RANDOM SEQUENCES IN VEHICLE ROUTING PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET EMİN GÜLŞEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

SEPTEMBER 2022

Approval of the thesis:

RANDOM SEQUENCES IN VEHICLE ROUTING PROBLEM

submitted by **MEHMET EMİN GÜLŞEN** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Selçuk Kestel
Dean, Graduate School of **Applied Mathematics**

Assoc. Prof. Dr. Oğuz Yayla
Head of Department, **Cryptography**

Assoc. Prof. Dr. Oğuz Yayla
Supervisor, **Cryptography, METU**

Examining Committee Members:

Assoc. Prof. Dr. Ali Doğanaksoy
Mathematics Department, METU

Assoc. Prof. Dr. Oğuz Yayla
Institute of Applied Mathematics, METU

Assist. Prof. Dr. Oumout Chouseinoglou
Information Systems and Technologies, Bilkent Uni.

Assist. Prof. Dr. Barbaros Yet
Department of Cognitive Science, METU

Assoc. Prof. Dr. Fatih Sulak
Mathematics Department, Atılım University

Date:

ABSTRACT

RANDOM SEQUENCES IN VEHICLE ROUTING PROBLEM

Gülşen, Mehmet Emin
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Oğuz Yayla

September 2022, 47 pages

Vehicle Routing Problem (VRP) is a classical combinatorial optimization and integer programming problem. The goal of VRP is to find the optimal set of routes to given set of destination points with a fleet of vehicles. In this thesis, we have focused on the a variant of VRP which is Capacitated Vehicle Routing Problem (CVRP) and present two different combination of heuristic algorithms with random projection clustering technique and also provide comparison of random number generators on Monte Carlo Simulation to solve CVRP instances with combination of random projection clustering algorithm. In the first part, we show that the random projection clustering approach improves the cost compared to the core heuristic solution. In the second part, we study the choice of the random number generators on simulation based techniques on CVRP. A Monte Carlo simulation based Clarke and Wright's Savings (CWS) algorithm implemented and experiments conducted with five different random number generators. Results have shown the choice of random number generators affects the performance of the simulation.

Keywords: Monte Carlo simulation, Random number generators, Capacitated vehicle routing problem, Random projection clustering

ÖZ

ARAÇ ROTALAMA PROBLEMİNDE RASTGELE DİZİLER

Gülşen, Mehmet Emin
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Oğuz Yayla

Eylül 2022, 47 sayfa

Araç Rotalama Problemi, klasik bir kombinatoryal optimizasyon ve tamsayı programlama problemidir. Araç Rotalama Probleminde amaç, bir araç filosu ile belirli varış noktalarına en uygun rota grubunu oluşturmaktır. Bu tezde, Araç Rotalama Probleminin bir çeşidi olan Kapasiteli Araç Rotalama Problemi üzerinde duruldu ve iki farklı sezgisel algoritmanın, rastgele izdüşümsel ağaç yapısını kullanan bir kümeleme tekniği ile birleşimini sunduk ve ayrıca Kapasiteli Araç Rotalama Problemini çözmek için Monte Carlo Simülasyonu üzerinde rastgele sayı üreteçlerinin karşılaştırmasını yaptık. Çalışmanın ilk kısmında, rastgele izdüşümsel ağaç yapısının, araç rotalama problemlerinde kullanılan sezgisel algoritmalarla birleşimi sonucu ortaya çıkan iyileştirmeler gösterildi, İkinci kısımda ise rastgele sayı üreteçlerinin, Monte Carlo simülasyonu ve rastgele izdüşümsel kümelendirme algoritmaları kullanılarak geliştirilen yöntem üzerindeki etkileri üzerine ilişkisi sunuldu. Bu ilişki ile ilgili deneylerin yapılabilemesi için Monte Carlo simülasyonunun, Clarke ve Wright tasarruf algoritmasıyla birleşiminden oluşan bir yöntem kullanıldı ve yöntem beş farklı rastgele sayı üreteci ve çeşitleriyle test edildi. Bu deneyler sonucunda rastgele sayı üreteçlerinin Monte Carlo simülasyonu üzerindee etkisi gözlemlendi.

Anahtar Kelimeler: Monte Carlo Simülasyonu, Rastgele sayı üreteçleri, Kapasiteli araç rotalama problemi, Rastgele izdüşüm kümelemesi

ACKNOWLEDGMENTS

First and the biggest appreciation needs to be done to my supervisor Assoc. Prof. Dr. Oğuz Yayla for his motivation, enthusiasm and guidance. I am aware that his guidance, kindness and patience during this research was privilege for me. I would also like to thank the thesis jury members, Asst. Prof. Dr. Oumout Chouseinoglou, Asst. Prof. Dr. Barbaros Yet and Assoc. Prof. Dr. Fatih Sulak for their generosity and time in reading this thesis.

Special thanks to my colleagues, Ahmet Serdar Karadeniz and Mehmet Fatih Karadeniz for their encouragement and supports during this period. I would like to thank to my close friends especially Mustafa Çağrı Güven, Mustafa Yağız Demircioğlu and Cengizhan Ünsal for their presence and support. I also would like to express my heartfelt thanks to Gökçe Tidim who have supported me throughout entire process, both by keeping me harmonious and helping me putting pieces together.

Most importantly, I am grateful for my family's unconditional, unequivocal, and loving support.

TABLE OF CONTENTS

ABSTRACT vii

ÖZ ix

ACKNOWLEDGMENTS xi

TABLE OF CONTENTS xiii

LIST OF TABLES xvii

LIST OF FIGURES xviii

LIST OF ABBREVIATIONS xix

CHAPTERS

1 INTRODUCTION 1

2 VEHICLE ROUTING PROBLEM 3

2.1 The Traveling Salesman Problem 3

2.2 The Vehicle Routing Problem 5

2.2.1 Problem Formulation 5

2.3 Variants of Vehicle Routing Problems 6

2.3.1 Capacitated Vehicle Routing Problem (CVRP) . . 6

2.3.2 Vehicle Routing Problem with Time Windows . . . 8

2.3.3 Open Vehicle Routing Problem (OVRP) 8

2.3.4 Multi Depot Vehicle Routing Problem (MDVRP) . 9

2.3.5 Stochastic Vehicle Routing Problem (SVRP) . . . 10

2.4 Solution Methods 10

2.4.1 Nearest Neighbor Insertion (NNI) 11

2.4.2 Clark & Wright's Savings (CWS) Algorithm . . . 12

2.4.3 Monte Carlo Method 14

2.4.3.1 Monte Carlo Tree Search (MCTS) . . 14

2.4.4 Monte Carlo Simulation Applied on CWS 15

2.4.4.1 Binary-CWS 15

2.4.4.2 Binary-CWS-MCS 16

2.4.5 Clustering Approach on Vehicle Routing Problem . 17

2.4.5.1 Random Projection Trees to Cluster the CVRP Instances 21

3 RANDOM NUMBER GENERATORS 25

3.1 Linear Congruential Generator 25

3.1.1 LCG with Shift Transformation (LCGS) 26

3.2 Multiple Recursive Generator (MRG) 27

3.3 Inversive Congruential Generator 27

3.4 Mersenne Twister Pseudo Random Number Generator 28

3.5 Permuted Congruential Generator (PCG) 30

4 RESULTS 31

4.1 CVRP Benchmark Instances 31

4.2 Applying Clustering with Heuristic Methods 35

4.2.1 Applying Topological Clustering with Heuristic Methods 36

4.2.2 Applying Random Projection Clustering with Heuristic Methods 38

4.3 Comparison of Pseudo Random Generators on Binary-CWS-MCS 39

5 CONCLUSION 43

REFERENCES 45

LIST OF TABLES

Table 2.1 Distance Matrix 12

Table 2.2 Saving Matrix 12

Table 3.1 LCG Parameters 26

Table 3.2 MRG Parameters with Mersenne-Prime Modulus 27

Table 3.3 MRG Parameters with Sophie-Germain Prime Modulus 27

Table 3.4 Mersenne-Twister Parameters 29

Table 4.1 Results of ToMaTo with CWS - Golden Set 36

Table 4.2 Results of ToMaTo with CWS - M Set 37

Table 4.3 Results of ToMaTo with CWS - E Set 37

Table 4.4 Results of ToMaTo with CWS - Tai Set 38

Table 4.5 Comparison of NNI, CWS and RPC-NNI, RPC-CWS 39

Table 4.6 Comparison of Random Number Generators 41

LIST OF FIGURES

Figure 2.1 A solution for an instance of TSP 4

Figure 2.2 A solution visualization for an instance of Vehicle Routing Problem 5

Figure 2.3 A solution visualization for an instance of Open Vehicle Routing Problem 9

Figure 2.4 A solution visualization for an instance of MDVRP 10

Figure 2.5 Steps of Monte Carlo Tree Search 15

Figure 2.6 Probability Tests of E-n51-k5 instance 18

Figure 2.7 Probability Tests of E-n22-k4 instance 18

Figure 2.8 An example of connectivity-based clustering 19

Figure 2.9 An example of centroid-based clustering 20

Figure 2.10 An example of distribution-based clustering 20

Figure 2.11 An example of density-based clustering 21

Figure 3.1 Permuted Congruential Generator 30

Figure 4.1 E-n76-k7 Instance 32

Figure 4.2 M-n151-k12 Instance 32

Figure 4.3 Tai385 Instance 33

Figure 4.4 G9 Instance 34

Figure 4.5 X-n219-k73 Instance 34

LIST OF ABBREVIATIONS

CVRP	Capacitated Vehicle Routing Problem
CWS	Clarke and Wright's Savings Algorithm
DBSCAN	Density-based spatial clustering of applications with noise
GFSR	Generalised Feedback Shift Register
HCA	Hierarchical Clustering Analysis
ICG	Inversive Congruential Generator
LCG	Linear Congruential Generator
LCGS	Linear Congruential Generator with Shift Transformation
MCTS	Monte Carlo Tree Search
MDVRP	Multi-depot Vehicle Routing Problem
MRG	Multiple Recursive Generator
MRGS	Multiple Recursive Generator with Shift Transformation
MT	Mersenne Twister Pseudo Random Number Generator
NNI	Nearest Neighbor Insertion Algorithm
OVRP	Open Vehicle Routing Problem
PCG	Permuted Congruential Generator
RNG	Random Number Generator
RPC-NNI	Random Projection Clustering combination with Nearest Neighbor Insertion
RPC-CWS	Random Projection Clustering combination with Clarke and Wright's Savings Algorithm
SVRP	Stochastic Vehicle Routing Problem
TSP	Traveling Salesman Problem
TGFSR	Twisted. Generalised Feedback Shift Register
ToMaTo	Topological Mode Analysis Tool
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

CHAPTER 1

INTRODUCTION

In order to improve the delivery services of goods to customers with different demands, distribution centers must arrange optimal vehicle routes to ensure the lowest transportation and distribution costs. Planning optimal routes for the distribution of goods to customers can generate important savings for companies; several benefits can be obtained if the vehicle routing problem (VRP) is solved. Consequently, VRP is an important task in many private and public corporations. VRP has became the one of the mostly studied problems in the field of combinatorial optimization. The first example of the problem was introduced by Dantzig [10] in 1959 and is applied to the design of optimal routes, which seek to serve a number of customers with a fleet of vehicles. The objective of this problem is to determine the optimal route to serve multiple clients, using a group of vehicles to minimize the overall transportation cost.

VRP and its variants are classified as NP-hard problems. Because of this reason, for finding an optimal solution, many factors are has to be considered with many possibilities of permutation and combinations. VRP becomes more complex as constraints and number of customers increase. There are multiple variations of VRP with different type of constraints such as Capacitated VRP (CVRP), Multi-depot VRP (MDVRP), Periodic VRP (PVRP), Stochastic VRP (SVRP), and VRP with Time Windows (VRPTW), among others [9].

The aim of this study is to both propose a hybrid approach to solve the CVRP instances and make a comparison based on pseudo random number generators in the use case of Monte Carlo Simulation for generating routes. Since the CVRP is also NP-Hard and the problem instances contains high number of customer nodes, the study is built around the cluster-first-route-second approach. For the purpose of cluster part of the hybrid approach we have tested different type of clustering algorithms with combination of Nearest Neighbor Insertion (NNI) and Clarke and Wright's Savings (CWS) algorithms [8]. The random projection tree structure is adopted as the clustering part of the hybrid approach and the results of experiments show that the employing clustering as the pre-process for solving CVRP instances with heuristics can improve the solutions.

The contribution of the study can be examined in two perspective, first is to provide results about the importance of random number generator choice on solving VRP and second is to proposing a cluster-first-route-second approach for solving VRP instances. In order to test the performance of pseudo random generators on Monte Carlo simulation for solving CVRP instances, the Binary-CWS-MCS algorithm from the study [29] adopted with random projection clustering method. Linear Congruential Generator (LCG), Multiple Recursive Generator (MRG), Inversive Congruential Generator (ICG), Permuted Congruential Generator (PCG) and Mersenne-Twister (MT) pseudo random generator are employed for a comparison of their cost performances. The results show that the PCG and MT pseudo random number generators are better performing than the others. In addition, using the shift transformation with LCG and MRG can increase the cost performance on Monte Carlo Simulation applications. The implementation of the method can be seen on the following repository [19]:

`https://github.com/mehmetemingulsen/rpc-binary-cws-mcs.`

The outline of the thesis is as follows. In the next chapter, detailed information and problem formulation are presented with the solution methods in the literature. After giving information about the problem and the methods, the random number generators that used in this study are briefly introduced. Finally, the results about the both type of experiments are given in details.

CHAPTER 2

VEHICLE ROUTING PROBLEM

The objective of this section is to briefly present the problem studied in this thesis, Capacitated Vehicle Routing Problem (CVRP). Traveling Salesman Problem, Vehicle Routing Problem (VRP) and the variants of the VRP will be introduced to provide a deeper knowledge about the core of the problem. Each problem will be introduced in words and mathematical definitions. Firstly, TSP will be examined to give a brief introduction about the evaluation of the VRP and CVRP.

2.1 The Traveling Salesman Problem

Traveling Salesman Problem (TSP) is a well-known NP-Hard problem in computational mathematics. Objective of the TSP can be described as finding the the cheapest way starting from the depot to all destination points and return to depot with a given finite set of destination points with the known cost of travel between each destination points [1]. The problem domain of the TSP can be introduced as an undirected weighted graph where the destination points are the vertices and the paths between them are the edges of the graph. Let $G = (V, E)$ be an undirected graph with the set of vertices V and set of edges $E = \{(x, y)|x, y \in E\}$. Each edge $e \in E$ has a cost of c_e. Let H be the set of all Hamiltonian cycles, the objective of the TSP is to find the cycle $h \in H$ with the sum of costs c_e is minimized. A simple visual example of TSP can be seen in Figure 2.1.

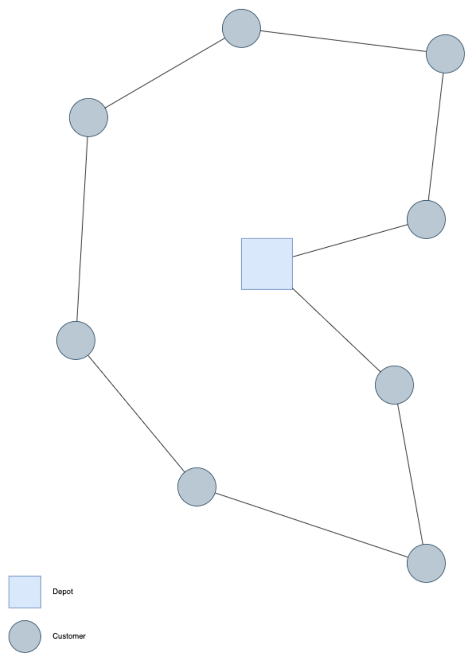


Figure 2.1: A solution for an instance of TSP

Problem Formulation

TSP can be expressed as integer linear programming formulation with letting the destination points in the problem start from 1 to n and let the c_{ij} be the distance between i and j [26].

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{i \neq j, i=1}^{n} x_{ij} = 1$$

$$\sum_{j \neq i, j=1}^{n} x_{ij} = 1$$

$$x_{ij} = \begin{cases} 1 & \text{if the path goes from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

2.2 The Vehicle Routing Problem

The Vehicle Routing Problem was first introduced by Dantzig as "Truck Dispatching Problem" with the concern of generation of expanded form of Traveling Salesman Problem (TSP) in 1959 [12]. The Truck Dispatching Problem was concerned to find the optimum routing of a fleet of gasoline delivery trucks between a bulk terminal and a number of stations supplied by terminal. The aim in the vehicle routing problem is to establish the optimized routes for serving the set of customers by the given fleet of vehicles. A simple visual example of a VRP instance is given in Figure 2.2. A VRP instance also can described as an undirected weighted graph same as TSP $G = (V, E)$. The customers in the VRP are the vertices in the graph and the paths between them are the edges of the graph G [10].

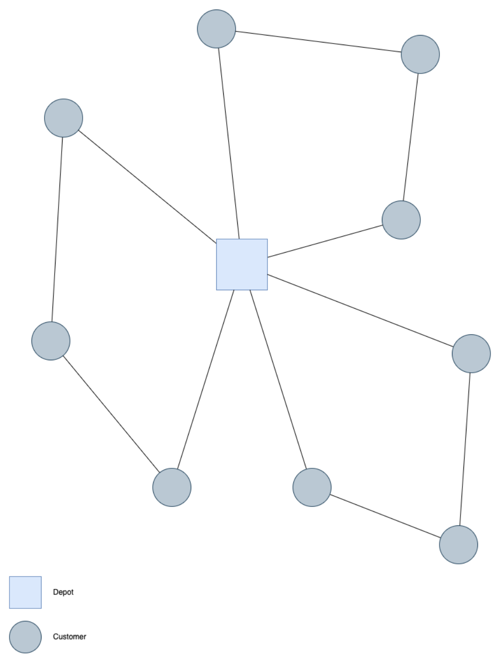


Figure 2.2: A solution visualization for an instance of Vehicle Routing Problem

2.2.1 Problem Formulation

VRP also can be described as integer linear programming formulation. Let the destination points in the problem start from 0 to n and let the 0 to be the depot of the problem. Let c_{ij} is the distance between point i to j, K is the total number of vehicles in the problem instance and each vehicle expressed as k. Assigning a binary value to k makes each edge appears once while the capacity of the vehicles can not be exceeded as given in Equation 2.1.

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

$$\sum_{k=1}^{K} \sum_{i \neq j, i=1}^{n} x_{ijk} = 1, \forall i \in V - \{0\}$$

$$x_{ijk} = \begin{cases} 1 & \text{if the vehicle } k \text{ goes from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

2.3 Variants of Vehicle Routing Problems

Vehicle Routing Problem has many variants due to the specific requirements in the field of transportation and logistics. Since the problem can be extended with other constraints such as time, capacity and demands. The each VRP variant also can be combined and used for generating new variants with more complex structures. In this section, Capacitated Vehicle Routing Problem (CVRP), Vehicle Routing Problem with Time Windows (VRPTW), Multi-depot Vehicle Routing Problem (MVRP) and Open Vehicle Routing Problem (OVRP) will be introduced to give better understand about the topic.

2.3.1 Capacitated Vehicle Routing Problem (CVRP)

The Capacitated Vehicle Routing Problem is one of the most common and known variant of VRP. The difference between the VRP and CVRP is the vehicles used in the CVRP has a capacity unlike VRP. There are two sub-variants of CVRP which are changing with respect to the capacity variability of the problem. The first variant is the Homogeneous Capacitated Vehicle Routing Problem, the vehicle capacity in this variant of the problem is same for all the vehicles in the instance and the other variant is the Heterogeneous Capacitated Vehicle Routing Problem, the vehicles in this instance may have different capacities than each other, this variant also called Mixed Capacity Vehicle Routing Problem. The formulation of CVRP with homogeneous capacity constraint and heterogeneous capacity variants are given below. The destination points in the problem start from 0 to n and node with index 0 is set to be the depot of the problem. The c_{ij} value presents the distance between point i to j, K represents the total number of vehicles in the problem instance and each vehicle expressed as k and each vehicle has capacity of Q for homogeneous variant and the Q_i for heterogeneous variant. The q_i value in the instance represents the demand value of each destination points. As given on both of the equations below, the x_{ijk} values represents edges from i to j assigned to vehicle k.

Homogeneous Capacited Vehicle Routing Problem Formulation:

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{k=1}^{K} \sum_{i \neq j, i=1}^{n} x_{ijk} = 1, \forall i \in V - \{0\}$$

$$\sum_{i=1}^{n} q_i \sum_{i=1}^{n} x_{ijk} \leq Q, \forall k \in K$$

$$x_{ijk} = \begin{cases} 1 & \text{if the vehicle } k \text{ goes from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

Heterogeneous Capacited Vehicle Routing Problem Formulation:

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij}$$
$$\sum_{k=1}^{K} \sum_{i \neq j, i=1}^{n} x_{ijk} = 1, \forall i \in V - \{0\}$$
$$\sum_{i=1}^{n} q_i \sum_{i=1}^{n} x_{ijk} \leq Q_i, \forall k \in K$$

$$x_{ijk} = \begin{cases} 1 & \text{if the vehicle } k \text{ goes from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

2.3.2 Vehicle Routing Problem with Time Windows

The most prominent constraint of this variant is its time window for delivering a product or demand to the destination on the specified range of time. The goal on solving VRPTW is to generate the route list with minimum cost without violating the time windows and vehicle capacities [21]. Since, the problem require the vehicles serve at a specific range of time, the vehicles may arrive early or late to the destination points. This possibility generates two different situations for VRPTW variant which is the acceptance of early or late delivery with some form of penalty. The models which accepts the early or late delivery with penalty called as "soft" VRPTW and the other type of models called as "hard" VRPTW. Most of the studies has been made on "hard" variant of the problem.

2.3.3 Open Vehicle Routing Problem (OVRP)

Open Vehicle Routing Problem also contains the same constraints with the traditional VRP but the main difference is the OVRP does not have the constraint of the returning to the depot. This implies to the routes to be non-cyclic and the space of solutions in this variant to be increased. Today, the OVRP is encountered in practice in the home delivery of packages and newspapers. Contractors who are not employees of the delivery company use their own vehicles and do not return to the depot. A visual example of an OVRP instance can be seen in Figure 2.3. Detailed information about OVRP variants can be seen on [23].

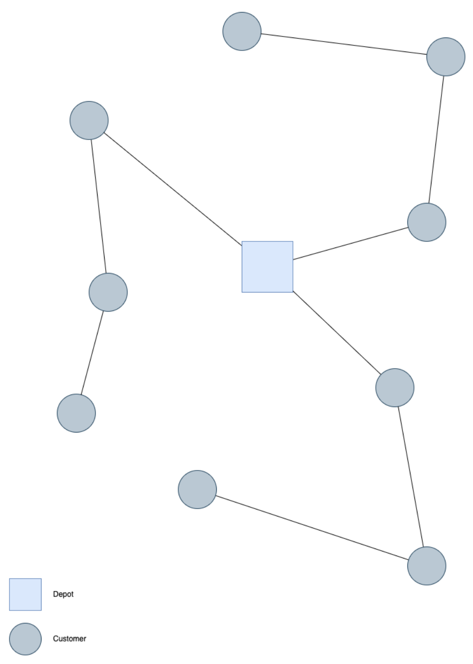


Figure 2.3: A solution visualization for an instance of Open Vehicle Routing Problem

2.3.4 Multi Depot Vehicle Routing Problem (MDVRP)

The case of MDVRP is posed if a company may have more than one depots from which it can serve its customers. If the customers are clustered around depots, then a number of independent VRPs or TSPs can be performed to serve customers. A visual example of a solution of an MDVRP instance can be seen in Figure 2.4. There can be constraints such as the total number of vehicles that has left a depot must be equal to the total number of vehicles that arrive to that depot at the end of the procedure. Furthermore, there can be a constraint for each vehicle assigned to a depot and they should be on that assigned depots at a given time window.

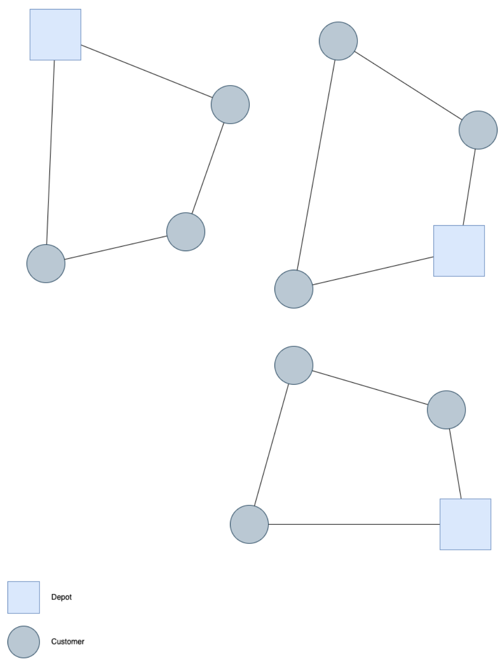


Figure 2.4: A solution visualization for an instance of MDVRP

2.3.5 Stochastic Vehicle Routing Problem (SVRP)

Stochastic Vehicle Routing Problem has became an popular research topic due to increasing service demands on supply chains with uncertainty in real life conditions. Stochasticity in VRP can be provided by expressing the aspects of the problem instance with probability. The SVRP variant is a generalized definition for all the variants that includes stochastic constraints. One of the SVRP variant includes a probability of presence of the customer in problem which is called as CVRP with stochastic customers (CVRPSCD), another variant includes stochasticity to problem by concerning changing demands for each customers that variant is called as CVRP with stochastic demands (CVRPSD). The time windows that the customers have to be served also can be stochastic and this variant called as the CVRP with stochastic time windows (CVRPTW) [16].

2.4 Solution Methods

The CVRP is mainly focused in this study. There are exact methods, heuristic methods and meta heuristic methods in the literature to create feasible solutions for the

CVRP. In this section, the mostly known two heuristic algorithms, will be examined and after, the binary version of CWS Algorithm with Monte Carlo Simulation (MCS) technique will be presented as Binary-CWS-MCS. Finally, mostly known and used clustering will be represented with random projection trees.

2.4.1 Nearest Neighbor Insertion (NNI)

Nearest Neighbor Insertion algorithm is one of mostly known heuristic algorithms for solving TSP and VRP by constructing a route list with appending the nearest node to the route. The process is initialized with randomly chosen node from the problem set and continues until all the vehicle capacities got consumed or all the demands are satisfied. The pseudo code of the NNI can be seen in Algorithm 1. NNI algorithm requires the number of vehicles, capacity of the vehicles and the node list to conclude on a feasible solution.

Algorithm 1 Nearest Neighbor Insertion Algorithm

```
Require: Demand list $D$, Number of vehicles $k$, Vehicle Capacity (C), List of Unvisited Nodes (N)
1: RouteList = Empty list for routes
2: $v = 0$
3: while $v < k$ do
4:     route = []
5:     capacity = 0
6:     $j$= Random Unvisited Node
7:     while capacity + D[j] $\leq$ C do
8:         Append j to route
9:         capacity = capacity + D[j]
10:        j = Nearest Unvisited Node
11:    end while
12: end while
13: return RouteList
```

2.4.2 Clark & Wright's Savings (CWS) Algorithm

The best known approach for generating feasible solutions for the single depot CVRP is Clark and Wright's Savings algorithm. The CWS algorithm is an iterative process that enables of the quick selection of near-optimum routes [8]. The algorithm requires distances to be representing in the form of an $(n + 1) \times (n + 1)$ matrix called as distance matrix, C. The entries c_{ij} of C are the distance between nodes i and j for $i, j \in 0, 1, 2, ..., n$. An example for a distance matrix is given in Table 2.1. A saving matrix S with the values s_{ij} is built using the equation, $s_{ij} = c_{0i} + c_{0j} - c_{ij}$, with the distance matrix.

Table 2.1: Distance Matrix

c_{ij}	i_0	i_1	i_2	i_3	i_4	i_5
i_0	0	28	31	20	25	34
i_1		0	21	29	26	20
i_2			0	38	20	32
i_3				0	30	27
i_4					0	25
i_5						0

If we consider Table 2.1 as the input of the CWS algorithm, the saving matrix S can be computed as Table 2.2.

Table 2.2: Saving Matrix

s_{ij}	i_1	i_2	i_3	i_4	i_5
i_1	0	38	19	27	42
i_2		0	13	36	33
i_3			0	15	27
i_4				0	34
i_5					0

The steps of CWS algorithm can be seen in Algorithm 2. While producing a feasible solution to a VRP, CWS algorithm firstly require the saving matrix which is generated using Algorithm 3. After generating the saving matrix, the CWS algorithm process starts with a for loop and continues until the savings list is empty as seen in Algorithm 4.

Algorithm 2 CWS Algorithm

```
Input: Distance matrix $C$, number of nodes $n$
Output: Route List
1: Route List = []
2: S : Generate Savings Matrix($C$,$n$)
3: for k = 0, k < length($S$), k++ do
4:     Route List : Process($S[k]$], Route List)
5: end for
```

Algorithm 3 Generate Saving Matrix

```
Input: Distance matrix $C$, number of nodes $n$
Output: Saving Matrix
1: $S$ : Saving Matrix
2: for $i$ in range $(1, n+1)$ do
3:     for $j$ in range $(1, i)$ do
4:         $S_{ij} = C_{0i} + C_{j0} - C_{ij}$
5:     end for
6: end for
7: Sort $S$ descending respect to $S_{ij}$
8: return $S$
```

CWS algorithm also provides savings with process step with assigning the nodes to routes based on the nodes given to the process function, the nodes analyzed on the basis of three criteria can be seen on Algorithm 4. If both i and j not assigned to a route, a new route initialized by connecting the nodes. The savings obtained in this function comes from the second and third criteria which are related if one of the nodes is assigned to a route, if only one of i or j is assigned to end of a route, then the function link the other node to end of that route. If both are existed at the end of two distinct routes and the capacity of a vehicle is enough then both routes merged.

Algorithm 4 Process

```
Input: $i$, $j$, routeList
Output: routeList
1: if Both $i$ and $j$ not assigned to a route then
2:     Initialize a new route with $(i, j)$
3:     Add new route to routeList
4: end if
5: if $i$ or $j$ exists at the end of a route then
6:     Link $(i, j)$ in that route
7: end if
8: if Both $i$ and $j$ exists at the end of route then
9:     Merge two routes into one route
10:    Remove old routes from routeList
11: end if
12: return routeList
```

2.4.3 Monte Carlo Method

Monte Carlo method firstly introduced by John von Neumann and Stanislaw Ulam increases the accuracy of decision making. Monte Carlo method has wide range of applications in the industry including computational physics, computer graphics to artificial intelligence for games [4]. More details about Monte Carlo simulations and the theory can be seen in the study [5].

2.4.3.1 Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) is one of the earliest methods in the field of artificial intelligence. MCTS is mostly used in NP-Hard problems since using deterministic models is not applicable. The summary of MCTS steps can be seen on Figure 2.5.

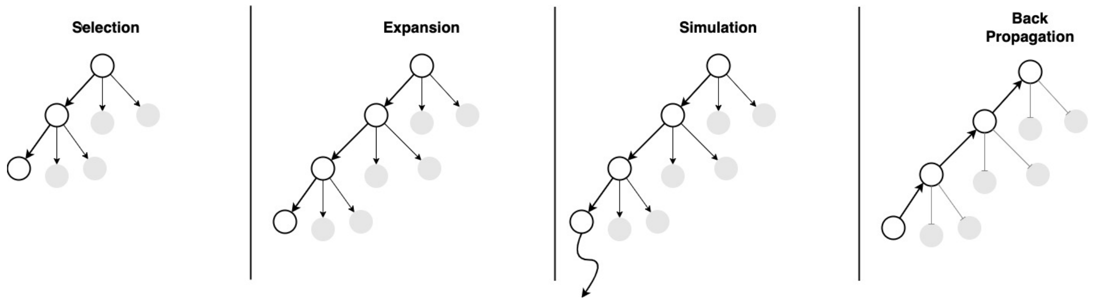


Figure 2.5: Steps of Monte Carlo Tree Search

The steps of MCTS can be examined on four different steps as Selection, Expansion, Simulation and Back propagation. The selection process basically starts from a root node and spans the tree based on an evaluation function. The selection of nodes iterates through the tree with getting the value of the evaluation function and continues with the maximum value generating option. The expansion step adds child nodes to the candidate solution and the process continues with simulation step that performs strategies until a feasible solution is obtained under the given constraints of the problem.

2.4.4 Monte Carlo Simulation Applied on CWS

In order to compare the random number generators on CVRP instances, the binary version of the Clarke and Wright's Savings algorithm with Monte Carlo simulation approach called Binary-CWS-MCS is given in [29].

2.4.4.1 Binary-CWS

The Binary-CWS algorithm can also be expressed as the selection process of the simulation. The most significant difference of the Binary-CWS from the CWS is to apply a threshold before processing the ordered node from the savings list. The binary expression arises from the probability threshold p of the process step from the CWS, see Algorithm 5. The required inputs are the savings list S, which is built such that within S the edges are ordered by descending saving value, and probability value, p. The S is the representation of the savings matrix in the form of (i, j, s_{ij}) and the p value represents the probability threshold. Process follows the same steps as CWS

Algorithm 5 Binary-CWS Algorithm

```
Input: Savings List ($S$), probability ($p$)
Output: routeList
1: Initialize the empty list routeList
2: Initialize tempList as [1,2,...,length of $S$]
3: while tempList is not empty do
4:     rand = random() mod 100
5:     for $i$ in tempList do
6:         if rand $\geq p$*100 then
7:             Process ($S$[i],routelist)
8:             Remove $i$ from templist
9:         end if
10:    end for
11: end while
12: return routeList
```

except the decision point for processing the following node from the savings list. This means that for some portion of savings list, the node will be skipped and carry on with the next node.

2.4.4.2 Binary-CWS-MCS

The Binary-CWS-MCS algorithm is built around the MCTS logic using the Binary-CWS algorithm with bulk operations. The Binary-CWS-MCS algorithm iterates through the savings matrix and for each iteration it computes the average of bulk operation for two different directions to manage the selection of next customer on the route. The simulation continues until the demand of each node is satisfied.

The critical decision for the Binary-CWS method which is presented in Algorithm 5, is to set the correct range for the p value. Because if the range for the p value set to close 0, the cost value of the Binary-CWS will converges to the cost value of CWS algorithm and reduces the improvement for the simulation, and if the range for the p value set to larger values both the time consuming of the algorithm will increase and

Algorithm 6 Binary-CWS-MCS Algorithm

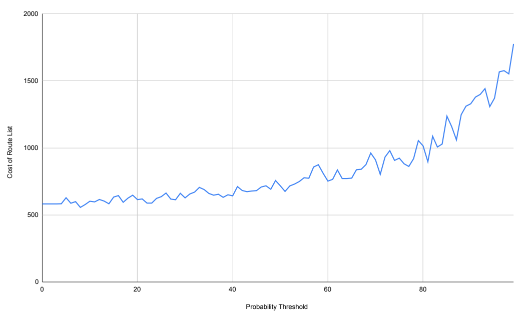
```
Input: Distance Matrix $C$, number of nodes $n$
Output: Route List
 1: Generate Savings Matrix($C$, n)
 2: list-ordered = order list descending with respect to savings $s_{ij}$
 3: route list = []
 4: while list-ordered is not empty for $s$ in range do
 5:     $t_1$: average of 1000 calls of score Binary-CWS($s$, list-ordered)
 6:     $t_2$: average of 1000 calls of score Binary-CWS($s+1$, list-ordered)
 7:     if $t_1 \geq t_2$ then
 8:         Process(list-ordered[$s$], route list)
 9:         Remove list-ordered[$s$] from list-ordered
10:     end if
11: end while
12: return route list
```

the cost value of the routes will be resulted on unsatisfactory performance compared to CWS cost/distance value.

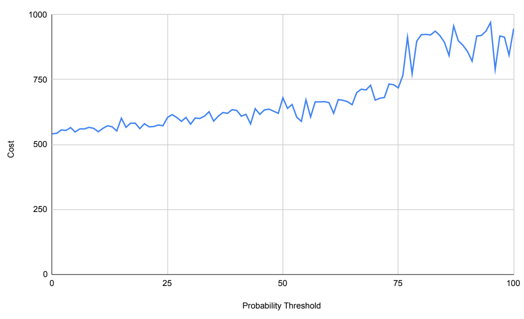
Two different CVRP instances are chosen to test the p-value in order to distinguish a better range. The E-n22-k4 (1 depot, 21 clients, 4 vehicles, vehicle capacity = 100) and E-n51-k5 (1 depot, 50 clients, 5 vehicles, vehicle capacity = 160) instances have been tested with both Binary-CWS and Binary-CWS-MCS algorithms. As it can be seen in Figure 2.6b and Figure 2.7b, the cost/distance value is mostly biased to p-value. Also, it can be seen from the figures, when p-value increases also the cost of the routes created increases. Thus, the p-value has been chosen such that $0.05 \leq p \leq 0.26$.

2.4.5 Clustering Approach on Vehicle Routing Problem

Clustering is the act of grouping comparable items into various groups, or more accurately, the partitioning of a data collection into subsets so that the data in each subset is sorted according to some distance metric. Since the CVRP instance requires to

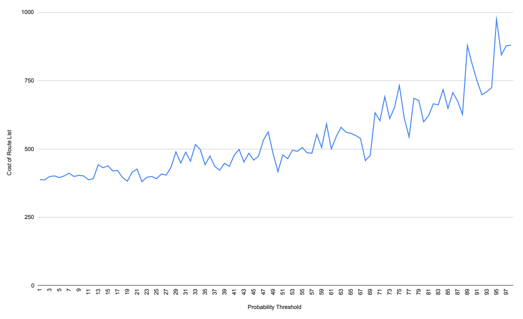


(a) Cost graph with increasing probability threshold using Binary-CWS

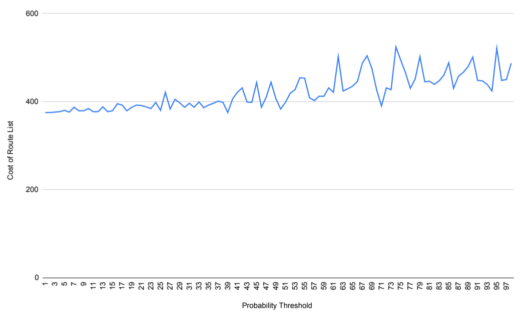


(b) Cost graph with increasing probability threshold using Binary-CWS-MCS

Figure 2.6: Probability Tests of E-n51-k5 instance



(a) Cost graph with increasing probability threshold using Binary-CWS



(b) Cost graph with increasing probability threshold using Binary-CWS-MCS

Figure 2.7: Probability Tests of E-n22-k4 instance

have more than one path or vehicles to conduct feasible solutions, the problem can be solved by clustering first and routing second approach. The most known and early development of cluster-first, route-second approach conducted by Gillett and Miller [17] and Fisher and Jaikumar [15]. Clustering the VRP and CVRP is widely using method for simplifying the instance complexity to decrease the time consume of the solution proposes. There are several types of clustering analysis techniques and algorithms in the literature used for different type of cases. Further explanations and types can be found in the study by Castro [13]. Major clustering techniques can be summarized into several techniques as follows;

- **Connectivity-based clustering** (also known as hierarchical cluster analysis or HCA) is a cluster analysis approach that aims to establish a hierarchy of clusters in data mining and statistics. The hierarchical clustering algorithms produce a dendrogram, which represents the layered grouping of items and the similarity levels at which groupings change. By cutting the dendrogram at the chosen similarity level, a clustering of the data items is produced. Clusters are combined or separated depending on a similarity metric which is chosen to optimize some requirement such as a sum of squares.

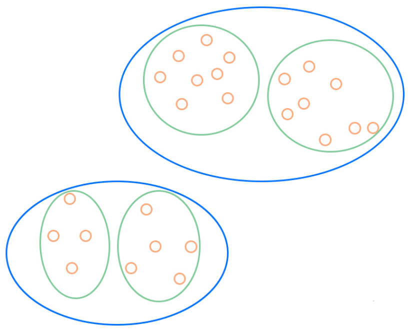


Figure 2.8: An example of connectivity-based clustering

- **Centroid-based clustering** is a vector quantization approach derived from signal processing that tries to divide n observations into k clusters, with each observation belonging to the cluster with the closest mean (cluster centers or cluster centroid), acting as a prototype of the cluster. The process of clustering can be summarized as, the first step is to finding the centroids of clusters and second step is to assign the objects or nodes to the nearest cluster center with given distance functions. A visual example of a centroid-based clustering can be seen in Figure 2.9.

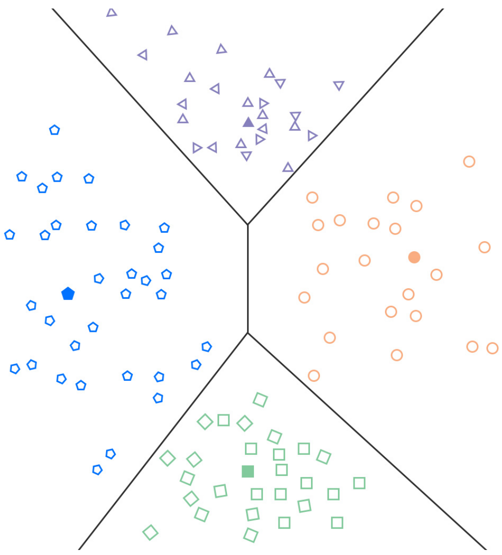


Figure 2.9: An example of centroid-based clustering

- **Distribution-based clustering** method considers that data is formed of distributions, and that as a node departs from the distribution's center, the likelihood that a point belongs to the distribution decreases. The probability decrease is illustrated by the bands. If the distribution of data is not known, different approach should be chosen.

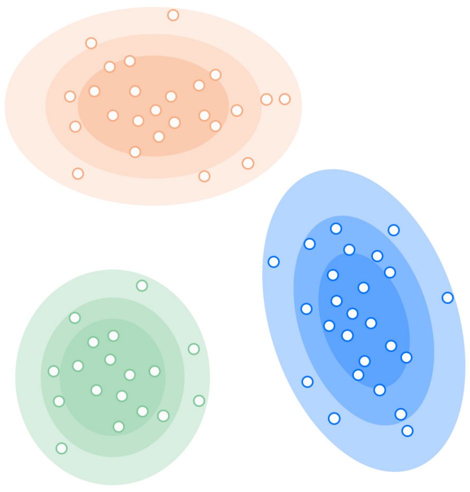


Figure 2.10: An example of distribution-based clustering

- **Density-based clustering** defines areas of higher density as clusters and extract the low density areas or nodes as noise in data. The idea behind this type of clustering method is that in a given radius of a node data set required to contain nodes or points.

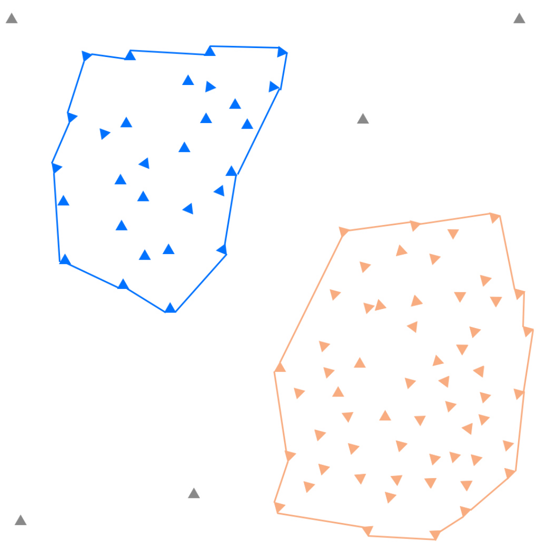
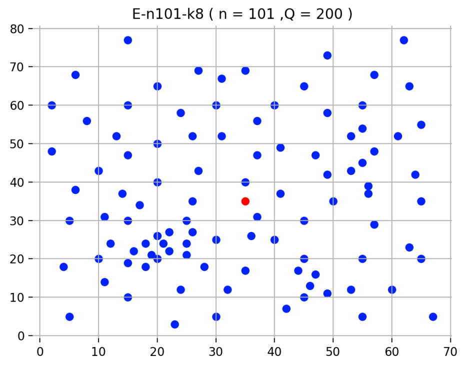


Figure 2.11: An example of density-based clustering

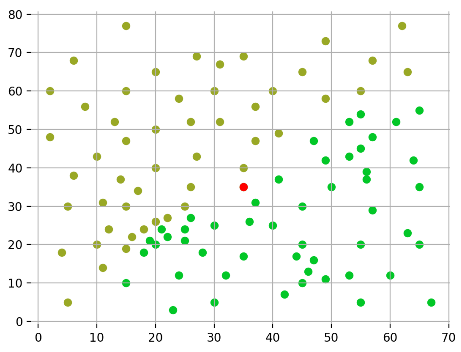
Clustering analysis for a given data set or a problem also can be achieved by other methods like machine learning or statistical distribution fit tests. Methods to divide a given data set can be expanded by any other evaluations based on the problem and the constraints. There are also methods that are seeking to find the similarity between data points to divide the data into similar sets like random projection based methods.

2.4.5.1 Random Projection Trees to Cluster the CVRP Instances

Random projection trees used mostly to reduce the dimensionality on big data applications and for use cases of k-nearest neighbor search on the data sets that contain higher number of columns for each entity.



(a) E-n101-k8 CVRP Instance


(b) Clustered E-n101-k8 CVRP Instance

Algorithm 7 MakeTree

```
Input: nodes in $x, y$ coordinates (points), threshold ($k$), clusters
1: if length of points $< k$ then
2:     add points to clusters
3:     return
4: end if
5: dotProductList = []
6: vector = GenerateRandomVector()
7: for i in range length of points do
8:     val = dot(vector,points[i])
9:     Add val to dotProductList
10: end for
11: median : Median of dotProductList
12: for i in range length of points do
13:     dst = dot(vector,points[i])
14:     if $dst \leq median$ then
15:         Add dst to the left
16:     else
17:         Add dst to the right
18:     end if
19: end for
20: MakeTree(left, $k$, clusters)
21: MakeTree(right, $k$, clusters)
```

Popular digital platforms and e-commerce systems mostly refer to random projection like methods to provide its users more relevant search systems with similar contents or products. We have tried to employ this approach for dividing CVRP instances into smaller sub problems. Mostly known use case of such a method is Annoy library which motivated to recommend similar content to the similar user types that has the potential content consumption habits [3].

A CVRP instance and clustering analysis of instance can be visualized as given in Figure 2.12a. If we consider the E-n101-k8 (1 depot, 100 clients, 8 vehicles, vehicle capacity = 200) instance given on Figure 2.12a as a sample input for the random

Algorithm 8 RandomProjectionClustering

```
Input: nodes in $x, y$ coordinates (points), threshold ($k$), number of tree ($n$)
Output: OptimalTree
 1: OptimalTree = []
 2: MakeTree(points,k,OptimalTree)
 3: MeasureOptimalTree = MeasureTree(OptimalTree)
 4: for i in range n do
 5:     Tree = MakeTree(points,k,clusters)
 6:     MeasureTree = MeasureTree(OptimalTree)
 7:     if MeasureTree $<$ OptimalTree then
 8:         OptimalTree = Tree
 9:     end if
10: end for
11: return OptimalTree
```

projection clustering, the method behind the process starts with creating a random vector and calculates dot product between each destination points and the vector. After calculation step, the median value from the dot products gotten and accepted as a threshold for clustering operation. If the median value is smaller or the point added to the left leaf, if not the point added to the right leaf. The process continues until the each leaf has at most k points. The detailed process can be seen in Algorithm 7.

The random projection tree generation in Algorithm 7 process can be executed multiple times to gather a better tree structure. In order to establish this goal, Algorithm 7 is employed as the core function of Algorithm 8. The process starts with an initial tree and measure the tree with respect to euclidean distance of each clusters, if the new generated tree has lower distance than the optimal tree replaces with the new tree.

We have combined this clustering technique with Binary-CWS-MCS. This process can be evaluated as a learning mechanism for the algorithm because by comparing each member of the problem with a randomly generated vector creates a similarity base data which can be used for clustering the points.

Algorithm 9 RPC-BinaryCWSMCS

```
Input: nodes in $x, y$ coordinates (points), threshold ($k$), number of tree ($n$)
Output: RouteList, TotalCost
1: RouteList = []
2: TotalCost = []
3: OptimalTree = RandomProjectionClustering(points,k,n)
4: for Leaf in OptimalTree do
5:     Route,Cost = BinaryCWSMCS(Leaf)
6:     TotalCost += Cost
7:     Add all routes of Route to RouteList
8: end for
9: return RouteList, TotalCost
```

CHAPTER 3

RANDOM NUMBER GENERATORS

Random numbers are utilized in many different applications, including modeling, encryption, sampling, numerical analysis and simulation based algorithms. Since, our study mostly focused on simulation based applications of random sequences, information about random number generators that employed mostly on simulations is introduced in this chapter. Five different random number generators with different parameters were employed in this study to test the quality of random number generators on Monte Carlo simulation.

3.1 Linear Congruential Generator

Linear Congruential Generator (LCG) firstly introduced by W.E.Thomson in 1958 [30]. LCG is the most widespread pseudo random number generator and easy to comprehend and implement. It is defined by a recursion as given in (3.1).

$$X_{n+1} = (a.X_n + c) \mod m \tag{3.1}$$

$m, 0 < m,$ the modulus

$a, 0 < a < m,$ the multiplier

$c, 0 \leq c < m,$ the increment

$X_0, 0 \leq X_0 < m,$ the start value, seed of the generator

As it can be seen in (3.1), the generation of random sequence starts with an initial value X_0 and iterates through the each generated value obtained with the equation. Although LCGs can generate pseudo random numbers that pass formal criteria for randomness, the output quality is highly dependent on the values of the parameters m and a. For instance, $a = 1$ and $c = 1$ generate a basic modulo-m counter with a lengthy period but is clearly non-random. The detailed analysis and the tables of known good parameters can be found in [24] and [28]. The period and parameters used for LCG employed in this study can be found in Table 3.1.

Table 3.1: LCG Parameters

LCG Parameters		
a	b	$Period$
18145460002477866997	1	2^{64}

3.1.1 LCG with Shift Transformation (LCGS)

Applying shift transformation to a random sequence is a mostly employed method to improve the randomness of sequence. If we assume that the output of a given LCG as r in binary form and the output of transformation operation as q, the operation of LCG combination with shift transformation can be expressed as following operations. The $x \gg n$ operation denotes for n bit-shift of x to the right and $x \ll n$ operations denotes for n bit-shift x to the left.

$$\begin{aligned}
t_0 &= r \\
t_1 &= t_0 \oplus (t_0 \gg 17) \\
t_2 &= t_1 \oplus (t_0 \ll 31) \\
t_3 &= t_2 \oplus (t_0 \gg 8) \\
q &= t_3
\end{aligned}$$

The LCG parameters used in this study for LCG with shift transformation is same with LCG and can be seen in Table 3.1. The shift transformations can be also seen in the equation above.

3.2 Multiple Recursive Generator (MRG)

Multiple recursive generators, a common type of pseudo random number generator for simulation based methods, are based on a linear recurrence of the form given in Equation 3.2. The prime modulus m was either chosen as Mersenne-Prime or Sophie-Germain Prime. The parameters used for generating random numbers with MRG with chosen Mersenne-Prime modulus can be seen in Table 3.2 and the chosen Sophie-Germain Prime modulus can be seen in Table 3.3. There are several researches to find good parameters for both MRGs and LCGs, the parameters used in this study obtained from L'Ecuyer et al. [22].

$$X_n = (a_1.X_{n-1} + a_2.X_{n-2} + ... + a_k.X_{n-k}) \mod m \tag{3.2}$$

Table 3.2: MRG Parameters with Mersenne-Prime Modulus

	a_1	a_2	a_3	a_4	a_5	$Period$	m
$mrg2$	1498809829	1160990996				2^{62}	$2^{31}-1$
$mrg3$	2021422057	1826992351	1977753457			2^{93}	$2^{31}-1$
$mrg4$	2001982722	1412284257	1155380217	1668339922		2^{124}	$2^{31}-1$
$mrg5$	107374182	0	0	0	104480	2^{155}	$2^{31}-1$

Table 3.3: MRG Parameters with Sophie-Germain Prime Modulus

	a_1	a_2	a_3	a_4	a_5	$Period$	m
$mrg3s$	2025213985	1112953677	2038969601			2^{93}	$2^{31}-21069$
$mrg5s$	1053223373	1530818118	1612122482	133497989	573245311	2^{155}	$2^{31}-22641$

3.3 Inversive Congruential Generator

Inversive congruential generators are a form of nonlinear congruential pseudo random number generator that produces the next number in a series by using the modular multiplicative inverse (if one exists). Modulo some prime q, the conventional formula for an inversive congruential generator is given in (3.3):

$$X_{n+1} = (a.X_n^{-1} + c) \mod q, \tag{3.3}$$

where

> q : the prime modulus
>
> $a, 0 < a < m$: the multiplier
>
> $c, 0 \leq c < m,$: the increment
>
> $X_0, 0 \leq X_0 < m$: the start value, seed of the generator

3.4 Mersenne Twister Pseudo Random Number Generator

Mersenne twister algorithm was introduced in 1997 by Makoto Matsumoto and Takuji Nishimura [25]. It has been proven that the period of $2^{19937} - 1$ can be established with the good parameters, and equidistributed in (up to) 623 dimensions (for 32-bit values), and runs faster than other statistically reasonable generators. The Mersenne twister algorithm mostly used and chosen for simulations and models that requires random number generators.

The Mersenne Twister creates integers in the range $[0, 2^w - 1]$ for a w-bit word length. The method is a twisted generalised feedback shift register (twisted GFSR, or TGFSR) with state bit reflection and tempering in rational normal form (TGFSR(R)). The algorithm requires the following parameters ;

> w : Word size
>
> n : Degree of recurrence
>
> m : Middile word, an offset used in the recurrence relation, $1 \leq m < n$
>
> r : Seperation point of one word, $0 \leq r \leq w - 1$
>
> a : Coefficients of the rational normal form twist matrix
>
> b, c : TGFSR(R) tempering bitmasks
>
> s, t : TGFSR(R) tempering bit shifts
>
> u, d, l : additional Mersenne Twister tempering bit masks

The x series is defined as a series of w-bit quantities with the recurrence relation :

$$x_{k+n} = x_{k+m} \oplus ((x_k^u \mid x_{k+1}^l)A) \ , \ k = 0, 1, ...$$

In the equation above, the | expresses the concatenation operation and the $\oplus$ denotes the bit wise XOR operation. A value in the equation means the matrix with the form below. I_{w-1} denotes the $(w-1) \times (w-1)$ identity matrix.

$$A = \begin{pmatrix} 0 & I_{w-1} \\ a_{w-1} & (a_{w-2}, a_{w-3}, ..., a_0) \end{pmatrix}$$

Mersenne twister algorithm includes a tempering process as given in the expressions below. The x value in the equations below denotes next random number from the series, the y value is an intermediate variable and the z value the denotes the output of the algorithm. In the given operations, $\gg$ and $\ll$ denotes the bit wise left and right shifts, $\&$ means the bit wise and operation.

$$\begin{aligned} y &\equiv x \oplus ((x \gg u) \& d) \\ y &\equiv y \oplus ((y \gg s) \& b) \\ y &\equiv y \oplus ((y \gg t) \& c) \\ z &\equiv y \oplus (y \gg l) \end{aligned}$$

The Mersenne Twister algorithm is used as default random number generator for many softwares and operating systems. The most widely used version is based on the Mersenne prime $2^{19937} - 1$ with the parameters given in Table 3.4.

Table 3.4: Mersenne-Twister Parameters

w	32
n	624
m	397
r	31
a	$(9908B0DF)_{16}$
u	11
d	$(FFFFFFFF)_{16}$
s	7
b	$(9D2C5680)_{16}$
t	15
c	$(EFC6000)_{16}$
l	18

3.5 Permuted Congruential Generator (PCG)

The PCG family of pseudo random number generators developed by O'neill in 2014 [27] and commonly used with the newly developed technologies. PCG variants has a common approach on generating random numbers, an RNG (LCG or MCG) is employed for internal state generation and the permutation function is applied to the output of internal state generator with truncation of the value. The variants of PCG are named with the permutation function applied to the internal generator. The members of PCG family algorithms are can be expressed as PCG-XSH-RR (XorShift, Random Rotation), PCG-XSH-RR (XorShift, Random Rotation), PCG-XSH-RS (XorShift, Random Shift) and PCG-XSL-RR (XorShift Low bits, Random Rotation). The random number generation process of PCG algorithm can be generalized as given in Figure 3.1.

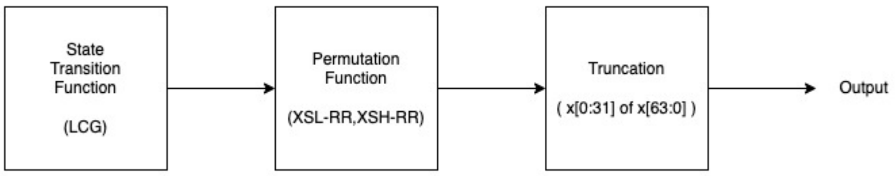


Figure 3.1: Permuted Congruential Generator

The LCG function used as internal state generator of PCG-XSL-RR and the permutation function of the PCG is given in the following equation. The $\ggg$ operation denotes the right rotation and the $\oplus$ denotes the XOR operation.

$$S_{i+1} = aS_i + c \mod 2^{128}$$
$$X_i = \Big(S_i[0:64] \oplus S_i[64:128] \ggg S_i[122:128]\Big)$$

As it can be understood from the equation above, first part of the sequence generation is same as the LCG. In the second part of the equation, the first 64 bit and second 64 bit of the binary of the generated random number used as the input values of XOR function and the output of the XOR operation rotated at most 127 bit.

CHAPTER 4

RESULTS

In this chapter, after giving a brief information about the CVRP benchmark instances, our results of the clustering with random projection trees approach combination with two heuristic methods NNI and CWS results on different set of benchmarks will be given. Finally, comparison of five different pseudo random number generators with changing parameters on Binary-CWS-MCS method will be introduced.

4.1 CVRP Benchmark Instances

There are several benchmarks for TSP, VRP and VRP variants in the literature. The most known benchmark for CVRP is Set E which is conducted based on different studies on CVRP [8, 12, 11]. The node and depot locations in the instances of Set E, generated in both from uniform random distribution and real world problem which is UK cities with customers located far from the depot. An instance from Set E called E-n76-k8 with single depot, 75 customer nodes with vehicle capacity of 180 from Set E can be seen in Figure 4.1.

Christofides et al. [7] had generated Set M by combining the locations of nodes from the Set E to gather larger problem instances in 1979. Location based scatter plot of an instance from Set M called M-n151-k12 with single depot, 150 customer nodes with vehicle capacity of 200 can be seen in Figure 4.2.

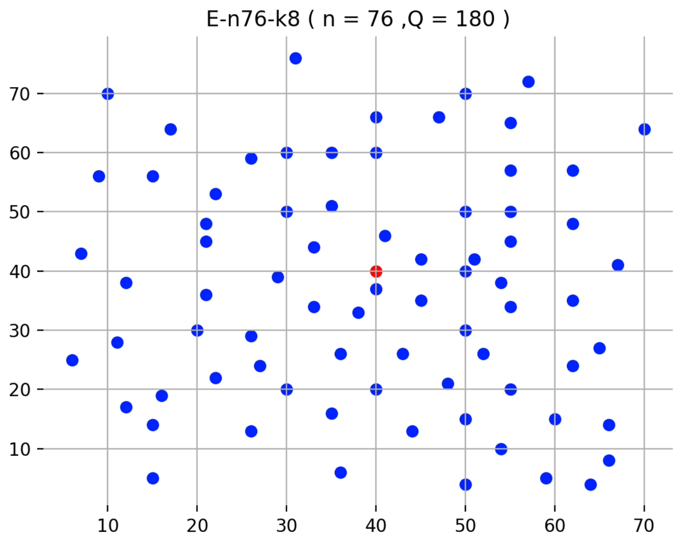


Figure 4.1: E-n76-k7 Instance

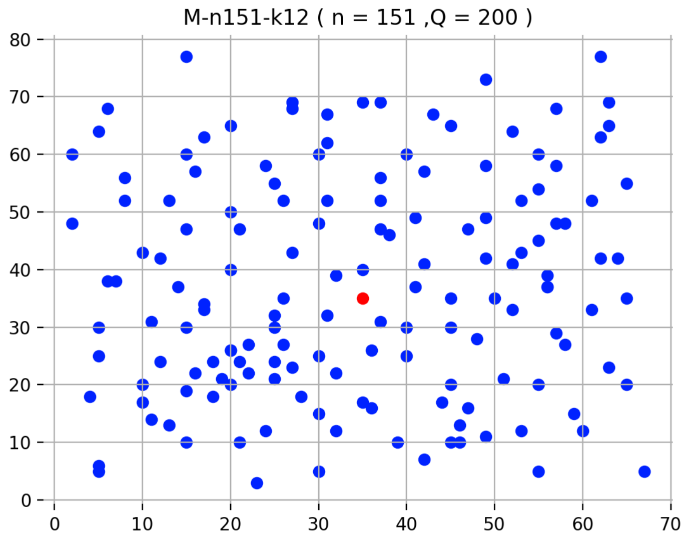


Figure 4.2: M-n151-k12 Instance

Set F has built by combining two different real life problems from a day of grocery deliveries from the Peterboro (Ontario terminal) of National Grocers Limited and data obtained from Exxon associated to the delivery of tires, batteries and accessories to gasoline service stations in 1994 by Fisher [14].

Rochat and Taillard had conducted another set of instances, Set Tai, each instance includes 75 to 150 customers and the data of the instances generated with a scheme that centers the depot and the nodes are clustered. The most customer including instance from Set Tai called Tai385 instance node location based scatter plot can be seen in Figure 4.3. The Tai385 instance includes 385 customer nodes with vehicle capacity of 65 were generated on real demand values gathered from the canton of Vaud in Switzerland.

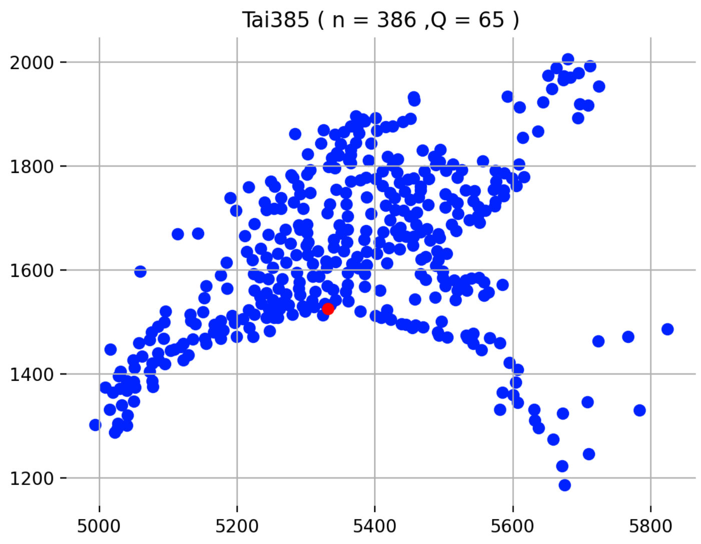


Figure 4.3: Tai385 Instance

A larger set of instances, Set G, produced in 1998 [18]. The node locations of Set G instances, follow concentric geometric patterns in shape of circles, stars and squares. Golden 9 instance with 256 customer nodes with vehicle capacity of 1000 from Set G, can be visualized as given in Figure 4.4.

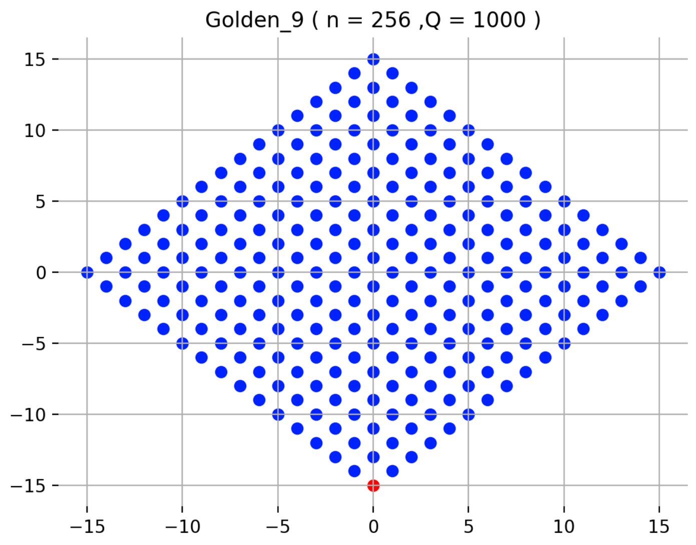


Figure 4.4: G9 Instance

Set X, mostly used set of benchmarks is generated in 2017 with range of 100 to 1000 number of nodes in each instance [31]. An instance called as X-n219-k73 with 219 customer nodes with 3 vehicle capacity node location based scatter plot can be seen in Figure 4.5. More information about the CVRP benchmark instances can be found in [31].

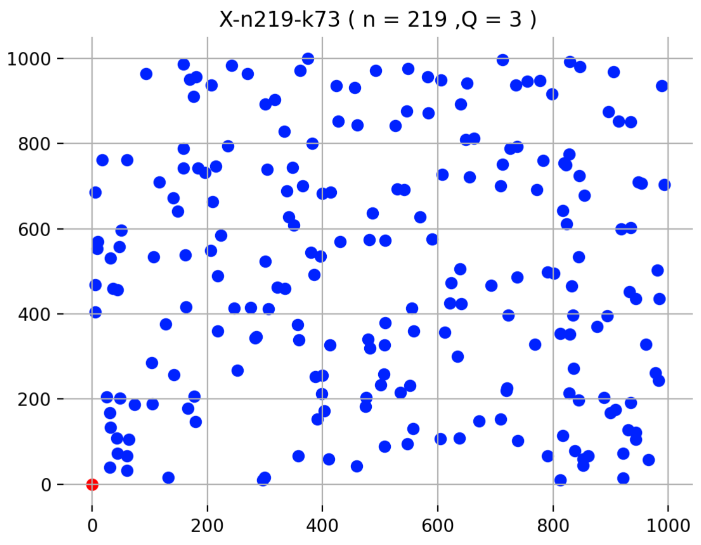


Figure 4.5: X-n219-k73 Instance

4.2 Applying Clustering with Heuristic Methods

Two different types of heuristic methods, NNI and CWS are chosen to be used with clustering approach to distinguish a combination of clustering with heuristic method gives better results than the core heuristic methods in this study. The experiments conducted on Golden, E, M, Li, Tai and X benchmark sets. In this section, results of topological clustering and the random projection clustering are given.

It is also important to note that we have also tested DBSCAN and K-means algorithms to cluster the CVRP instances and solving the sub-instance with heuristic methods such as CWS, NNI but the results of the process was not successful on any set of CVRP. Juan et. al had developed a hybrid algorithm called SR-GCWS [20] for solving the CVRP instances, pseudo code of the algorithm can be seen in Algorithm 10.

Algorithm 10 SR-GCWS Algorithm

```
Input: Nodes $N$, Capacity $C$, Distance Matrix $M$
Output: Route List
1: savingsList = GenerateSavingMatrix($C$,$n$)
2: cwsSol = ConstructCWSSolution($N$,$M$,$S$,$C$)
3: while stopping criteria not satisfied do
4:     routeList = constructRandomSolution($N$,$M$,$S$,$C$)
5:     routeList = improveSolUsingRoutesCache(routeList,$M$)
6:     if Cost of routeList < Cost of cwsSol then
7:         routeList = improveSolUsingSplitting($N$,$M$,$S$,$C$)
8:     end if
9: end while
10: return routeList
```

It can be seen in the pseudo code that SR-GCWS algorithm is an iteration based approach of starting from an initial solution and applying cache and splitting techniques to improve the obtained initial feasible solution. We have tried to improve the method by changing splitting approach with K-means clustering. The cache mechanism have implemented successfully but applying K-means clustering as a splitting policy on SR-GCWS algorithm did not perform compared to the both Binary-CWS-MCS and the results from [20].

4.2.1 Applying Topological Clustering with Heuristic Methods

The experiments made by adopting persistence based on the clustering technique is called as Topological Mode Analysis Tool (ToMaTo) from the study of Chazal et. al. [6]. The experiments have been conducted based on the idea of cluster-first-route-second approach by adopting ToMaTo as clustering algorithm and CWS as routing algorithm tested on Golden, Tai, M and E benchmark instance sets. The results of the experiments can be seen on Table 4.1. The results has shown that employing clustering on Golden benchmark set gives better results but as given in the CVRP Benchmark Instances section, the benchmarks contained in Golden set follow an artificial pattern and also the demand required by nodes in the set contains statistical weakness as discussed in [31].

Table 4.1: Results of ToMaTo with CWS - Golden Set

Problem	**CWS**	**ToMaTo-CWS**	**Diff**
Golden_10	722	602	16.62%
Golden_12	1072	905	15.58%
Golden_11	875	763	12.80%
Golden_15	1483	1382	6.81%
Golden_14	1214	1133	6.67%
Golden_9	557	524	5.92%
Golden_16	1812	1715	5.35%
Golden_13	940	906	3.62%
Golden_17	753	776	-3.05%
Golden_18	1079	1140	-5.65%
Golden_19	1467	1553	-5.86%
Golden_20	1972	2104	-6.69%

Because of the statistical weakness of Golden Set, other experiments made on another two different benchmark sets which are M, E and Tai Sets. It can be seen in Table 4.2 that the combination of ToMaTo with CWS algorithm did not perform better than the CWS algorithm.

Table 4.2: Results of ToMaTo with CWS - M Set

Problem	**CWS**	**ToMaTo-CWS**	**Diff**
M-n151-k12	1116	1439	-28.94%
M-n200-k16	1377	1745	-26.72%
M-n200-k17	1377	1745	-26.72%
M-n121-k7	1053	1194	-13.39%
M-n101-k10	836	1116	-33.49%

The results of experiments made on E Set can be seen in Table 4.3. The given method better perform on smaller instances on E Set. Especially, the performance of routes generated for E-n33-k4 (Single depot, 32 customer nodes and capacity = 8000), E-n23-k3 (Single depot, 22 customer nodes and capacity = 4500) instances is better performing compared to results obtained with CWS algorithm. Also, the performance of results gathered for E-n76-k7 (Single depot, 75 customer nodes and capacity = 220) and E-n51-k5 (Single depot, 50 customer nodes and capacity = 160) is slightly better. But, for the rest of results the cluster-first-route-second approach combination of ToMaTo and CWS is not performing better. The experiments made on Tai Set can be seen in Table 4.4.

Table 4.3: Results of ToMaTo with CWS - E Set

Problem	**CWS**	**ToMaTo-CWS**	**Diff**
E-n33-k4	843	654	22.42%
E-n23-k3	621	537	13.53%
E-n76-k7	747	734	1.74%
E-n51-k5	582	580	0.34%
E-n76-k10	896	898	-0.22%
E-n76-k8	781	785	-0.51%
E-n76-k14	1062	1078	-1.51%
E-n30-k3	485	498	-2.68%
E-n31-k7	612	631	-3.10%
E-n22-k4	387	422	-9.04%
E-n101-k8	880	1072	-21.82%
E-n101-k14	1140	1400	-22.81%
E-n13-k4	257	348	-35.41%

As a result of all experiments conducted on different benchmark sets with cluster-first-route-second approach built with adopting ToMaTo as clustering and CWS as clustering algorithm did not perform better than the CWS algorithm.

Table 4.4: Results of ToMaTo with CWS - Tai Set

Problem	**CWS**	**ToMaTo-CWS**	**Diff**
tai75c	1343	1306	2.76%
tai150c	2438	2385	2.17%
tai385	25214	24951	1.04%
tai100d	1578	1622	-2.79%
tai100b	2037	2139	-5.01%
tai150b	2876	3076	-6.95%
tai100c	1435	1605	-11.85%
tai75b	1354	1533	-13.22%
tai75a	1643	1879	-14.36%
tai150a	3290	3778	-14.83%
tai75d	1415	1828	-29.19%
tai150d	2774	3656	-31.80%
tai100a	2176	2929	-34.60%

4.2.2 Applying Random Projection Clustering with Heuristic Methods

In this section, you can see the results of the heuristic methods combined with clustering approach with random projection trees. We have chosen two mostly known heuristic methods which are Nearest Neighbor Insertion (NNI) and the Clarke and Wright's Savings (CWS) algorithms to use with random projection clustering approach. The experiments made on five different types of CVRP instance sets for the purpose of avoiding dependence between instance set type and method. The RPC-NNI values represents the NNI algorithm combined with random projection clustering and the RPC-CWS value represents the CWS algorithm combined with random projection clustering.

The Diff. NNI value represented on the table stands for the difference between the RPC-NNI and NNI and the Diff. CWS represented on the table stands for the difference between RPC-CWS and CWS value in percentile about the improvement on the performance. As the difference values given on Table 4.5, the RPC-NNI method has performed average of %7.17 than the Nearest Neighbor Insertion algorithm and RPC-CWS method has performed average of % 2.4 better than the Clarke and Wright's Savings algorithm based on the travel costs. RPC-NNI method has improved result

8.76% percent at most and had gathered worse results than NNI on two instances with average 2.75% percent.

Table 4.5: Comparison of NNI, CWS and RPC-NNI, RPC-CWS

Problem	NNI	RPC-NNI	Diff. NNI	CWS	RPC-CWS	Diff. CWS
tai100b	2465	2279	7.55%	1997	1998	-0.05%
tai100d	1979	1829	7.58%	1665	1640	1.50%
tai100a	2533	2463	2.76%	2166	2149	0.78%
tai75b	1852	1606	13.28%	1368	1368	0.00%
E-n76-k10	1049	977	6.86%	868	864	0.46%
E-n22-k4	459	413	10.02%	388	375	3.35%
E-n101-k14	1373	1295	5.68%	1136	1108	2.46%
Golden_4	16388	15334	6.43%	17198	16338	5.00%
Golden_16	1944	1804	7.20%	1752	1736	0.91%
Golden_6	10157	9385	7.60%	9942	9451	4.94%
Golden_7	12252	11179	8.76%	12242	11448	6.49%
Li_26	29146	27471	5.75%	27874	27109	2.74%
Li_32	47578	44246	7.00%	41131	38955	5.29%
M-n200-k16	1613	1585	1.74%	1347	1321	1.93%
M-n101-k10	1143	1037	9.27%	837	837	0.00%

The results of RPC-NNI and RPC-CWS can also be analyzed by concerning the type of routing algorithms. It can be said that the results generated with neighbor search type of algorithms such as NNI, can be improved by guiding its policy of neighbor search. Also, the performance of construction type of heuristic algorithms can be improved with the adoptation of clustering approach as the preparation step for generation of routes.

4.3 Comparison of Pseudo Random Generators on Binary-CWS-MCS

In order to compare the pseudo random number generators, the Binary-CWS-MCS method with random projection clustering implemented. For the purpose of Monte Carlo Tree Search, the Binary-CWS method has employed as selection process and simulation is accomplished with the bulk operations given in Algorithm 6. The comparison made on the basis of cost performances of the Binary-CWS-MCS combination of random projection clustering method. In order to compare the cost functions, the selection process at MCTS step is chosen and the different PRNGs used in this function to evaluate the differences. LCG, LCGS, MRG, MRGS, ICG, MT and PCG generators has been employed for comparison. The PRNGs that has been used in this thesis, has been implemented from two different well-known PRNG libraries and detailed information about the PRNGs can be obtained from the study [2]. While choosing the pseudo random number generators, the consideration is to both test the mostly used PRNGs on Monte Carlo Simulation and relatively new methods in the field. For this purpose, the LCG, LCGS, MRG, MRGS and ICG is adopted with MT and a variant from PCG family.

Three different sets of CVRP instances is chosen considering time required for running Monte Carlo simulations. Thus, we choose E, M and X instance sets. The analysis of comparison is made by using five different perception as the number of best solutions, number of worst solutions, loss against the median cost value, loss against the average cost value and the average loss against to average solution. As given in Table 4.6, the best performing PRNG was the PCG64 followed by MT by concerning the number of best solutions and number of worst solutions. If the number of worst solutions considered as the performance indicator, it can be seen that the minimum value appears on three different PRNGs, MRG3, MRG5 and PCG64. If we examine the average loss against to best solution parameter, minimum values appear on PCG64 and MRG5S, also it is important to note that MRG5S performs better compared to MT on the contrary to number of worst solutions. MRG5S also performs better than MT considering the average loss against median solution and the average loss against the average solution. The analysis also shows that shift transformation variants of LCG and MRG is better performing than the LCG and MRG. It can also can be examined that using different parameters for generating pseudo random numbers using MRG can produce better impact on results if we compare the results of MRG1, MRG3, MRG4 and MRG5. It is also important to note that while both MRG3S and MRG5S uses shift transformation, the performance indicators shows that MRG5S generates better solutions based on the cost function.

Table 4.6: Comparison of Random Number Generators

Instance	**LCG**	**LCGS**	**MRG2**	**MRG3**	**MRG4**	**MRG5**	**MRG3S**	**MRG5S**	**MT**	**PCG64**	**ICG**
E-n22-k4	381	376	376	376	389	376	376	376	376	376	376
E-n33-k4	928	927	927	941	927	928	927	927	927	939	927
E-n76-k8	851	807	846	840	815	832	827	850	836	840	825
E-n76-k10	946	946	954	943	926	951	936	937	946	923	956
E-n76-k14	1143	1148	1159	1152	1151	1137	1161	1141	1171	1139	1200
E-n101-k14	1217	1239	1230	1269	1237	1211	1234	1238	1252	1203	1263
E-n51-k5	591	585	585	585	584	561	589	579	580	583	577
E-n76-k7	728	741	725	731	714	717	718	741	726	718	714
E-n23-k3	606	622	616	622	608	616	630	606	606	613	606
X-n101-k25	28860	29133	29064	29025	29519	29524	29218	29228	28925	29502	31992
X-n120-k6	14713	15191	14955	14767	14888	15222	15681	14687	14296	15305	14514
X-n106-k14	27856	27648	27795	27907	28080	28119	27498	27993	27790	27902	28918
X-n115-k10	14529	14238	14584	14547	14378	14579	14465	14408	14574	14143	14789
X-n110-k13	16369	16672	16495	16863	16621	16819	16474	16502	16869	16770	16457
X-n139-k10	18564	17794	17811	18280	18718	19143	19188	17503	18127	17594	19491
X-n204-k19	27928	27062	26802	27150	26471	27375	26925	27341	27294	26223	27380
X-n228-k23	34589	32669	36324	36037	34013	34281	35579	36646	35191	33569	35594
X-n979-k58	172011	168262	165857	164940	162359	144835	166314	167789	164008	166173	165647
M-n101-k10	889	914	920	914	907	905	907	906	906	896	902
M-n200-k17	2014	2033	2039	1983	1977	2114	1960	1972	2097	1911	1944
Number of best solutions	3	4	2	1	2	3	3	4	4	5	4
Number of worst solutions	3	1	0	2	1	0	2	2	1	0	6
Average Loss to best solution	-2.37%	-2.02%	-2.13%	-2.69%	-2.06%	-2.48%	-2.76%	-1.87%	-2.08%	-1.56%	-3.60%
Average Loss to median solution	-0.14%	0.20%	0.09%	-0.46%	0.16%	-0.24%	-0.52%	0.35%	0.14%	0.65%	-1.35%
Average loss to average solution	-0.16%	0.18%	0.07%	-0.48%	0.14%	-0.26%	-0.54%	0.32%	0.11%	0.63%	-1.37%

CHAPTER 5

CONCLUSION

The improvements of the technology and rapid growth of population result in more demand for transportation and delivery services. In order to overcome the increasing resource consumption, better performing algorithms for problems like VRP and its variants are required. Therefore, the capacitated vehicle routing problem (CVRP) has been studied in this thesis. The CVRP is a problem concerned with the finding the optimum routes to traverse a given set of customers with a given fleet of vehicles.

Two different types of experiments have been made on this thesis, firstly we have tried to combine different clustering approaches with two different heuristic methods and the second type of experiments made to compare the pseudo random number generators with a Monte Carlo simulation type of algorithm is called as Binary-CWS-MCS combination with random projection tree based clustering technique. The first type of experiments has been done with four different clustering techniques which are K-means, DBSCAN, ToMaTo and random projection clustering, combined with two heuristic routing algorithms called as NNI and CWS. As the results showed, the clustering approach with heuristic methods can improve the performance of the solutions. For the second part, we have conducted experiments based on the pseudo random number generator types and their variants on three different CVRP instance sets and on twenty different instances. The random number generators used in this study is gathered from on respect to two perspective, our first aim was to compare the mostly known and used RNGs on simulation based studies and we have implemented the library known as Tina's RNG library [2] and the others chosen concerning the improvements in the random number generator algorithms which are MT, PCG. Additionally, our results have shown that the choice of RNG can affect the performance of the solutions. As results of experiments presents, the best performing random number generator for the given set of instances was from the family of Permuted Congruential Generator(PCG-64). If we compare the number of best solutions for LCG and LCGS, LCG with shift transforms has concluded on better results than the normal LCG and also the average loss values were better for LCGS. Parallel to this result and parameters, MRGS generators also has produced better results than the MRG. PCG64 and MT has concluded on better results than other RNG types on the concern of both number of best solutions value and average loss parameters. ICG has performed relatively bad than the other RNGs.

The research in this manner can be continued by both adding a feedback mechanism to the simulation process for improving the performance. Also the improvements in the random projection clustering with a guided approach can be made in the specific case of VRP and its variants.

REFERENCES

[1] D. Applegate, R. Bixby, W. Cook, and V. Chvátal. On the solution of traveling salesman problems. 1998.

[2] H. Bauke and S. Mertens. Random numbers for large-scale distributed monte carlo simulations. *Physical Review E*, 75(6):066701, 2007.

[3] E. Bernhardsson. *Annoy: Approximate Nearest Neighbors in C++/Python*, 2018. Python package version 1.13.0.

[4] G. Bird. Monte-carlo simulation in an engineering context. *Progress in Astronautics and Aeronautics*, 74:239–255, 1981.

[5] W. K. V. Chan. *Theory and applications of Monte Carlo simulations*. BoD–Books on Demand, 2013.

[6] F. Chazal, L. J. Guibas, S. Y. Oudot, and P. Skraba. Persistence-based clustering in riemannian manifolds. *Journal of the ACM (JACM)*, 60(6):1–38, 2013.

[7] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318, 1969.

[8] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[9] J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo. Vehicle routing. *Handbooks in operations research and management science*, 14:367–428, 2007.

[10] G. Dantzig. *Linear programming and extensions*. Princeton university press, 2016.

[11] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[12] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[13] V. Estivill-Castro. Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter*, 4(1):65–75, 2002.

[14] M. L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations research*, 42(4):626–642, 1994.

[15] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

[16] M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.

[17] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349, 1974.

[18] B. L. Golden, E. A. Wasil, J. P. Kelly, I. Chao, et al. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In *Fleet management and logistics*, pages 33–56. Springer, 1998.

[19] M. E. Gülşen and O. Yayla. Binary-CWS-MCS combination with Random Projection Clustering, 9 2022.

[20] A. A. Juan, J. Faulin, R. Ruiz, B. Barrios, and S. Caballé. The sr-gcws hybrid algorithm for solving the capacitated vehicle routing problem. *Applied Soft Computing*, 10(1):215–224, 2010.

[21] N. Kohl. Exact methods for time constrained routing and related scheduling problems. 1995.

[22] P. L'Ecuyer, F. Blouin, and R. Couture. A search for good multiple recursive random number generators. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 3(2):87–98, 1993.

[23] F. Li, B. Golden, and E. Wasil. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & operations research*, 34(10):2918–2930, 2007.

[24] P. L'ecuyer. Tables of linear congruential generators of different sizes and good lattice structure. *Mathematics of Computation*, 68(225):249–260, 1999.

[25] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[26] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[27] M. E. O'Neill. Pcg: A family of simple fast space-efficient statistically good algorithms for random number generation. Technical Report HMC-CS-2014-0905, Harvey Mudd College, Claremont, CA, Sept. 2014.

[28] G. L. Steele Jr and S. Vigna. Computationally easy, spectrally good multipliers for congruential pseudorandom number generators. *Software: Practice and Experience*, 52(2):443–458, 2022.

[29] F. Takes and W. A. Kosters. Applying monte carlo techniques to the capacitated vehicle routing problem. In *Proceedings of 22th Benelux conference on artificial intelligence (BNAIC 2010)*, 2010.

[30] W. Thomson. A modified congruence method of generating pseudo-random numbers. *The Computer Journal*, 1(2):83–83, 1958.

[31] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.