operator



(a) before performing 2-OPT **(b)** after performing 2-OPT

Figure 1: An example of 2-OPT operator

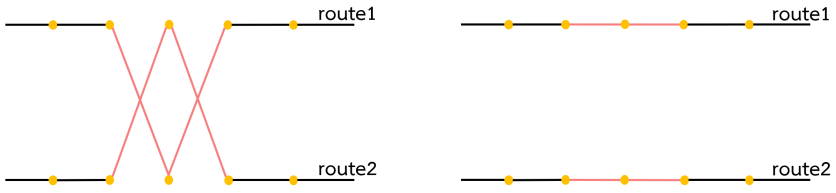
- Inter-route operator
  - 2-OPT* operator



(a) before performing 2-OPT* **(b)** after performing 2-OPT*

Figure 2: An example of 2-OPT* operator

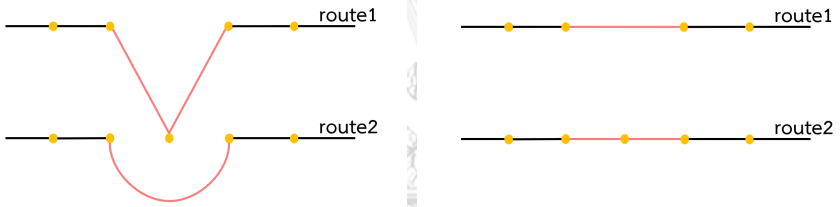
– SWAP operator



(a) before performing SWAP **(b)** after performing SWAP

Figure 3: An example of SWAP operator

– RELOCATE operator

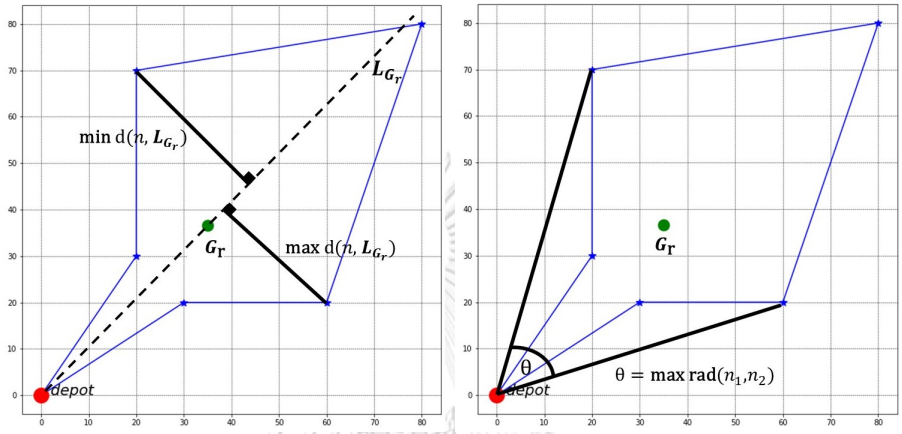


(a) before performing RELOCATE **(b)** after performing RELOCATE

Figure 4: An example of RELOCATE operator

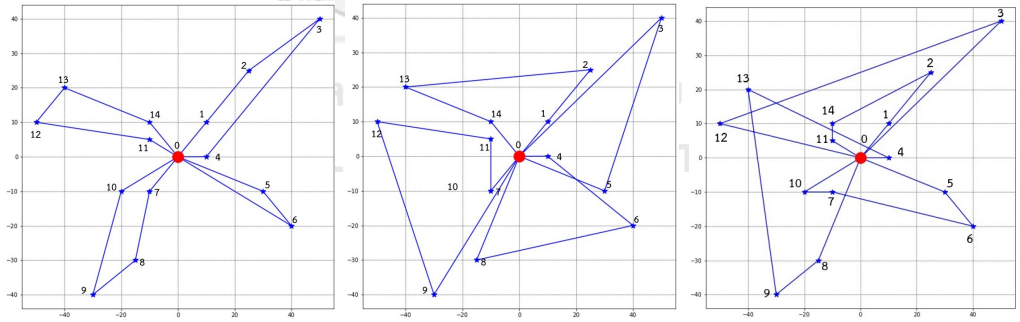
APPENDIX B : Solution metrics and CVRP solutions

Some solution metrics can reflect a geometrical appearance of each route in the solution such as the solution metric S5 and S6 which can be implied to a width of each route. The figures below show a component for calculating the solution metric S5 and the solution metric S6.



(a) Components for calculating S5 **(b)** Components for calculating S6

Figure 5: Visualization of solution metrics S5 and S6



(a) Solution-A **(b)** Solution-B **(c)** Solution-C

Figure 6: Visualization of different solutions of the same CVRP

We genarate a Euclidean CVRP instance and determine three solutions which are different in an appearance as the following.

- Solution-A consists of narrow routes and n
- Solution-B consists of wide routes
- Solution-C consists of overlapped routes

The solution metrics S1 to S12 of each solution is calculated in Table 1. The cost of each solution metrics is also included in the table.

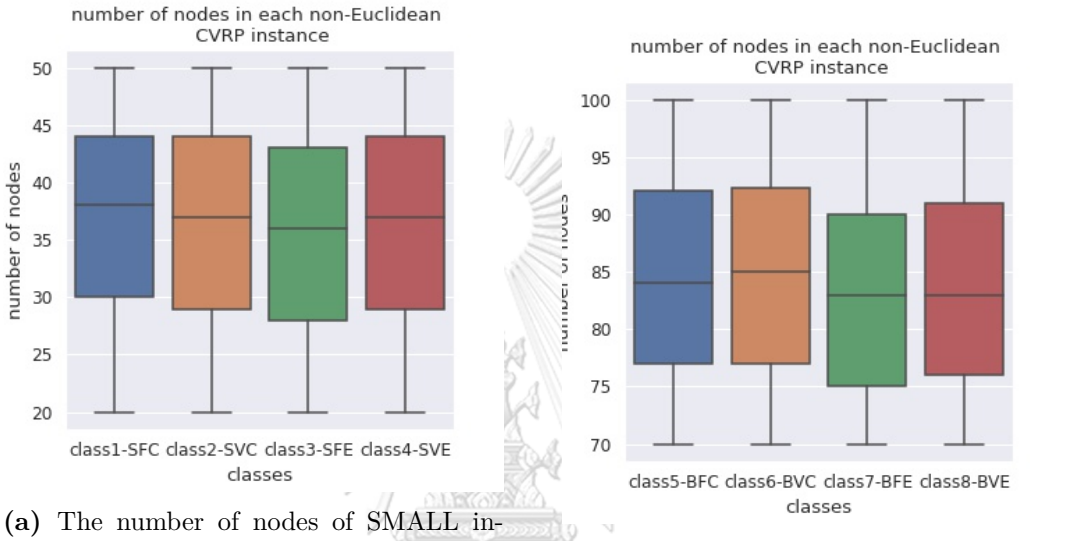
Metrics	**solution-A**	**solution-B**	**solution-C**
S1	0	0.25	1.75
S2	35.25	56.50	63.0
S3	20.00	28.75	29.37
S4	38.40	31.97	22.71
S5	13.73	55.86	63.73
S6	0.54	1.58	2.30
S7	4.11	19.28	27.17
S8	0.18	0.72	1.03
S9	52.00	50.500	48.25
S10	0.90	0.87	0.90
S11	29.96	35.57	38.59
S12	28.68	33.79	33.5
COST	130	145	174

Table 1: Comparison between CVRP solutions and solution metrics

APPENDIX C : Demographic information by CVRP attributes

We draw boxplots to show overall and distribution of attribute values of the generated non-Euclidean CVRP instances. The plots are as follows:

- the number of nodes

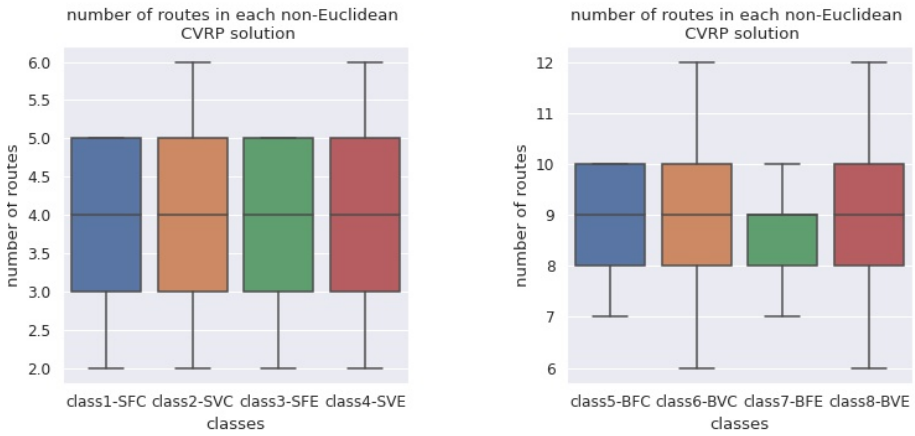


(a) The number of nodes of SMALL instances

(b) The number of nodes of BIG instances

Figure 7: Boxplots of number of nodes of SMALL and BIG instances

- the number of routes

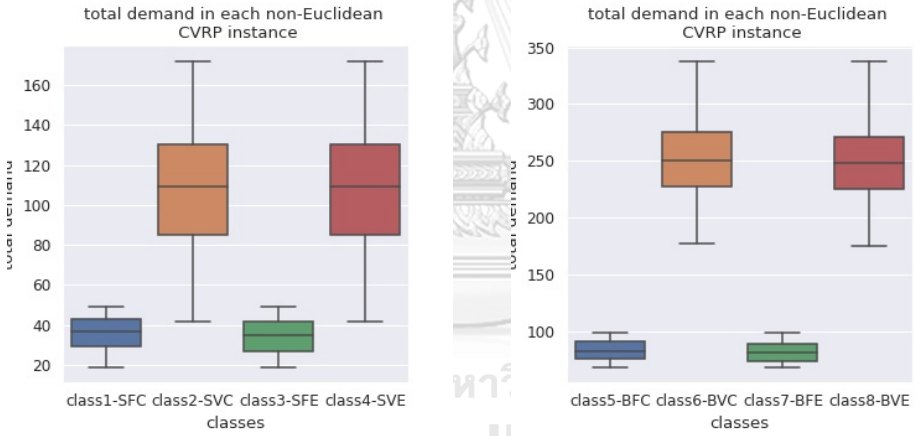


(a) The number of routes of solution in SMALL instances

(b) The number of routes of solution in BIG instances

Figure 8: Boxplots of number routes of solutions in SMALL and BIG instances

- total demand



(a) Total demand of SMALL instances

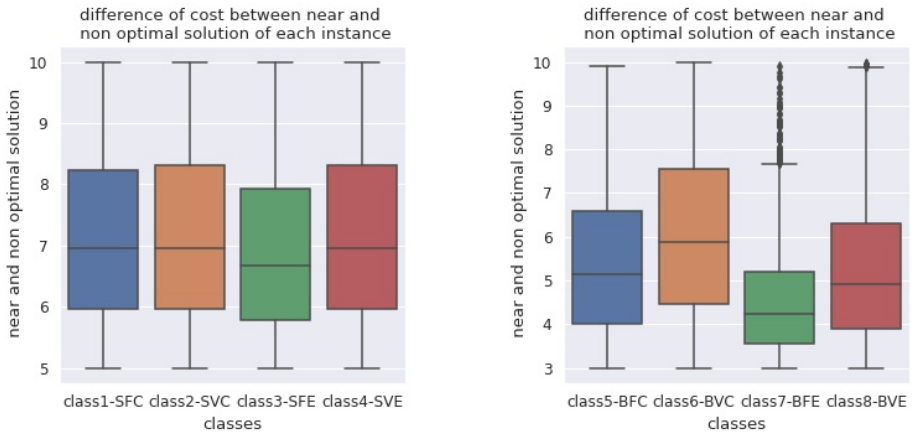
(b) Total demand of BIG instances

Figure 9: Boxplots of total demand of SMALL and BIG instances

- difference of cost between near-optimal and non-optimal solutions

- capacity utilization

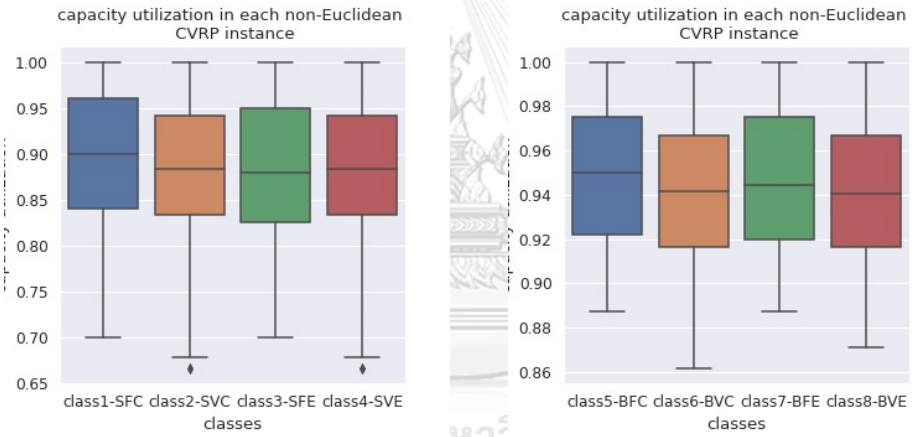
$$\text{capacity utilization of solution} = \frac{\text{total demand}}{\#\text{routes} \times \text{truck capacity}} \tag{1}$$



(a) percentage of difference between cost of non-optimal solution and near-optimal solution in SMALL instances

(b) percentage of difference between cost of non-optimal solution and near-optimal solution in BIG instances

Figure 10: Boxplot of difference between cost of non and near optimal solution



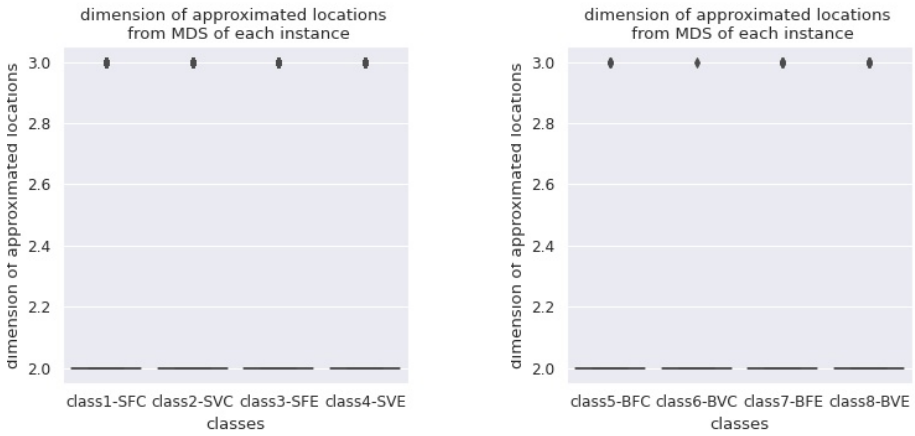
(a) Capacity utilization of SMALL instances

(b) Capacity utilization of BIG instances

Figure 11: Boxplots of capacity utilization of SMALL and BIG instances

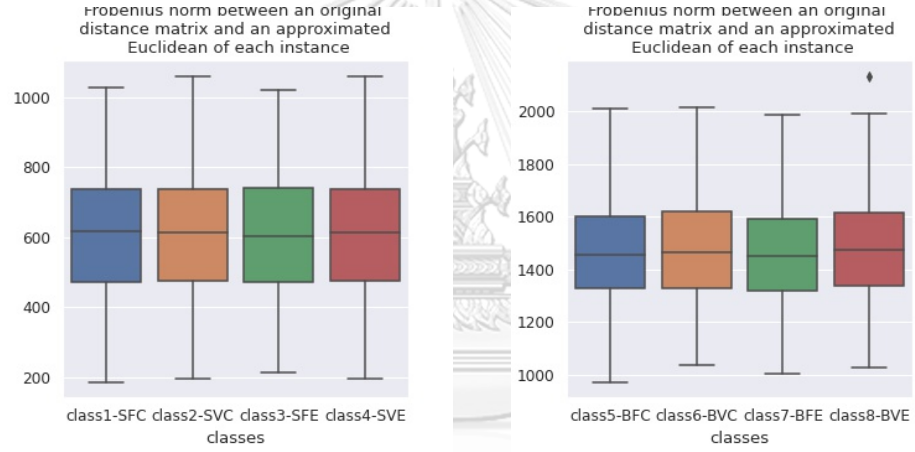
APPENDIX D : Demographic information of outcome of MDS

- The number of dimension of approximated Euclidean locations
- Frobenius norm of difference between an approximated Euclidean distance matrix and non-Euclidean distance matrix
- Mantel correlation coefficient between an approximated Euclidean distance matrix and non-Euclidean distance matrix



(a) #dimension in SMALL instances **(b)** #dimension in BIG instances

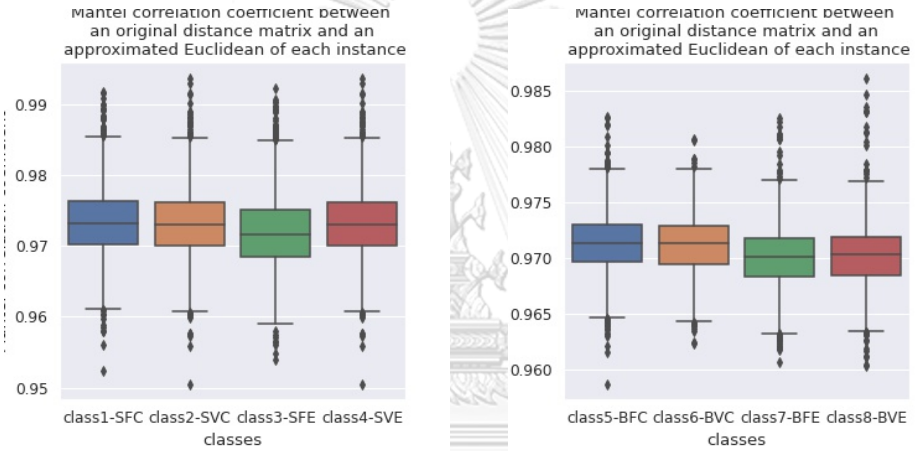
Figure 12: Boxplots of the number of dimension of approximated Euclidean locations in SMALL and BIG instances



(a) Frobenius norm from MDS step of SMALL instances **(b)** Frobenius norm from MDS step of BIG instances

Figure 13: Boxplots of Frobenius norm from MDS step in SMALL and BIG instances

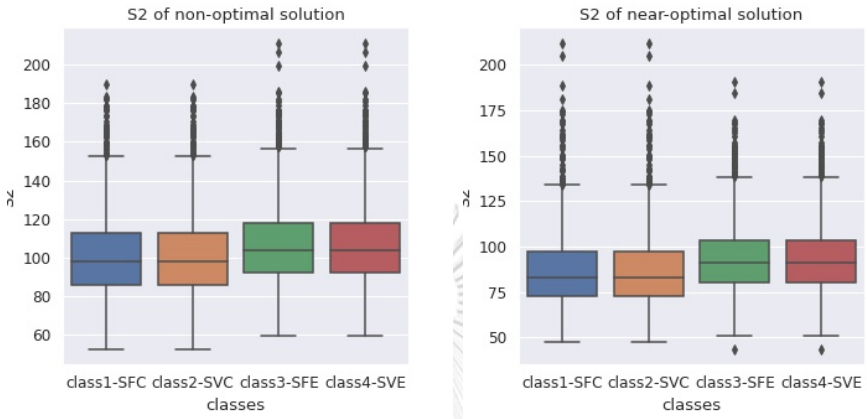
APPENDIX E : Overview of solution metrics



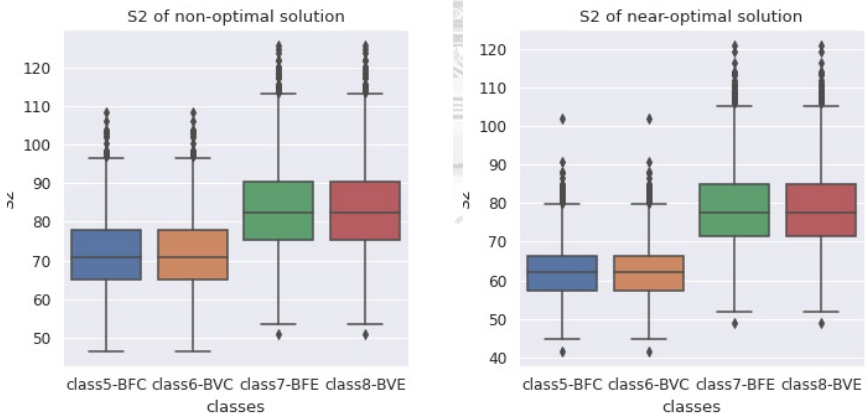
(a) Mantel correlation coef. from MDS step of SMALL instances

(b) Mantel correlation coef. from MDS step of BIG instances

Figure 14: Boxplots of Mantel correlation coefficient from MDS step of BIG instances of SMALL and BIG instances

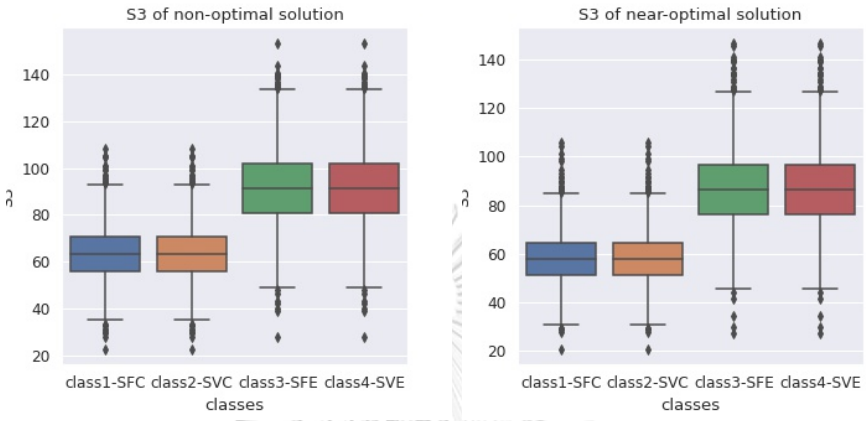


(a) S2 of non-optimal solution in SMALL **(b)** S2 of near-optimal solution in SMALL



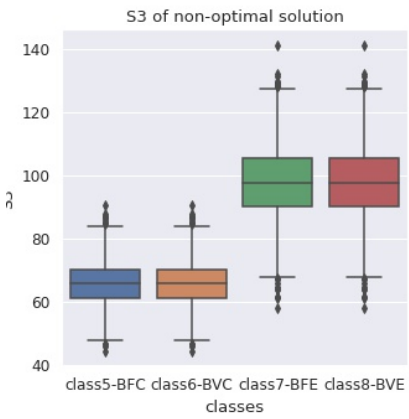
(c) S2 of non-optimal solution in BIG **(d)** S2 of near-optimal solution in BIG

Figure 15: Boxplots of S2 by size of instance and label of solution

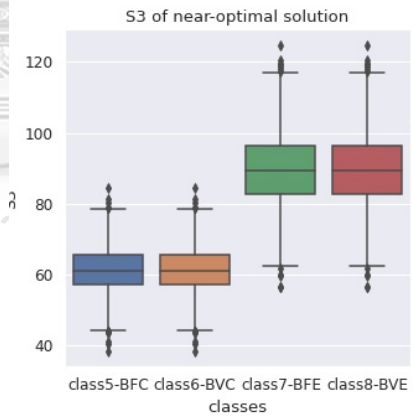


(a) S3 of non-optimal solution in SMALL

(b) S3 of near-optimal solution in SMALL

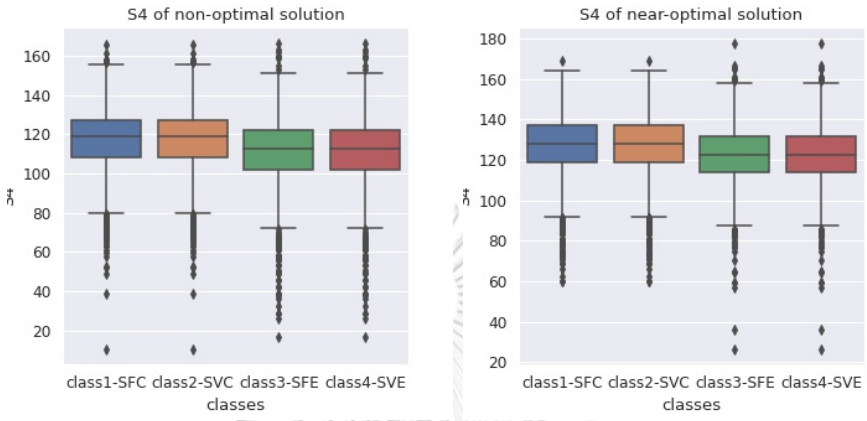


(c) S3 of non-optimal solution in BIG

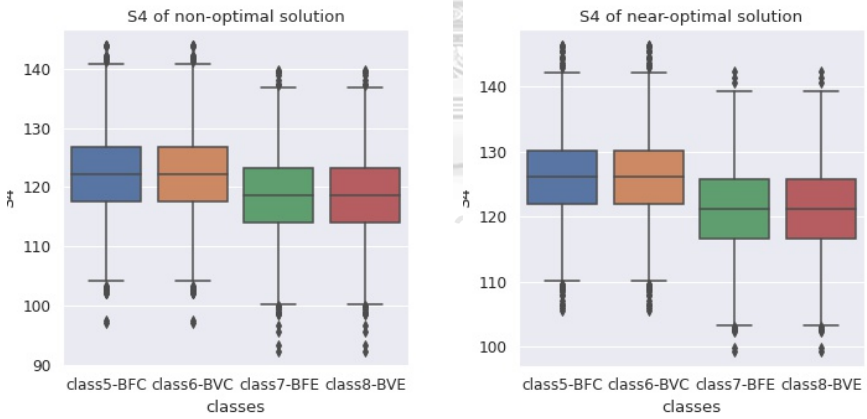


(d) S3 of near-optimal solution in BIG

Figure 16: Boxplots of S3 by size of instance and label of solution

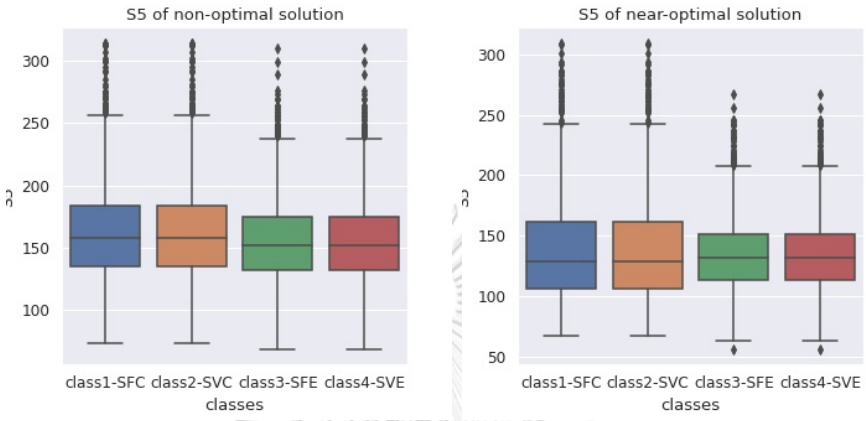


(a) S4 of non-optimal solution in SMALL **(b)** S4 of near-optimal solution in SMALL

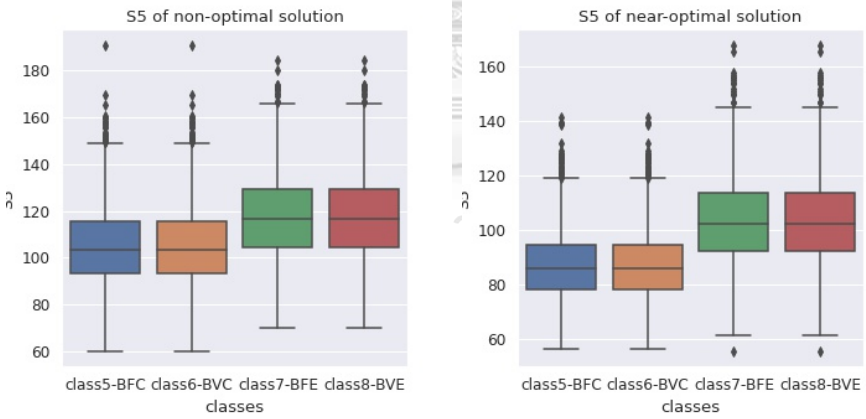


(c) S4 of non-optimal solution in BIG **(d)** S4 of near-optimal solution in BIG

Figure 17: Boxplots of S4 by size of instance and label of solution

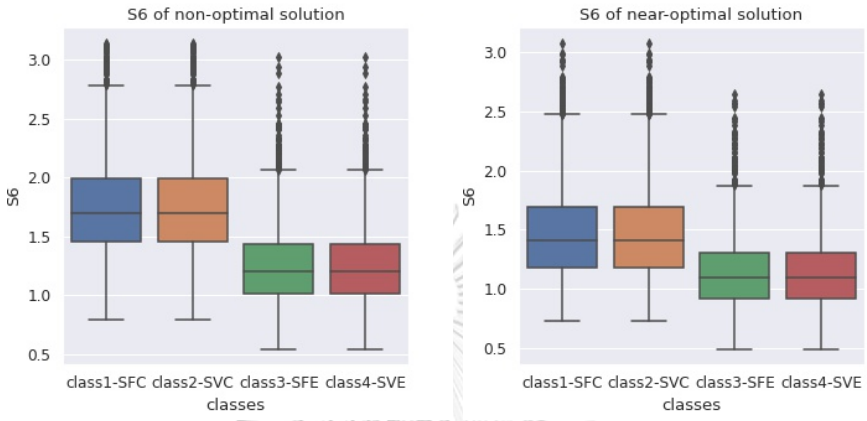


(a) S5 of non-optimal solution in SMALL **(b)** S5 of near-optimal solution in SMALL



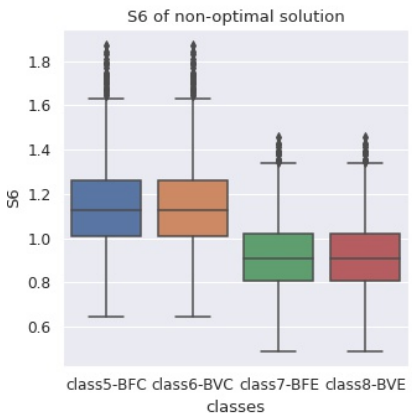
(c) S5 of non-optimal solution in BIG **(d)** S5 of near-optimal solution in BIG

Figure 18: Boxplots of S5 by size of instance and label of solution

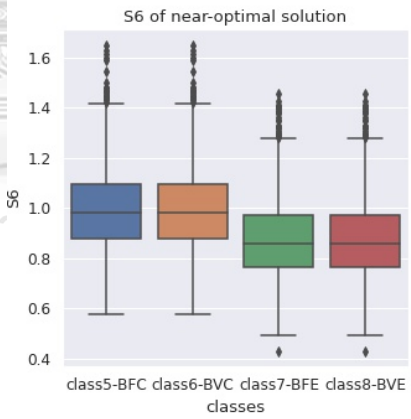


(a) S6 of non-optimal solution in SMALL

(b) S6 of near-optimal solution in SMALL

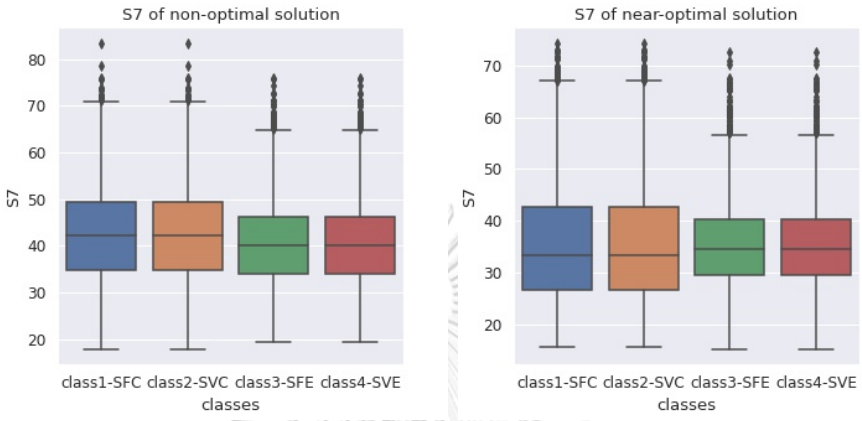


(c) S6 of non-optimal solution in BIG

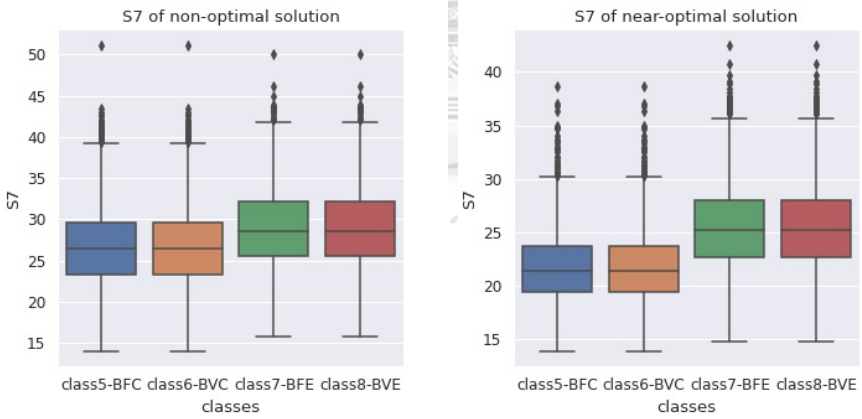


(d) S6 of near-optimal solution in BIG

Figure 19: Boxplots of S6 by size of instance and label of solution

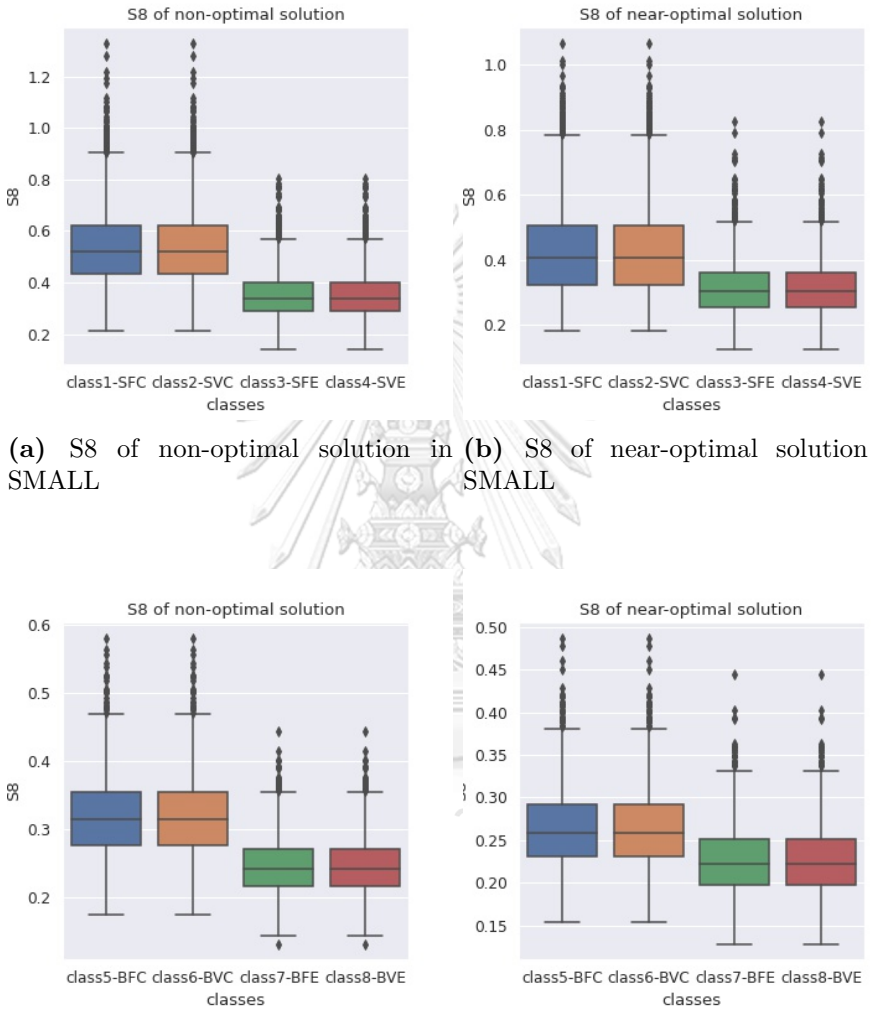


(a) S7 of non-optimal solution in SMALL **(b)** S7 of near-optimal solution in SMALL



(c) S7 of non-optimal solution in BIG **(d)** S7 of near-optimal solution in BIG

Figure 20: Boxplots of S7 by size of instance and label of solution



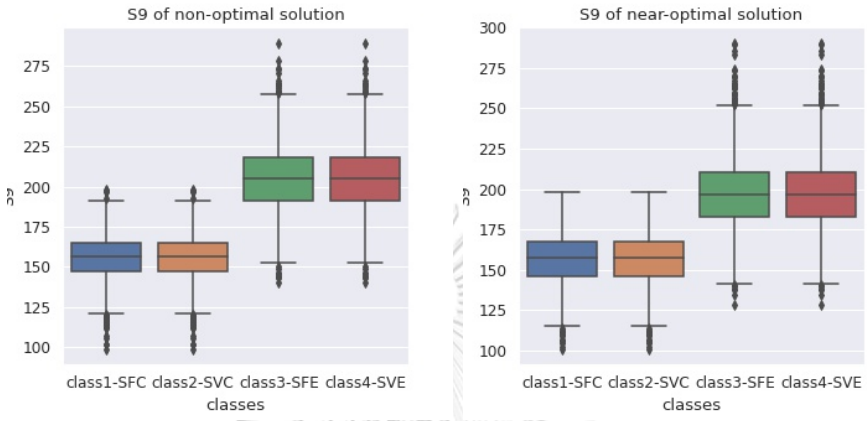
(a) S8 of non-optimal solution in SMALL

(b) S8 of near-optimal solution in SMALL

(c) S8 of non-optimal solution in BIG

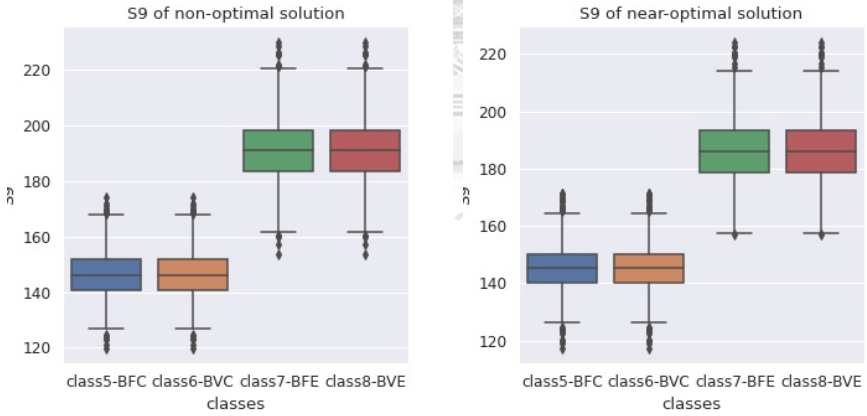
(d) S8 of near-optimal solution in BIG

Figure 21: Boxplots of S8 by size of instance and label of solution



(a) S9 of non-optimal solution in SMALL

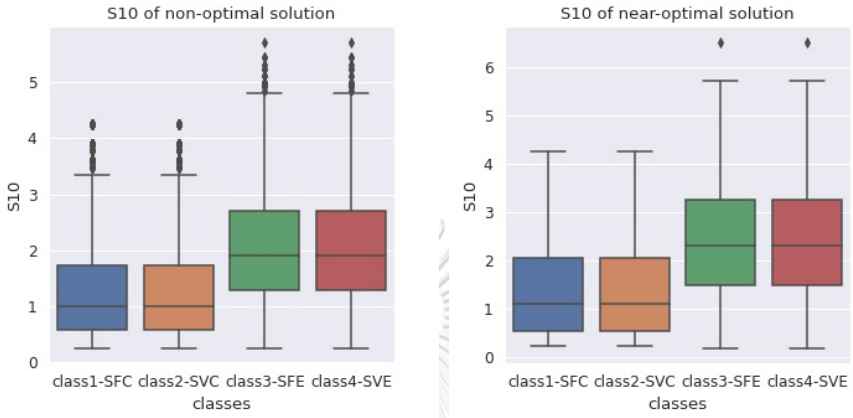
(b) S9 of near-optimal solution in SMALL



(c) S9 of non-optimal solution in BIG

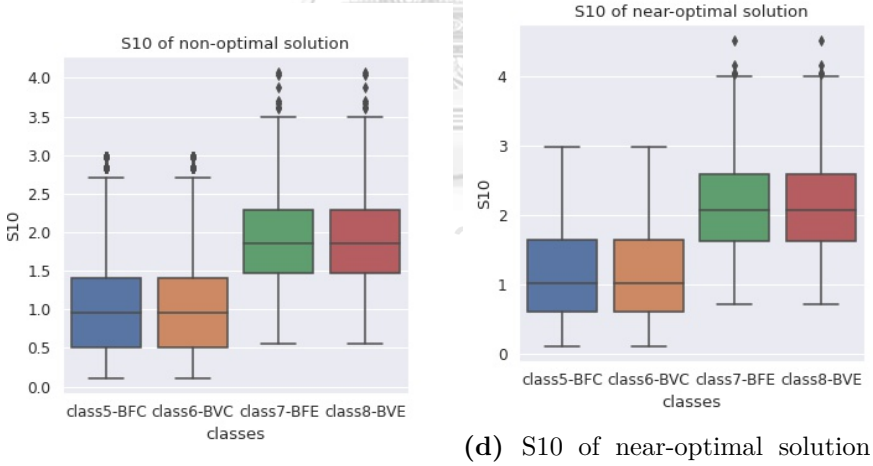
(d) S9 of near-optimal solution in BIG

Figure 22: Boxplots of S9 by size of instance and label of solution



(a) S10 of non-optimal solution in SMALL

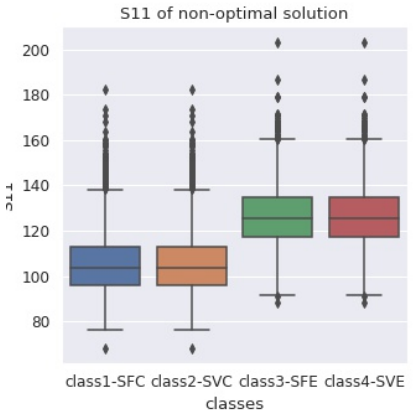
(b) S10 of near-optimal solution in SMALL



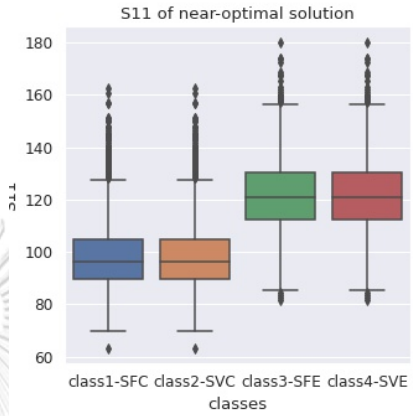
(c) S10 of non-optimal solution in BIG

(d) S10 of near-optimal solution in BIG

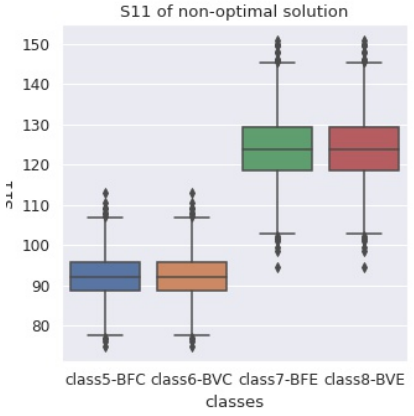
Figure 23: Boxplots of S10 by size of instance and label of solution



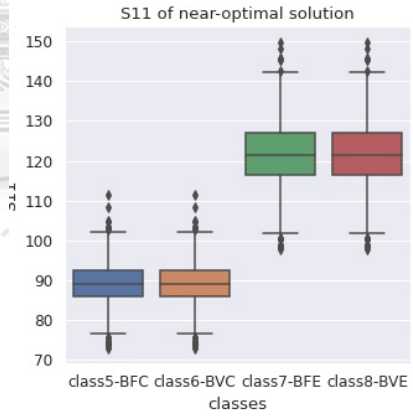
(a) S11 of non-optimal solution in SMALL



(b) S11 of near-optimal solution in SMALL

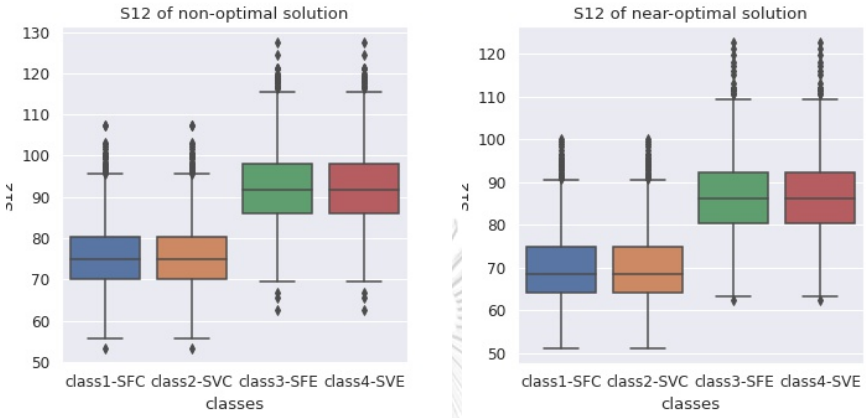


(c) S11 of non-optimal solution in BIG



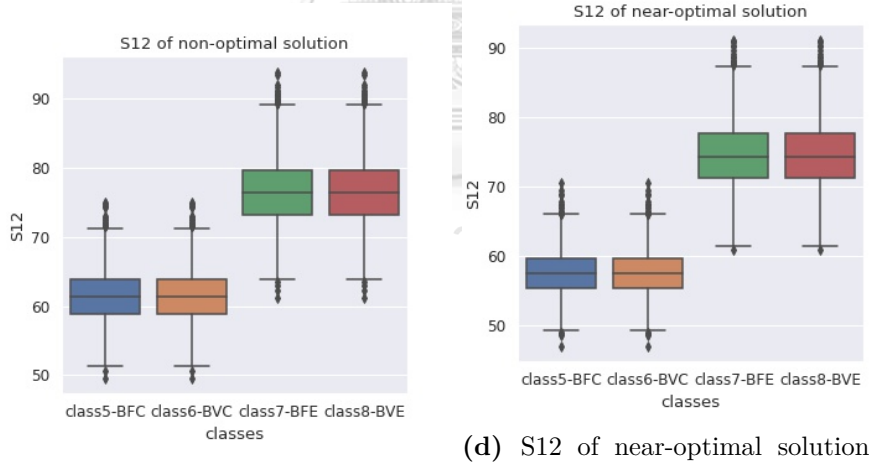
(d) S11 of near-optimal solution in BIG

Figure 24: Boxplots of S11 by size of instance and label of solution



(a) S12 of non-optimal solution in SMALL

(b) S12 of near-optimal solution in SMALL



(c) S12 of non-optimal solution in BIG

(d) S12 of near-optimal solution in BIG

Figure 25: Boxplots of S12 by size of instance and label of solution

BIOGRAPHY

Name MR. Piyabut Inbunsong

Date of Birth March 5, 1997

Place of Birth Lampang, Thailand

Educations B.Sc. (Mathematics), Chiang Mai University, 2018

Scholarships Development and Promotion of Science and Technology Talents Project (DPST), Institute of the Promotion of Teaching Science and Technology (IPST)

Publications

- Piyabut, I & Boonyarit, I 2022, 'Identifying the most distinctive characteristics of a good solution for non-Euclidean CVRP using statistical learning model', Operations Research Network 2022 Conference, Songkla, Thailand, 16-18 March 2022, pp. 176-182.