SOLVING MULTI-OBJECTIVE DYNAMIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS USING MULTI-OBJECTIVE ALGORITHM

KHOO THAU SOON

DOCTOR OF PHILOSOPHY (SCIENCE)

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN
APRIL 2022

SOLVING MULTI-OBJECTIVE DYNAMIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS USING MULTI-OBJECTIVE ALGORITHM

By

KHOO THAU SOON

A thesis submitted to the Department of Electrical and Electronic Engineering,
Lee Kong Chian Faculty of Engineering and Science,
Universiti Tunku Abdul Rahman,
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy (Science)
April 2022

ABSTRACT

SOLVING MULTI-OBJECTIVE DYNAMIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS USING MULTI-OBJECTIVE ALGORITHM

Khoo Thau Soon

Logistics plays a very important role in the business economy. It is over a trillion of dollars in revenue annually and increase exponentially over the years One of the current trends is to solve the last mile is to optimize the delivery routes. One of the best ways to optimize the delivery routes is to study and implement the multi-objective dynamic vehicle routing problem with time windows because it resembles the online delivery services that are ubiquitous and propagate over the year, especially during the COVID-19 pandemic.

During the past decade, there is an increasing trend of published papers dealing with dynamic vehicle routing problems with time windows (DVRPTW) but not on multi-objective dynamic vehicle routing problems with time windows (MODVRPTW). Therefore, it brings a significant contribution if this study can be carried out because it represents the daily real-life problem in transportation. To solve this problem, it needs to be modelled and an algorithm is needed to be developed and tested to ascertain its efficiency and effectiveness.

It is difficult and challenging to develop an algorithm that can produce consistent near-optimal solutions even after many runs, average near-optimal solutions that have the least difference in magnitude, broader Pareto set, and achieve near-optimal solutions but highly sought after if it is commercially viable. Our algorithm uses non-fitness evolutionary distributed parallelized adaptive large neighbourhood search (NEDPALNS). The non-fitness evolutionary distributed (NED) takes advantage of the exploitation of the search space and the parallelized adaptive large neighbourhood search (PALNS) makes full use of the exploration and exploitation of its inner strength. These combinations achieve near-optimal solutions consistently. We compare our results using hypothetical datasets and real datasets. Our results are competitive and outperform other published algorithms and best-known solutions in both static and dynamic environments.

ACKNOWLEDGEMENTS

When I started this Ph.D. journey, I thought that studying part-time would give me more time and less pressure to do research. Little did I know the overwhelming challenges and obstacles that I will face to reach this final point. I thought it should be easier than the Master or Graduate level since I have more time to finish and there is no coursework need to take. Well, I guess I have underestimated it. I want to say that it is a super-exciting journey. I enjoy every moment and, like it or not, it is one of my life missions. I gained a lot of experience, but also ages, weight, and I have met a lot of remarkable people. Throughout the Ph.D. journey, some people have helped me directly and indirectly and I wish to thank them. Without them, I don't think I can achieve it and with them, I can always look forward every single day.

First, I would like to thank UTAR for accepting me as a Ph.D. candidate knowing that my case is unique. I would like to thank the Malaysian government for offering me a scholarship. I would also like to thank my ex-supervisors, Dr. Tay Yong Haur and Dr. Kheng Cheng Wai who have guided me for some time but left me for seeking greener pasture elsewhere. I shall remember you all. I want to thank my other ex-supervisor, whom I can change to a new supervisor for my betterment. You know that I am happier and more satisfied now.

I want to thank one very important person, my main supervisor, Dr. Mohammad Babrdel Bonab. You are an advocate for a free thinker. You are my lunch companion, my friend, my adviser, my lecturer, and the list goes on. Without your guidance and support, I might not be able to publish not just papers, but Q1 papers and many more papers in the making. Under your supervision, I progress faster, achieve better quality presentations, write quality papers and thesis. You are just like a convenience shop seven eleven, 24/7 always available. What else can I ask more for? I can say that I am very lucky to have you. I want to thank my co-supervisor, Dr. Wong Voon Hee for his knowledge of UTAR policy and procedure so that I don't go on the wrong path.

I want to thank Prof. Dr. Goi Bok Min, Dr. Yap Wun She, and Dr. Khaw Chwin Chieh for handling my complaint efficiently and effectively. I want to thank Dr. Chua Kein Huat for sharing his previous WCS presentation slides and thesis.

I want to thank the UTAR Lab personnel who are very cooperative and understanding. They are Mr. Teo Gee Man, Puan Nur Halimaton Sakdiah Binti Harun, Mr. Lim Eng Cheong, Encik Mohd Hafizul Bin Rameli and Encik Mohd Asraf Bin Ishak. I shall miss you all.

I want to thank Mr. Lau Kean Hong, Ms. Chong Yong Shean, Mr. Nick Lum Chan Fai, Mr. Tang Xin Jie, Mr. Wong Yi Hong, Ms. Ho Ming Cheng, Ms. Soh Ying Wei, Ms. Donica Kan, Mr. Chok Eu-Tjin, Mrs. Nneka Onubogu, Dr. Toh Chia Ming, and Mr. Peter Ng for your companionship and lunch partner. I have a great time with all of you. It is great to know you all. I know I missed some of your guys' names. My apologies for not remembering your name.

To my demised grandfather and grandmother, I know you are watching me there and will surely be proud of me. I would like to thank my parent who will never give up hope on me and bring me to this earth.

I want to especially thank my lovely wife, Ms. Ling Yu Yuen for her unparallel support, love, and dedication. Taking care of our children is not an easy task. With you, I am protected from interruptions, so I have the privilege of focusing on my research. To my son, Leo Khoo Ming Sheng, I want to thank you for your understanding that my study is of utmost importance than you playing your computer games. To my daughter, Letitia Khoo Hui Bao who supported me from Singapore and cannot come back due to travel restrictions caused by the coronavirus pandemic. I will always remember it. Also, my mother-in-law for making sure that I have plenty of fine dining food. Lastly, I want to dedicate this thesis to my wife, only with your understanding, love, and support, this Ph.D. journey will be a smooth sailing one. I will cherish every single moment.

APPROVAL SHEET

This dissertation/thesis entitled **"SOLVING MULTI-OBJECTIVE DYNAMIC VEHICLE ROUTING PROBLEM WITH TIME WINDOW USING MULTI-OBJECTIVE ALGORITHM"** was prepared by **KHOO THAU SOON** and submitted as partial fulfillment of the requirements for the Doctor of Philosophy (Science) at Universiti Tunku Abdul Rahman.

Approved by:

[illegible]

___________________________ Date: 12-April-2022

(Dr. Mohammad Babrdel Bonab)

Supervisor

Department of Internet Engineering and Computer Science

Lee Kong Chian Faculty of Engineering and Science

Universiti Tunku Abdul Rahman

[illegible]

___________________________ Date: 12-April-2022

(Dr. Wong Voon Hee)

Co-supervisor

Department of Mathematical and Actuarial Sciences

Lee Kong Chian Faculty of Engineering and Science

Universiti Tunku Abdul Rahman

LEE KONG CHIAN FACULTY OF ENGINEEING AND SCIENCE

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 12-April-2022

SUBMISSION OF THESIS

It is hereby certified that ***Khoo Thau Soon*** (ID No: ***13UED08520***) has completed this thesis entitled "***SOLVING MULTI-OBJECTIVE DYNAMIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS USING MULTI-OBJECTIVE ALGOIRTHM***" under the supervision of ***Dr. Mohammad Babrdel Bonab*** (Supervisor) from the Department of Internet Engineering and Computer Science, Faculty of Lee Kong Chian Faculty of Engineering and Science, and ***Dr. Wong Voon Hee*** (Co-Supervisor) from the Department of Department of Mathematical and Actuarial Sciences, Faculty of Lee Kong Chian Faculty of Engineering and Science.

I understand that University will upload softcopy of my thesis in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

[illegible]

(*Khoo Thau Soon*)

DECLARATION

I (KHOO THAU SOON) hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

[illegible]

(KHOO THAU SOON)

Date: 12-April-2022

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ALNS	Adaptive large neighbourhood search
AFIT	Air Force Institute of Technology
AI	Artificial Intelligence
BKS	Best-known solutions
CVRP	Capacitated vehicle routing problem
CEA	Coevolutionary algorithm
CAGR	Compound Annual Growth Rate
CA	Cultural algorithms
DoD	Degree of Dynamism
DM	Decision-maker
DE	Differential evolution
DALNS	Distributed adaptive local neighbourhood search
MODVRPTW	Dynamic vehicle routing problem with time windows
DVRPTW	Dynamic vehicle routing problems with time windows
EDA	Estimation of distribution algorithms
ES	Evolution strategies
ES	Evolution strategies
EA	Evolutionary algorithm
EC	Evolutionary computation
EVOPs	Evolutionary operators
EP	Evolutionary programming
FSM	Finite State Machine
GENMOP	General Multi-objective Evolutionary Algorithm
GVNS	General variable neighbourhood search

GD	Generational Distance
GA	Genetic algorithm
GP	Genetic programming
HTTP	Hypertext transfer protocol
HV	Hypervolume
ILNS	Improved local neighbourhood
IR	Industrial revolutions
IR	Insertion time
IGD	Inverted Generational Distance
IGD+	Inverted Generational Distance Plus
LNS	Large Neighbourhood Search
MPFE	Maximum Pareto Front Error
Micro-GA	Micro-Genetic Algorithm
MOEA	Multi-objective evolutionary algorithm
MOEA/D	Multi-objective evolutionary algorithm based on decomposition
MOEA	Multi-objective Evolutionary Algorithms
MOGA	Multi-Objective Genetic Algorithm
MOGPGA	multi-objective goal programming and genetic algorithm
MOMGA	Multiobjective Messy Genetic Algorithm
MOP	Multi-objective optimization problems
MOSGA	Multi-objective Struggle GA
MOVRPTW	Multi-objective vehicle routing problem with time windows

NSF	National Science Foundation
NPGA	Niched-Pareto Genetic Algorithm
NSGA	Nondominated Sorting Genetic Algorithm
NEA	Non-fitness evolutionary algorithm
NEDPALNS	Non-fitness evolutionary distributed parallelized adaptive large neighbourhood search
VN	Number of vehicles used
OX	Order crossover
OMOEA	Orthogonal Multi-Objective Evolutionary Algorithm
PALNS	Parallelized adaptive large neighbourhood search algorithm
PAES	Pareto Archived Evolution Strategy
PESA	Pareto Envelope-based Selection Algorithm
PF	Pareto front
PDPTW	Pickup and delivery with time windows
QI	Quality Indicators
RR	Rejection ratio
REST	Representational state transfer services
R&R	Ruin and recreate
DARP	Single-vehicle dial-a-ride problem
S	Spacing
SPEA	Strength Pareto Evolutionary Algorithm
TW1	Time windows 1
TD	Total travelled distance

TI	Transport Intelligence
UTAR	University of Tunku Abdul Rahman
VEGA	Vector Evaluated Genetic Algorithm
VRP	Vehicle routing problem
VRPTW	Vehicle routing problem with time windows
VRPSD	VRP with Stochastic Demand

CHAPTER 1

INTRODUCTION

1.1. Background

According to transport intelligence (TI), the global logistics industry is worth about 5.275 trillion euros in 2020 (Intelligence, 2021). The forecast period for recovery from the Covid-19 pandemic remains healthy at a compound annual growth rate (CAGR) of 4.7% from 2020 to 2024. This shows that growth and prospects in 2021 are expected to look vibrant and stronger as the logistics market is projected to recover from contractions in 2020 (Intelligence, 2021). Logistics costs are defined as the total of all expenditures incurred to make goods and services available to the end consumer. If logistics cost is to be broken down into different costs composition, the transportation cost has the highest share of the cost as they are accounted for nearly half of the logistic cost (Rodrigue, 2020) while the second-highest cost inventory carrying cost is only one-fifth of the total costs. The study of transportation cost is important because transportation remains one of the largest industries in the world and a key element in the logistic chain. Transportation connects supply chain components in visible and communicable ways.

One of the current trends in transportation is to harness artificial intelligence (AI) to provide efficient transport of goods across roads, seas, and air (HTEC, no date). AI transforms the traditional ways of achieving efficient paths to automatically design the superfluous and better optimal solution. One of the best options to apply AI is to learn and experiment with the vehicle routing problem (VRP) using AI.

VRP is a combinatorial optimization that addresses the optimal number of routes for a given fleet of vehicles to traverse and deliver goods to a given set of customers. VRP first appeared in a paper in 1959 by authors named George Dantzig and John Ramser (G.B. Dantzig, 1959). VRP is widely studied by both academic and non-academic researchers to mimic real-life scenarios. These studies do not end in just similar features VRP instead, more complex features have been added. The new features trend has increased exponentially over the year. These features could include adding time windows, time-dependent travel times, pick-up, and delivery, among others. With these complex features, the VRP has evolved into other variants. Among other variants, the vehicle routing problem with time windows (VRPTW) receives most academic spotlights even to this day. VRPTW has used different model parts of supply chain design and operation such as school bus routing, waste collection, food delivery service, goods distribution, urban newspaper distribution among others (Kallehauge and Solomon, 2005). It is also widely studied by academics and non-academics and has appeared in many quality journals.

In VRPTW, the main objective is to achieve the least total travelled distance within the given constraints. However, human nature has complex desires. They may have two or more desires to be fulfilled. These could be achieved both with the least number of vehicles and the least total travelled distance. These desires are called the multi-objective problem. This problem can appear in many disciplines, such as manufacturing, distribution, production, and economy, among others. In a multi-objective vehicle routing problem with time windows (MOVRPTW), there are two or more objective functions to be solved. These objectives are both the least total travelled distance and the least number of vehicles. There could be more than one feasible solution in the MOVRPTW. These feasible solutions are the non-dominating solution. This means that none of the solutions is dominating other solutions. Hence, these non-dominating solutions are the Pareto optimal set. The MOVRPTW objective is to achieve optimal in the broader Pareto set.

MOVRPTW could use static and deterministic information to calculate its objectives. This means that the information on the customers is known in advance and can be used for planning the routes. However, in the real-life scenario, not all customer information such as time to serve the customers, customer location, customer demand, among others, are known before the route planning starts. Therefore, the decision to plan the routes and serve these customers cannot be carried out simply. This type of problem is called a multi-objective dynamic vehicle routing problem with time windows (MODVRPTW). During the past decade, there was an increasing trend of published papers on dynamic vehicle routing problems with time windows (DVRPTW). However, in MODVRPTW, to the best of our knowledge, it is rarely studied. Even if it is, it is not frequent and may appear in a different form (Ghannadpour *et al.*, 2014). Therefore, MODVRPTW contributes significantly if the study can be conducted, as it represents the real-life problem in transportation activity that we face daily.

To solve this problem, the problem needs to be modelled and an algorithm is needed to be developed and tested to ascertain its efficacy and effectiveness. Some algorithms may produce consistent near-optimal solutions even after many runs, but do not support a broader Pareto set (Ursani *et al.*, 2011; Xu *et al.*, 2015; Zhang, Yang, and Weng, 2018). Other algorithms may generate consistent near-optimal solutions even after many runs but the Pareto set may not have the least difference in magnitude (Ghoseiri and Farid, 2010; Qi *et al.*, 2015a; Dong *et al.*, 2018). Some algorithms generate consistent near-optimal solutions after many runs but do not show average optimal solutions with the least difference in magnitude (Ropke and Pisinger, 2006; Sartori, 2016; Curtois *et al.*, 2018). It is hard to find an algorithm that can produce consistent near-optimal solutions even after many runs, average near-optimal solutions with the least difference in magnitude, broader Pareto set, and achieve near-optimal solutions. In addition, it is a challenge to develop an algorithm that achieves all this. Hence, such an algorithm is highly sought after and commercially beneficial if it can be developed and put into production.

1.2. Research Objectives

The objectives of this research are as follows:

- To develop a multi-objective algorithm with a distributed parallelized adaptive rebuilding capability that uses cyclic and non-cyclic optimization strategies
- To be able to support hypothetical and real datasets that consistently generated near-optimal solutions and to achieve an optimized Pareto set.
- To evaluate the performance of the proposed algorithm against the recently published results and best-known solutions

1.3. Research Methodology

This research aims to develop, test, analyse, and evaluate an algorithm that can achieve the following: -

- Produce consistent near-optimal solution even after many runs.
- Achieve the least difference in magnitude in average near-optimal solutions.
- Generate a broader Pareto set.
- Demonstrate outstanding solutions in another VRP variant.

Our research methodology has six steps. They are listed as follows:

Step 1: Literature review

Review and identify the VRPTW, MOVRPTW, DVRPTW, and MODVRPTW challenges, strength, weaknesses, and their state-of-the-art algorithms that are thoroughly investigated. This includes the extraction of the general concept and principle, as well as the usage of terminology.

Step 2: Problem formulation and solution

Formulate and modelling VRPTW, MOVRPTW, DVRPTW, and MODVRPTW problem using the Unified Modelling Language (UML) and coding. This also includes identifying the dataset that is used for testing.

Step 3: Agile development

Establishing proposed algorithm features and prioritizing them into backlog items. Break down the backlog items into workable items that can be completed within a few days. Plan sprint backlog task and finalize the sprint iteration. Perform requirement gathering, analysis, design, code, test, and user acceptance in each of the sprint tasks. Perform a daily stand-up meeting to evaluate whether it is behind schedule, plan for the next task and identify issues and take corrective action if needed.

Step 4: Deploy to production

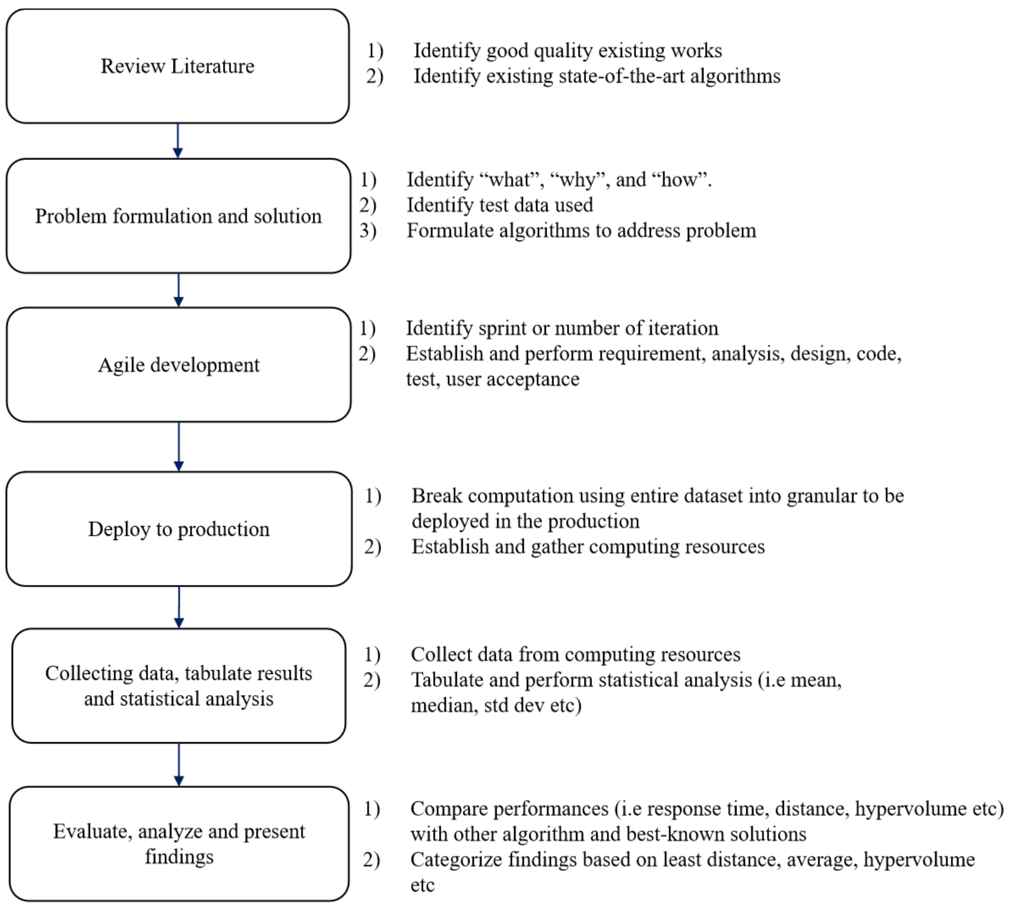
Establish and gather computing resources. Set up the production environment. Break the computation using the entire dataset into granular enough to be deployed to production. Automate the software deployment process.

Step 5: Collecting data tabulate results and statistical analysis

Collect data from computing resources. Extract data, tabulate and organize results into Excel sheet and perform statistical analysis using mean, standard, deviation, and average.

Step 6: Evaluate, analyse and present findings

Compare and contrast performance against the published algorithm and the best-known solutions. Categorize and present findings based on a metric. Figure 1.1 illustrates the research methodology steps.



Review Literature
1) Identify good quality existing works
2) Identify existing state-of-the-art algorithms
Problem formulation and solution
1) Identify "what", "why", and "how".
2) Identify test data used
3) Formulate algorithms to address problem
Agile development
1) Identify sprint or number of iteration
2) Establish and perform requirement, analysis, design, code, test, user acceptance
Deploy to production
1) Break computation using entire dataset into granular to be deployed in the production
2) Establish and gather computing resources
Collecting data, tabulate results and statistical analysis
1) Collect data from computing resources
2) Tabulate and perform statistical analysis (i.e mean, median, std dev etc)
Evaluate, analyze and present findings
1) Compare performances (i.e response time, distance, hypervolume etc) with other algorithm and best-known solutions
2) Categorize findings based on least distance, average, hypervolume etc

Figure 1.1: Research methodology steps

1.4. Research Scope

The scope of this research is to develop and design a multi-objective algorithm that consistently generates optimal solutions with the least difference in magnitude, broader Pareto set, and least difference in average optimal solutions in an online or offline environment. The algorithm is designed and developed based on a distributed architecture that provides seamless execution of the rebuilding algorithm asynchronously and addresses MODVRPTW. The experiments are conducted using static and dynamic datasets to determine the effectiveness of the algorithm in an online and offline environment. The results are compared with the published algorithms, the best known solutions using quantitative metrics to ascertain performance.

1.5. Thesis Organization

The content of the thesis is organized as follows:

Chapter 2 explains the variants of VRP and its differences. It also distinguishes the different types of evolutionary algorithms and the rebuilding algorithm used in solving the variants of VRP. It dives deep into one of the popular VRP variants in which it highlights the Pareto set, the classification of multi-objective evolutionary algorithms, types of multi-objective evolutionary algorithms, and the options available in the multi-objective qualitative assessment.

Chapter 3 explains the general definition of DVRPTW. The DVRPTW system characteristics and optimization strategies are used in developing the proposed algorithm. It elaborates on the proposed algorithm and describes its architecture, representation in microservices, and process flow. It explains each function of the proposed algorithm and its characteristics, its purposes, and what NEDPALNS is made of.

Chapter 4 focuses on two types of datasets to evaluate the proposed algorithm. These datasets are based on the degree of dynamism to achieve different results at different dynamism. It also focuses on the proposed algorithm parameter settings to achieve near-optimal solutions and the testing environment in which it operates. The comparison and assessment of the results using the different degrees of dynamism. Qualitative and quantitative metrics are used for the comparison of the results.

Chapter 5 concludes the key findings of the research and its implications. The limitations and opportunities for future enhancement of the proposed algorithm are explained.

1.6. List of Publications

Table 1.1: List of publications

No	**Authors, Title, Link, and Status**	**Journal(J)/ Proceeding(P)/ Book Chapter(B) Conference (C)**	**Index/ Impact Factor/**
1	Thau Soon Khoo and Babrdel Bonab Mohammad, "A Distributed Non-elitist Evolutionary Scalable Asynchronous Rebuilding Algorithm for Solving Pickup and Delivery Problems with Time Windows"	(C): 2021 International Conference on Decision Aid Sciences and Application (DASA)	**Accepted (2021)**
2	Thau Soon Khoo and Babrdel Bonab Mohammad, "The parallelization of a two-phase distributed hybrid ruin-and-recreate genetic algorithm for solving multi-objective vehicle routing problem with time windows," https://doi.org/10.1016/j.eswa.2020.114408.	(J): Expert Systems with Applications	ISI/WOS Q1 Paper, IF – 6.725 (2020) **Published (2021)**
3	Thau Soon Khoo, Babrdel Bonab Mohammad, Voon Hee Wong, Yong Haur Tay, and M. Nair, "A Two-Phase Distributed Ruin-and-Recreate Genetic Algorithm for Solving the Vehicle Routing Problem With Time Windows," https://doi.org/10.1109/ACCESS.2020.3023741.	(J): IEEE Access	ISI/WOS Q1 Paper, IF - 3.745 (2019) **Published (2020)**
4	Bonab M.B., Tay Y.H., Mohd Hashim S.Z., Soon K.T. (2019) "An Efficient Robust Hyper-Heuristic Algorithm to Clustering Problem." In: Omar S., Haji Suhaili W., Phon-Amnuaisuk S. (eds) Computational Intelligence in Information Systems. CIIS 2018. Advances in Intelligent Systems and Computing, vol 888. Springer, Cham. https://doi.org/10.1007/978-3-030-03302-6_5)	(B): Springer	SCOPUS **Published (2019)**
5	Thau Soon Khoo and Babrdel Bonab Mohammad, "Solving Multi-objective Pickup and Delivery with Time Windows using Mediocre Evolutionary Distributed Microservices Re-optimization Algorithm"	(J): Applied Soft Computing	Final review (2022)
6	Thau Soon Khoo and Babrdel Bonab Mohammad, "Solving Multi-objective Vehicle Routing Problem with Time Windows using MOVRPTW dataset using a Non-fitness Evolutionary and Adaptive Local Neighbourhood Search Algorithm.	(J): Expert Systems and Applications	Under review (2022)
7	Thau Soon Khoo and Babrdel Bonab Mohammad, "Solving Dynamic Vehicle Routing Problem with Time Windows: A Non-Fitness and Unified Approach"	(J): Transportation Research, Part E: Logistics and Transportation Review	Under review (2022)
8	Thau Soon Khoo and Babrdel Bonab Mohammad, "A Non-fitness Parallel Adaptive Approach for Solving Multi-objective Dynamic Vehicle Routing problem with Time Windows"	(J): Omega	Under review (2022)
9	Thau Soon Khoo and Babrdel Bonab Mohammad, "Solving Dynamic Vehicle routing problem with Time Windows using Real Dataset using Non-Elitist	(J): IEEE Transactions on	Under review (2022)

	Evolutionary Parallel Adaptive Local Neighbourhood Search Algorithm.”	Evolutionary Computation	
10	Thau Soon Khoo and Babrdel Bonab Mohammad, “Solving Multi-Objective Dynamic Vehicle routing problem with Time Windows using MOVRPTW Dataset using non-elitist Adaptive Genetic Local Neighbourhood Search Algorithm.”	(J): IEEE Transactions on Cybernetics	Under review (2022)

1.7. Summary

This chapter explains the motivation behind the research, the reasons for performing this research, the methodological approach to conduct the research, the specific research area to be conducted, the organization of thesis content into chapters, and the publications deriving from this research.

CHAPTER 2

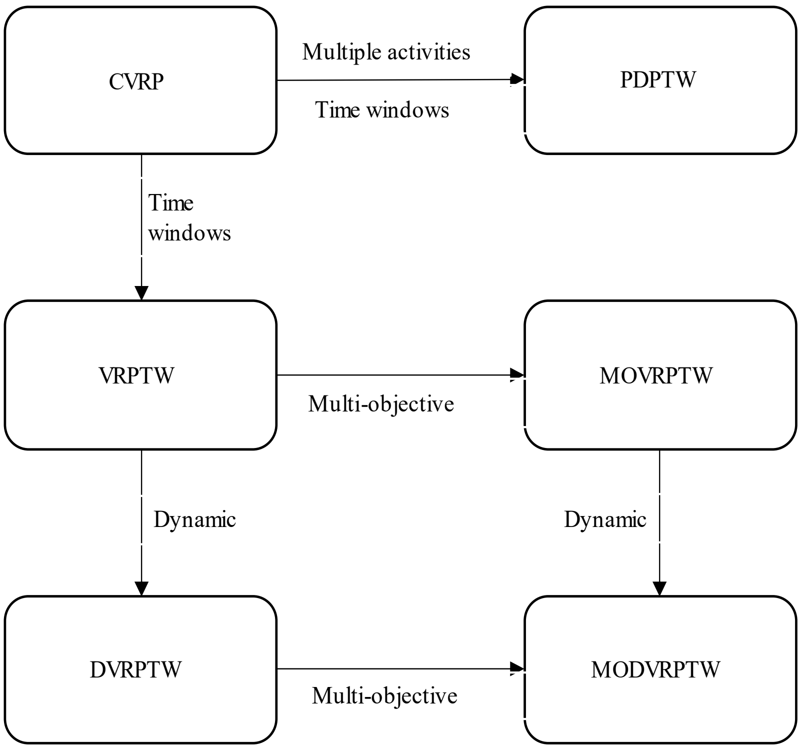
LITERATURE REVIEW

2.1. Introduction

This chapter presents and reviews the popular variants of VRP that are related to this research. It explains the underlying problems and constraints associated with each variant that led to the ultimate variant, namely the MODVRPTW. The popular algorithms, the definition of problems, and constraints are explained in each variant. The evolutionary algorithm and local neighbourhood search algorithm are seemingly popular among the variants are explained in length. It also explains the characterizations, techniques, and algorithms used in multi-objective optimization such as the Pareto optimality, multi-objective evolutionary solutions, multi-objective evolutionary algorithms, and quality indicator to assess the multi-objective algorithm.

2.2. Taxonomy of Vehicle Routing Problems

Vehicle Routing Problem (VRP) is an NP-hard problem (Yu *et al.*, 2017). This means the solution cannot be obtained within a reasonable time using exact solutions if an instance used has a large customer size. Therefore, it is important to study the method of solving this large-scale customer size to obtain a near-optimal solution. VRP has many variants. Each of these variants has its problems. Figure 2.1 shows the VRP variants and their relationships.



Legend
CVRP – Capacitated vehicle routing problem
VRPTW – Vehicle routing problem with time windows
MOVRPTW – Multi-objective vehicle routing problem with time windows
DVRPTW – Dynamic vehicle routing with time windows
MOVRPTW – Multi-objective vehicle routing problem with time windows
PDPTW – Pickup and delivery with time windows

Figure 2.1: VRP variants

In this figure, the Capacitated VRP aims to achieve near-optimal routes if the total demand of customers does not exceed the vehicle capacity. In VRPTW, the near-optimal routes are calculated within depot availability time and customer availability time (customer time windows). In MOVRPTW, the objective is to achieve two or more objective functions such as both the least number of vehicles and the least total travelled distance. In DVRPTW, the objective is like VRPTW except for the problem that some customer information is not available during the planning time instead it is only available during execution time.

In MODVRPTW, this problem is similar to MOVRPTW but of dynamic nature. Each of these variants is explained in detail in the following sections. Several types of algorithms can be used to obtain solutions. They are brute-force, heuristic, and metaheuristic. However, it takes a longer time to derive solutions using brute force if it is using a large-scale dataset. Therefore, brute force is best used for small-scale problems. A suitable type of algorithm is to use the heuristic or metaheuristic that will attain the solution within a reasonable time, and most solutions obtained are near-optimal.

2.2.1 Capacitated Vehicle Routing Problem

CVRP is one of the popular VRP variants (Altabeeb *et al.*, 2021) and is extensively studied (Yu *et al.*, 2017). It operates on static customer information, which means that all data about customer information are known during the planning time. It aims to determine the least routing cost using the homogenous vehicle. The following defines the CVRP model (Yu *et al.*, 2017):

- Directed graph (G) is (V, A) where vertices set is $V = \{0, . . ., N\}$ and arcs set is $A = \{(i, j)\}$.
- There is a list of customers denoted as $V = \{1, . . ., N\}$ where $1 \ldots N$ represent customers and 0 represents a depot.
- There is a fleet of vehicles (K) and each vehicle has a capacity (Q). The demand of customer i is q_i, where the demand is between $0 < q_i < Q$.

- There is a cost matrix $C = c_{ij}$, in which c_{ij} is the travel cost between customer i and customer j.

The objective function is to minimize the total travelled distance by the vehicles. Euclidean distance (2.1) is used to calculate the distance between customer $i(v_i)$ to customer $j(v_j)$:

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2.1}$$

There are some hard constraints listed as follows:

- The customer can only be serviced by one vehicle.
- The total demand of all customers on a given route must not exceed the loading capacity (Q) of the vehicle.
- There is only one depot. All vehicles begin and end at that depot.

Earlier work on the VRP was based on exact algorithms. However, the scale and complexity of VRP have increased over the year due to the dynamic economic climate and competition. This has led to an evolution in the adoption of algorithms. Table 2.1 categorizes two types of algorithms (Talbi, 2007) that are used to approach VRP, and the assessment of each algorithm is shown in Table 2.1.

Table 2.1: Assessment of the Usage of Algorithms on VRP

Category	Algorithm		Assessment
Exact algorithm	Branch and X method (P. Augerat, J.M. Belenguer, E. Benavent, A. Corber´an, D. Naddef, 1995)		- Time-consuming to find a moderately optimal solution for a small-scale customer size.
	Dynamic programming (Kok, Hans, Schutten, & Zijm, 2010)		- Take up more memory, suitable for reasonable size problem. - Unnecessary memory utilization.
	Set partitioning formulation (Baldacci, Mingozzi, & Roberti, 2012)		- Use an exponential number of variables.
	Integer programming algorithm (Andres Figliozzi, 2012)		- Time-consuming to calculate and complex.
	Mixed-integer linear programming (Çetinkaya, Karaoglan, & Gökçen, 2013; Urbanucci, 2018)		- Risk of the high dimensionality of the problem.
	Traditional heuristic	Saving heuristic (Solomon, 1987b; Wang & Zhou, 2016)	- Fast computation but hard to generate the high-quality solution.
		Sweep algorithm (Garcia-Najera & Bullinaria, 2011; Panagiotis P. Repoussis, Tarantilis, & Ioannou, 2009)	- Small-scale dataset.
		Greedy algorithm (Kirci, 2016; Suárez & Anticona, 2010)	- Straightforward and efficient. - No guarantee can solve the problem.
	Metaheuristics	Ant algorithm (Y. H. Huang, Blazquez, Huang, Paredes-Belmar, & Latorre-Nuñez, 2019)	- Not efficient dealing with large scale. - Improper selection of parameters may lead to a non-optimal solution.
		Artificial immune algorithm (Hassen, Tounsi, & Bachouch, 2019; Shukla & Jharkharia, 2013)	- Higher convergence rate. - Hard to obtain the global optimal solution.
		Bee colony (Szeto, Wu, & Ho, 2011; Yazdani & Meybodi, 2014)	-Slow convergence speed and Improper exploitation ability in solving a complicated problem.
		Cultural algorithm (Farrokhi-Asl & Tavakkoli-Moghaddam,2016; Xue, 2020)	- Large dimensionality of data premature convergence.
		Coevolutionary algorithm (Farrokhi-Asl & Tavakkoli-Moghaddam, 2016; Xue, 2020)	- CEA pathologies can cause the ability to find good solutions

		Co-variance matrix adoption evolution strategy (Nand, Sharma, & Chaudhary, 2021; Vidal, Crainic, Gendreau, Lahrichi, & Rei, 2012)	- Not performing under low-dimensional functions, separable functions with or without negligible - Dependencies between design variables among others
		Differential evolution (Krömer, Abraham, Snasel, Berhan, & Kitaw, 2013; Mingyong & Erbao, 2010)	- Convergence is unstable in the last period. - Easily trap into local optimal.
		Evolutionary programming (Bräysy, Dullaert, & Gendreau, 2004; Nagata, 2007)	- Difficult parameter tunning. - No guarantee of convergence.
		Evolution strategies (Mester, Bräysy, & Dullaert, 2007; P P Repoussis, Tarantilis, Bräysy, & Ioannou, 2010)	- Convergence into bad local optimal.
Heuristic Algorithm		Genetic algorithm (Baker & Ayechew, 2003; Nazif & Lee, 2012)	- Tendency to converge into local optimal. - Terminating criteria is not clear if the best individual only compares to other individuals.
		Great deluge (Dueck, 1993; Saputra, Muklason, & Rozaliya, 2020)	- Speed of convergence. - A problem-based parameter setting.
		Guided local search (Kilby, Prosser, & Shaw, 1999; Tarantilis, Zachariadis, & Kiranoudis, 2008)	- Not easy to decide on a feature to penalize.
		Genetic programming (Gulić & Jakobović, 2013; Liu, Mei, Zhang, & Zhang, 2020)	- No guarantee of finding an exact or acceptable solution. - Prematurely converge upon a local optimum. - Performance depends on problem complexity.
		Greedy adaptive search Procedure (Parreño, Alvarez-Valdes, Oliveira, & Tamarit, 2010; Tchapnga-Takoudjou, Deschamps, & Dupas, 2012)	- Time consuming - Converging to local optima by limiting search space.
		Iterated local search (Merz and Huhse, 2008)	- Slow convergence - Easily trap in a local optimum
		Neural network algorithm (Merz and Huhse, 2008)	- Slow convergence - Easily fall into local optimum
		Particle swarm optimization (Merz and Huhse, 2008)	- Premature convergence
		Simulated annealing (Merz and Huhse, 2008)	- Poor solution when the problem is large. - Tuneable parameters must be carefully chosen.
		Tabu search (Merz and Huhse, 2008)	- Time consuming and depend on the initial solution.

	Variable neighbourhood search (Merz and Huhse, 2008)	- Lack of memory

2.2.2 Vehicle Routing Problem with Time Windows

VRPTW is an important variant of VRP and extensively studies combinatorial optimization problems (Qi *et al.*, 2015a). VRPTW follows the similar objective and constraint of VRP but includes some of the following (Zhang, Yang, and Weng, 2018):

- The time window constraints denote a predefined time interval for the customers. This is also known as customer availability time. The customer availability time has the customer's earliest availability time and the latest availability time. If the vehicle arrives before the time window, it will have to wait until the customer's earliest availability time is reached. The customer will not be able to serve if the vehicle arrives after the customer's latest availability time.
- There is also an allocated service time to service the customer.

2.2.3 Pickup and Delivery with Time Windows

Another generalization of VRPTW is the PDPTW (Baldacci *et al.*, 2010) which consists of pickup and delivery activities. The objective is to achieve the least number of vehicles used and the least total travelled distance. Each route has a set of pickups ($P = \{p_1 \ldots p_n\}$) with the corresponding deliveries ($D = \{d_1 \ldots d_n\}$) at the respective customers locations ($V = \{v_0 \ldots v_n\}$). Each pickup must precede each delivery on the same route and execute within the vehicle capacity and each activity has the given time windows or customer availability time ($e_1 ... e_n$, $l_1 ... l_n$). There is a service duration ($S = \{s_1 ... s_n\}$) attached to each pickup and delivery activity. However, there are no pickup and delivery activities at the depot (v_0). The following define the model (Holborn, Thompson, and Lewis, 2012):

- Each vehicle must start at a depot and must perform at least one pickup and delivery before returning to the depot.
- Each pickup must have the corresponding delivery activity.
- All vehicles have a similar capacity (Q), and each vehicle must not exceed its capacity.
- Each vehicle must wait if they arrive early at the customer location, and they must not service if it arrives beyond the customer's latest availability time.
- There is only one depot. All vehicles begin and end at that depot.
- Within each route, the delivery cannot take place if the pickup is not initiated.

2.2.4 Multi-objective Vehicle Routing Problem with Time Windows

Another increasing research trend on the VRP variant is the MOVRPTW. The trend is due to extend of the single-objective into multi-objective (Baños *et al.*, 2013). In MOVRPTW, the objective is to attain the least number of vehicles used and the least total travelled distance.

2.2.5 Dynamic Vehicle Routing Problem with Time Windows

DVRPTW is an extension of VRPTW. To the best of our knowledge, the research in this area is quite limited. However, DVRPTW is regarded as a practical and important (Necula, Breaban, and Raschip, 2017) problem to be solved. DVRPTW includes a dynamic nature of the problem in which some of the customer information was never revealed during the planning period. The information of these customers was only updated on an ongoing basis during the execution time. This simulates the real-life scenario of the problem. DVRPTW encompasses all constraints established in VRPTW plus the dynamic nature of the customer's appearance. DVRPTW is sometimes referred to as online or real-time VRP.

2.2.5.1 General Definition

The first reference to the dynamic vehicle routing problem equivalent first appeared in a single-vehicle dial-a-ride problem (DARP) by Wilson and Colvin (Wilson and Colvin, 1977). DARP is based on pickup and delivery requests between the origin of the location and the destination. The aim is to achieve a minimum distance cost and accommodate as many customers as possible under a set of constraints. A typical example of DARP is the door-to-door transportation of elderly or disabled people. Over the years, the technological advances, and the industrial revolutions (IR) 4.0 have caused data to be grown immensely, smartphone and mobile devices have become a daily necessity, tracking, and online ordering has become a norm, and tracking in real-time manner to stay competitive. This means that dynamic or real-time requests are ubiquitous in the delivery and pickup orders, and there are important problems to be solved.

In contrast to static routing, dynamic routing involves new challenges such as deciding the worthiness of the given route plan which can increase the complexity of the decisions. In some instances, such as courier service, the delivery company may reject customer requests as it may increase the cost of delivery or affect its service guarantee. Also, it must be able to decide whether to divert a moving vehicle to a nearby request for additional revenue, which requires rapid support and online information received from the service provider regarding the position of the vehicle. Dynamic routing may differ in its objective function compared to static routing which only focuses on minimizing travelling distance, the number of used vehicles, or both. It can emphasize service level, throughput (maximization of customer requests), maximization of revenue, minimizing the delay between the request arrival and its services, among other objectives. Also, dynamic routing may not compromise decision quality (delay in accepting or rejecting the customer request decision) for servicing customers.

A typical DVRPTW can be illustrated in Figure 2.2 in which the vehicle, static, and dynamic order changes states during the planning, execution, and completion stage in the DVRPTW. At the planning stage, an instance of the DVRPTW consists of a central depot, 12 static customers requested before the journey starts. The plain house represents static customer order, the black house represents dynamic customer order, and the blurred house represents rejected customer order as shown in Figure 3.1(a). In this stage, a set of static customers were known in advance before the journey starts.

During the execution stage (*time* = t_1), the routes are planned, and 3 vehicles are assigned to deliver these orders to a set of static customers. In Figure 3.1(b), the three dynamic orders (black houses) are received intermittently during an interval time (time = t_1) while the vehicles are en route to serve other customers as shown in orange dotted lines. However, two dynamic requests (orders from customers 14 and 15) were rejected due to requests coming in too late for the vehicles to accommodate their requests (not connected by any lines) and one dynamic customer request (customer 13) can be accommodated due to it appearing before the vehicle passes the customer

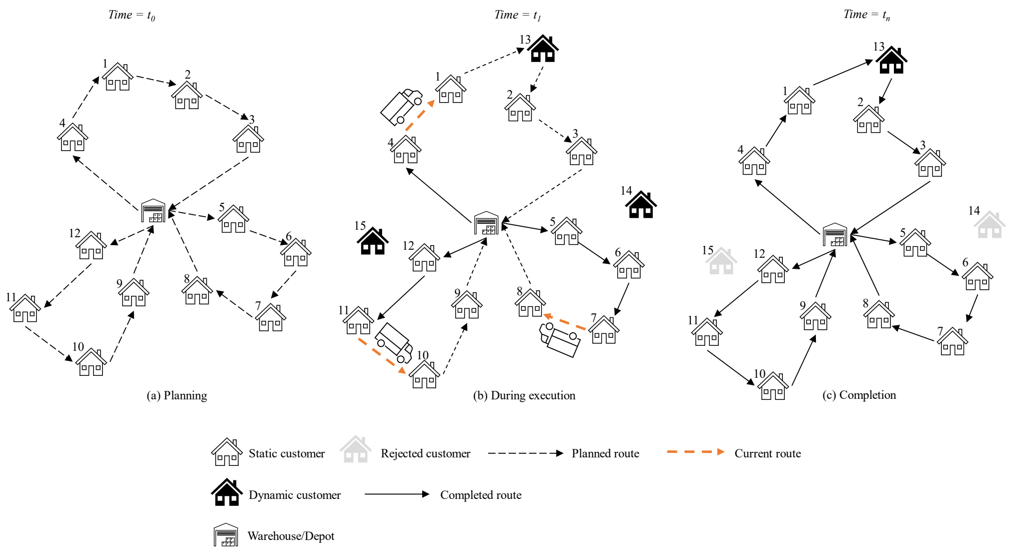


Figure 2.2: Dynamic Vehicle Routing Problem with Time Windows

location, which may seem economical from the distance standpoint. After the journey, all vehicles return to the depot, 13 customers are served and 2 customer requests are rejected, as shown in Figure 3.1(c).

2.2.5.2 Mathematical formulation

DVRPTW is defined as a complete graph $G = (V, E)$ where V represents a set of vertices that consists of a depot node v_0 and customer nodes ($v_1...v_n$. $E = \{(i, j)$: i, j V, i j represents a set of arcs, each representing the known travel cost (t_{ij}) between node i and j (Chen *et al.*, 2018). Static customers are customers' information explicitly available before planning or execution. It is denoted as V_s. For dynamic customers V_d, the customers' information is available during execution. Therefore $V = V_s \cup V_d = \{v_1, v_2, . . ., v_n\}$ represents all customers. Each customer $v_i \in V'$ *is* represented as a vector $v_i = (x_i, y_i, q_i, s_i, e_i, l_i, T_i, b_i)$ is denoted as the location of customer v_i (x_i, y_i), customer demand (q_i), customer service time (s_i), earliest availability time (e_i), latest availability time (l_i), request service time (T_i) and begin service time (b_i). For static customers, the service time is represented as $T_i = 0$. A vehicle must wait if it arrives early at the customer v_i before the e_i. Each arc $(i, j) \in E$ is associated with a cost of travel distance (d_{ij}) or travel time (t_{ij}). A customer can only be served once by a vehicle k on a single route. The aggregate demands on that route must be less than or equal to the vehicle loading capacity (Q_k). The binary variable $\mathcal{E}_{ijk} = 1$ if arc (i, j) is travelled by vehicle k, and 0 otherwise. Another binary variable $X_k = 1$, if vehicle k is used and 0 otherwise. The DVRPTW model is listed as follows:

Minimize $\sum_{(i,j)\in E} \sum_{k\in K} d_{ij} . \varepsilon_{ijk} + \gamma \,.\, \sum_{k\in K} X_k . g_k$ (2.2)

Where g_k is the fixed cost of vehicle k but subject to:

$\sum_{i\in V} \varepsilon_{ijk} = \sum_{i\in V} \varepsilon_{ijk}\ j \in V', k \in K$ (2.3)

$\sum_{k\in K} \sum_{j\in V} \varepsilon_{ijk} = 1\ i \in V'$ (2.4)

$\sum_{j\in V} \varepsilon_{0jk} = \sum_{i\in V} \varepsilon_{i0k} = 1\ k \in K$ (2.5)

$\sum_{i\in V'} \sum_{i\in V} q_i \varepsilon_{ijk} \le Q_k k \in K$ (2.6)

$a_i = b_{i^-} + s_i + t_{i,i-1}\ i \in V'$ (2.7)

$b_i = \max\ \{ a_i, e_i \}$ (2.8)

$e_i \le b_i \le l_i$ (2.9)

$\varepsilon_{ijk}, X_k \in \{0,1\}$ (2.10)

The objective function (2.2) is to minimize the total travelled distance and the number of vehicles where γ is a coefficient. Constraint (2.3) is a flow conservation constraint. The in-degree of each customer should be equal to the out-degree, which is at most one. Constraint (2.4) represents that each customer must be visited by only one vehicle. Constraint (2.5) ensures that each route starts and ends at the central depot. Constraint (2.6) represents the capacity of the vehicle. Constraints (2.7), (2.8), and (2.9) represent time windows. Lastly, constraint (2.10) imposes restrictions on the decision variables.

2.2.6 Multi-objective Dynamic Vehicle Routing Problem with Time Windows

MODVRPTW is the multi-objective form of DVRPTW. This means their optimal solution has several objectives that it wants to accomplish.

Generally, it could be reducing the number of used vehicles, total travelled distance, and rejection rates. To our knowledge, there are only a handful of studies on this problem. Despite that, the study of MODVRPTW may appear in different forms such as (Tang and Hu, 2005; Ghannadpour *et al.*, 2014; Kaiwartya, Kumar, D. K. Lobiyal, *et al.*, 2015).

2.2.7 Information Characteristics

The information available to real-world applications can be defined into two important dimensions that are the evolution and quality of information (Psaraftis, 1980a). The evolution of information concerns about information might experience sudden change during the execution of the routes such as the arrival of new customer requests, whereas the quality of information refers to uncertain data availability such as the rough estimate of the real demand of that customer. The nature of vehicle routing can exist in two fashions either static or dynamic. For example, VRP with stochastic demand (VRPSD) can be viewed in both fashions. Hence, this dimension of the real-world application can be further explained in Table 2.2.

Table 2.2: Taxonomy of vehicle routing problem (Pillac *et al.*, 2013)

		Information quality	
		Deterministic information	Stochastic information
Information evolution	Information is known beforehand	Static and deterministic	Static and stochastic
	Information changes over time	Dynamic and deterministic	Dynamic and stochastic

Statically, it is seen as a set of predetermined routes that may change slightly during execution (Bertsimas and Simchi-Levi, 1996; Gendreau, Laporte and Séguin, 1996), and dynamically, the vehicle routes are constructed in an ongoing fashion based on the state when the vehicle is in an idle state.

In static and deterministic problems, all customer information is known in advance, and vehicle routes do not change during execution. These problems have been extensively studied (Kritikos and Ioannou, 2010; Schneider, 2016; Utama *et al.*, 2020).

In static and stochastic problems, the information is partly unknown, which is the random variables, and the realization is known during the execution of the routes. In addition, the routes are known in advance and a small change is allowed subsequently such as skipping a customer and a trip back to the depot. In this problem, the three most studied areas are the stochastic customer (Bertsimas, 1988; Waters, 1989), stochastic times (Laporte, Louveaux and Mercure, 1992; Kenyon and Morton, 2003; Verweij *et al.*, 2003), and stochastic demands(Dror, Laporte and Trudeau, 1989; Secomandi, 2000; Gendreau, Laporte and Potvin, 2002; Christiansen and Lysgaard, 2007; Mendoza, Medaglia and Velasco, 2009; Secomandi and Margot, 2009; Mendoza *et al.*, 2011).

In dynamic and stochastic problems, part or all of the information may not be known in advance and dynamically revealed during routes executions. In dynamic and deterministic problems, not all information is known in advance but is revealed during route execution. In this thesis, we focus on this problem together with multi-objectives, and often, this problem may refer to as online, dynamic, or real-time in other works of literature. This dynamic information provides stochastic knowledge and vehicle routes can be re-planned continuously.

The level of dynamism of the problem can be categorized into two dimensions. They are the frequency of changes and the urgency of customer requests. The frequency of changes is a new information availability rate, and the urgency of customer requests is the interval time between an appearance of a new customer and its expected service time. There are many metrics used to measure the dynamism of a problem, such as a ratio between the number of dynamic customers (n_d) and the total number of customers (n_{total}) (Lund, Madsen, and Rygaard, 1996), the disclosure date, and the time windows of the dynamic customers (Larsen, 2000).

2.3. Evolutionary Algorithm

An evolutionary algorithm (EA) is a stochastic population metaheuristic. It has been applied to many real and complex problems such as multi-objective, highly constrained problems, and multimodal, among others. EA is one of the most studied population metaheuristics and has been successfully implemented in many areas such as combinatorial optimization, engineering design, data mining, machine learning, artificial intelligence, among others. Because of this reason, they are considered evolutionary computation (EC).

There are different schools of evolutionary algorithms accumulated over the past 40 years. The four most common are a genetic algorithm (GA), evolution strategies (ES), evolutionary programming (EP), and genetic programming (GP). Other models include an estimation of distribution algorithms (EDA), differential evolution (DE), coevolutionary algorithms (CEA), and cultural algorithms (CA). EC represents the evolution of species. Figure 2.3 represents a typical evolution in EC.

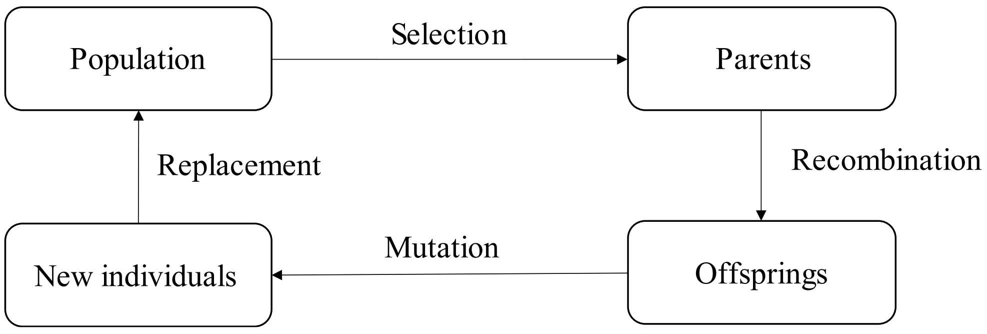


Figure 2.3: An Evolution(generation) in Evolutionary Algorithms

Initially, the individuals in the population are generated randomly. Each individual in the population represents and encodes a solution to the problem. Each fitness value is calculated and associated with the individual. Two individuals are chosen based on the selection paradigm and perform a crossover to generate offspring. These offspring are mutated into new individuals which replace the underperformed individuals in the population. These steps continue until the termination criteria are met. The surviving and most optimal individual after the generation steps is selected as output.

2.3.1 Genetic Algorithms

Genetic algorithms (GA) were developed by J. Holland in 1970 (Holland, 1992). GA consists of four common steps as shown in Figure 2.4.

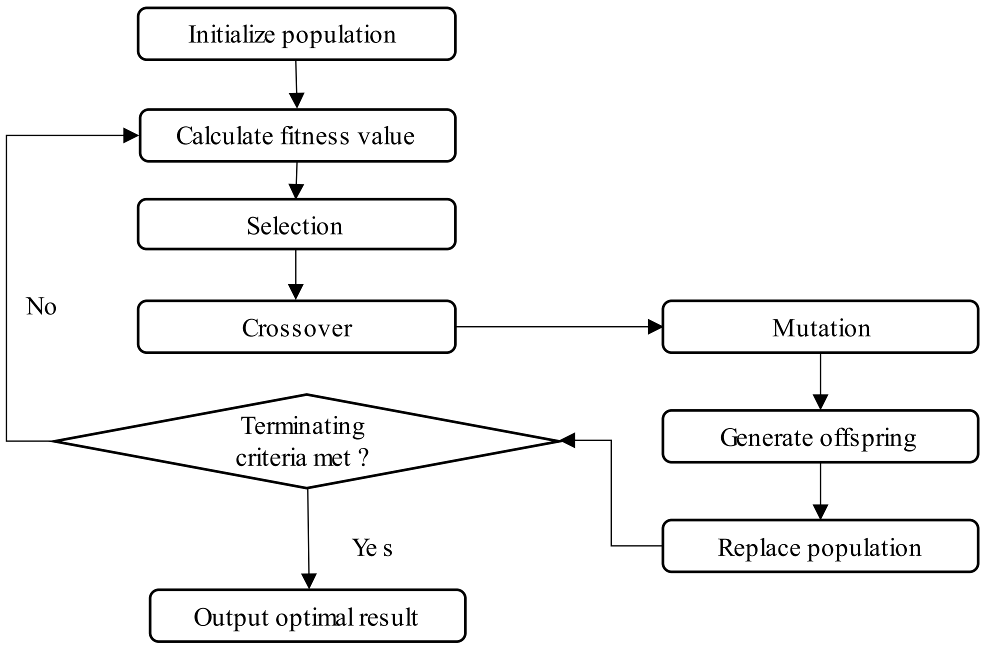


Figure 2.4: Genetic Algorithm

They are selection, crossover, mutation, and replacement. In the selection step, GA uses probabilistic selection to proportionately select individuals for crossover. There are a variety of selection methods that the selected individuals can use to select parents. These parents are subsequently crossover to generate offspring which is mutated into new individuals to be output in the population.

GA use crossover and mutation operator to modify the individual to promote diversity.

2.3.2 Evolution Strategies

Evolution strategies (ES) were developed by Rechenberg and Schewefel in 1964 at the Technical University of Berlin (Rechenberg, 1965; Vent, 1975) as shown in Figure 2.5.

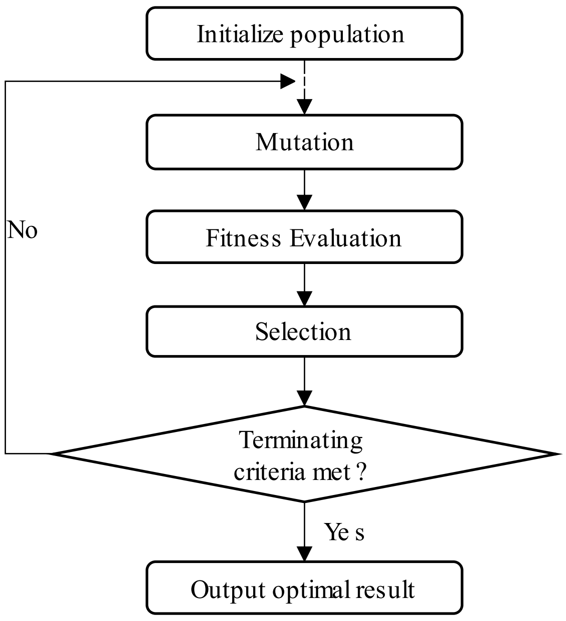


Figure 2.5: Evolution Strategies Algorithm

ES begins with the initialization of individuals in the population. Individuals are selected as a parent. The cycle iterates when offspring are generated from the selected parents and the offspring replace the individual in the population. ES uses mutation, recombination, and selection operators and is applied iteratively until a termination criterion is met. The selection operator is based on fitness ranking. Recombination can be discrete (uniform crossover) or intermediary (arithmetic crossover). The crossover is rarely used. An individual (solution) in the ES consists of floating decision variables and uses some other parameters for search guidance.

2.3.3 Evolutionary Programming

Evolutionary programming (EP) was introduced by Lawrence J. Fogel in 1960 while serving the National Science Foundation (NSF) (Lawrence J. Fogel, Alvin J. Owens, 1966) using Finite State Machine (FSM) at the early stage. The basic EP flow chart is shown in Figure 2.6. It uses a stochastic optimization strategy and focuses on the relationship between the parents and offspring. It is different from the genetic algorithm, EP simulates the evolution of species, unlike GA, which simulates genes. EP linking the species in its evolutionary steps. This means EP embraces evolution behaviour between parents and offspring, or good offspring can survive and not consider parents. EP uses the fitness value to select the offspring to compare. Compared to GA which uses the fitness value to select a parent. It is an approach that iteratively generates an appropriate solution using a fitness function in a stationary or non-stationary environment.

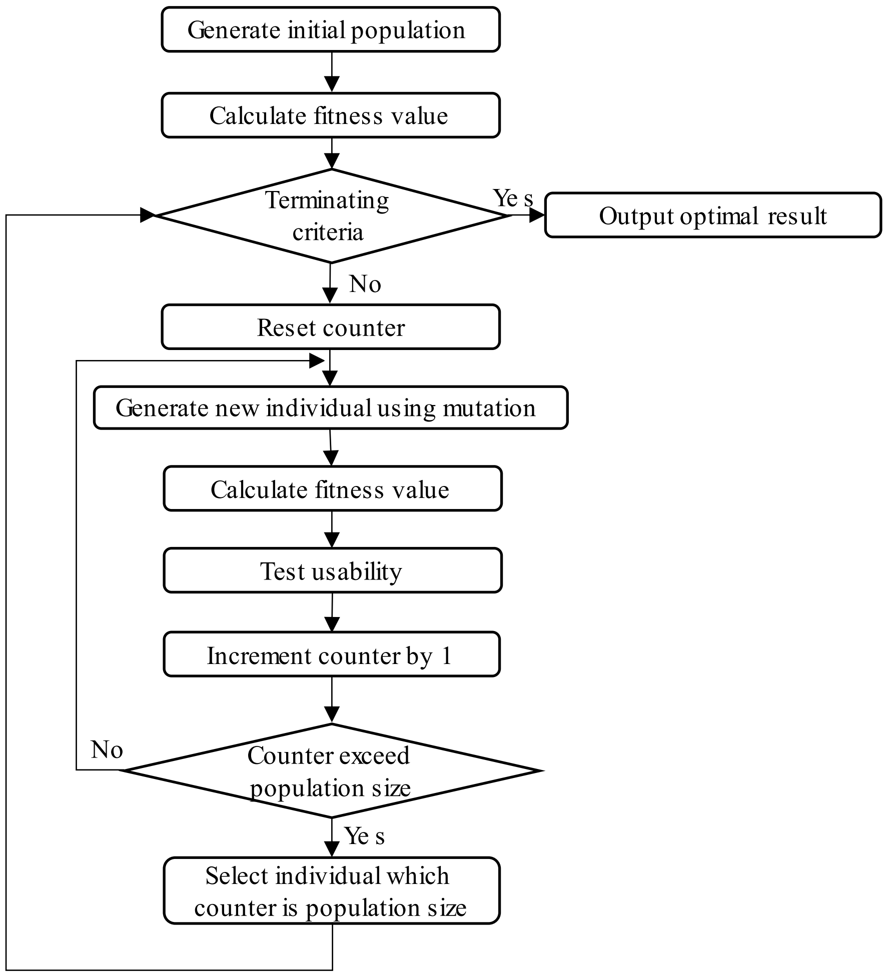


Figure 2.6: Evolutionary Programming Algorithm

2.3.4 Genetic Programming

Genetic programming (GP) is the work of John Koza (student of John Holland, founder of GA) which nicely coincides with his ongoing research on GA. First, it begins with generating the initial population as shown in Figure 2.7. Next, the fitness value is calculated and associated with each individual. The population contains individuals. Each individual is probabilistically selected from the population based on the fitness value. In this selection, the performing individual is highly likely to be selected over the worst-performing individuals.

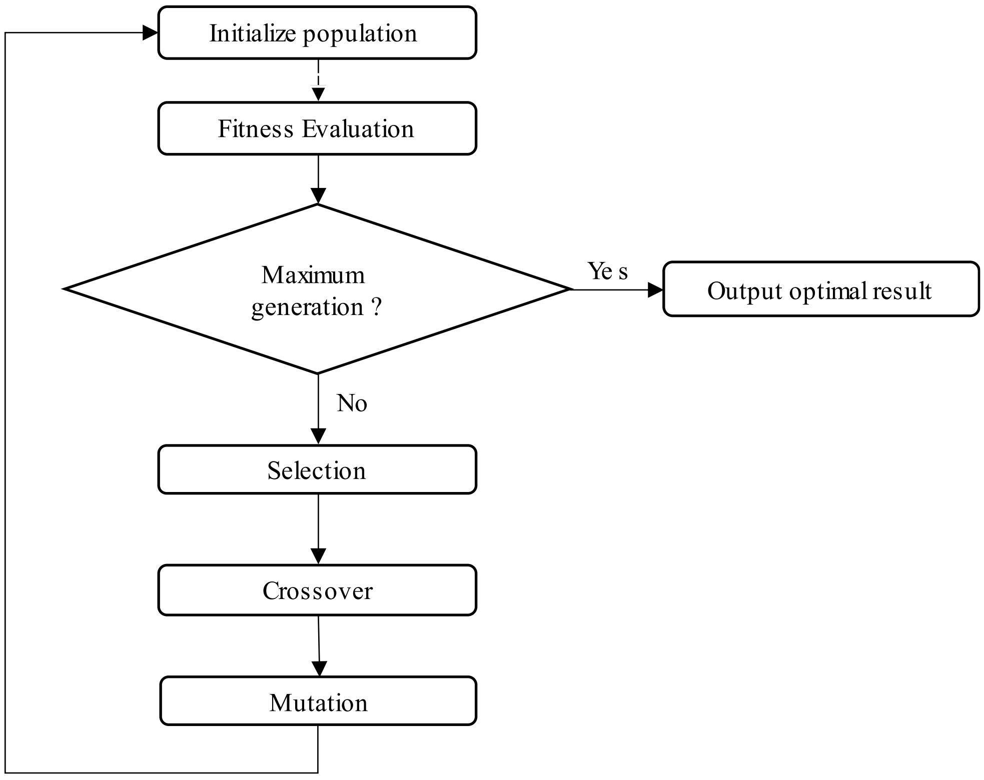


Figure 2.7: Genetic Programming Algorithm

However, the performing individual is not necessarily selected, and the worst-performing individual is not necessarily avoided. Next, the selected individuals perform crossover to generate offspring. These offspring replace the individuals in the population if the offspring result is better. This process iterates until the terminating criteria are met (number of generations). Finally, it outputs the best individual.

2.3.5 Other Evolutionary Algorithms

Estimation of Distribution Algorithms (EDA) is an evolutionary algorithm that uses a pool of individuals to perform beam searches (Mühlenbein and Paaß, 1996). EDA performs an evolutionary mechanism using estimation and simulation of the joint probability distribution. Initially, a population of individuals is generated. EDA consists of 3 main steps that are executed iteratively. Each iteration represents a generation. The first step is to select a subset of the best individuals. The second step is to learn the selected individuals, and the final step is to generate new individuals using the distribution model. In this manner, the population performance improves as more iterations are executed. The iterations are terminated after terminating criteria are met (several generations are reached or when the overall population performance does not improve). The basic EP flowchart is shown in Figure 2.8.

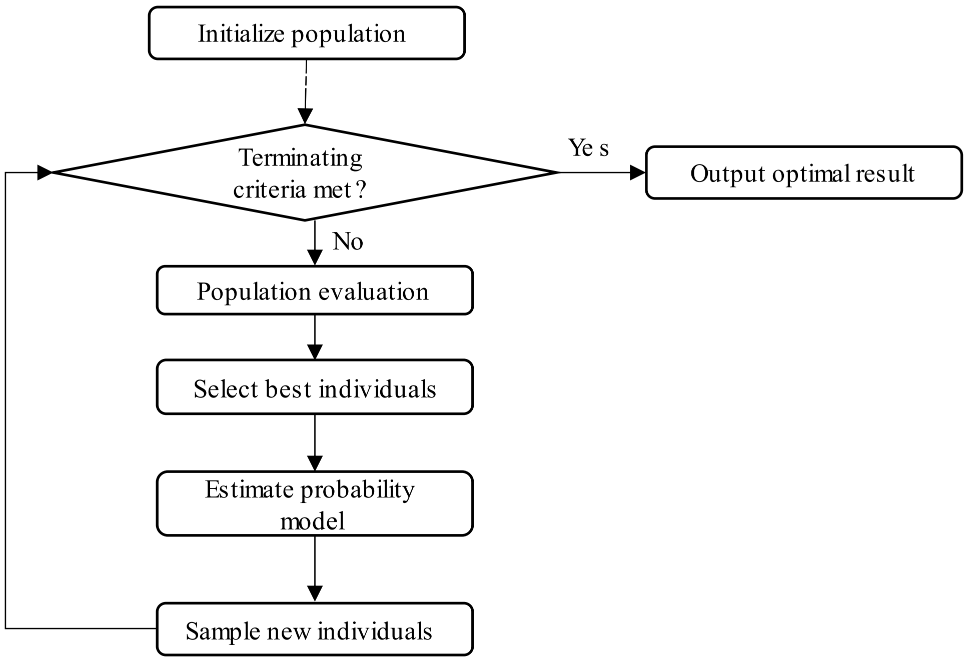


Figure 2.8: Estimation of Distribution Algorithm

Differential Evolution (DE) was discovered by Storn and Price (Storn, 1996; Storn and Price, 1997). It is a multi-faceted research area and appear in many application areas such as engineering, logistic, industrial engineering, among others. Figure 2.9 shows that the DE algorithm starts by initializing the population and the fitness of each individual is evaluated. This mutation process adds a weighted difference between the population vectors to produce a mutated vector. Next, the crossover mixes the mutated vector with the parameters of the target vector to produce the trial vector, which purportedly has better diversity. A selection process replaces the target vector with the trial vector, its offspring.

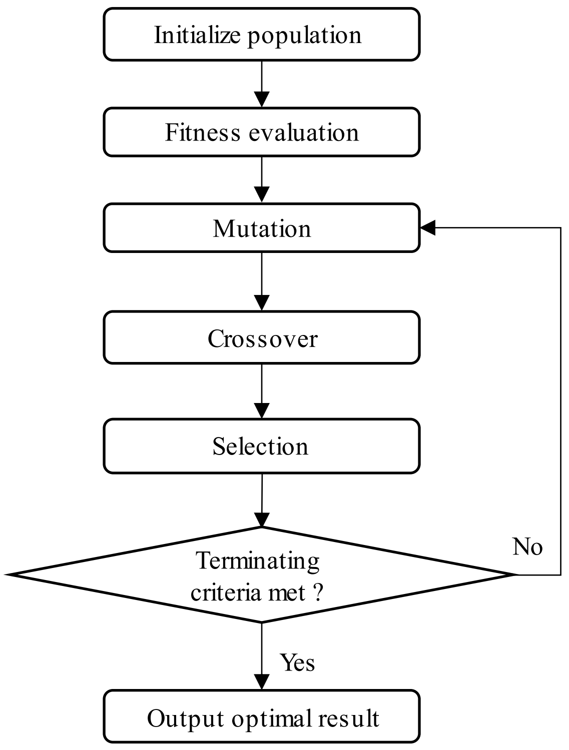


Figure 2.9: Differential Evolution

These iterations continue until the termination process is met. At each iteration, the DE first performs the mutations on the population to generate new solutions candidates.

2.3.6 Local Neighbourhood Search Algorithm

The Local Neighbourhood Search (LNS) algorithm is based on the concept of ruin and recreate (R&R) principle as formulated by Shrimpf et al (Schrimpf *et al.*, 2000a). Some other algorithms which have similar approaches are iterated local search (Stützle, 2006), large-step Markov chains (Martin, Otto, and Felten, 1991), variable neighbourhood search (Hansen, Mladenović and Pérez, 2010), and chained local optimization (Bouhmala, 2019). The basic principle is based on the removal part of an existing solution and repairs of the ruined solution. If these steps are performed repeatedly, high-quality solutions can be achieved. This way the generated solution escapes from a local optimum and find better solutions. Two ways to escape the local optimal is through exploration and exploitation. Exploration occurs when part of the solution is removed, and exploitation occurs when the ruined solution is repaired. This metaheuristic algorithm uses several ruin methods. In this thesis, we propose four ruin strategies and two recreate strategies (best insertion and regret insertion). The ruin strategies include critical procedure, related procedure, radial procedure, and random procedure. The recreate strategies contain best insertion procedure and regret insertion procedure.

2.4. Multi-objective Optimization

In the real world, many optimization problems have two or more objective functions. This is a common problem where the decision-maker (DM) or stakeholders want to strive for the best deal or find out whether it satisfies the requirements stated in a standard or recommended practice. However, there are cases where the objectives have contradicted each other. In this scenario, we are dealing with a set of trade-offs. This set of trade-offs is called a Pareto front. When the solutions are on the Pareto front, this means no other solutions in the search spaces are better than the solutions at the Pareto front. Many studies are being conducted on multi-objective optimization problems (MOP), particularly vehicle routing problems (Afsar Afsar, & Palacios, 2021; Huang, Li, Zhu & Qin, 2021; Kyriakakis, Marinaki, & Marinakis, 2021).

In mathematical terms, a general minimization of MOP (2.11) can be written as (Castro-Gutierrez, 2012) :

$$Minimize\ y = f(x) = (f_1(\bar{x}), f_2(\bar{x}), \ldots, f_n(\bar{x})) \tag{2.11}$$

subject to:

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \in X \tag{2.12}$$

where x_1 is the first decision variable, n is the number of decision variables, $\bar{x}$

(2.12) is the vector of decision variables, X is the feasible set and R^m is the decision space; $\bar{x} \in X \subset R^m$.

$$\bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_m \end{bmatrix} \in Y \qquad (2.13)$$

The objective vector is denoted by $\bar{y}$ (2.13). y_1 is the first objective function and m is the number of objective functions. $Y = f(X)$ is the objective feasible region and R^k is the objective space; $\bar{y} \in Y \subset R^k$. Therefore, in MOP, there is a multi-dimensional space for the objective functions and the decision variable space for the solution vector. This translates to every solution in the decision variable space, there is a point in the objective function space. Figure 2.10 shows the mapping of a decision space onto an objective function.

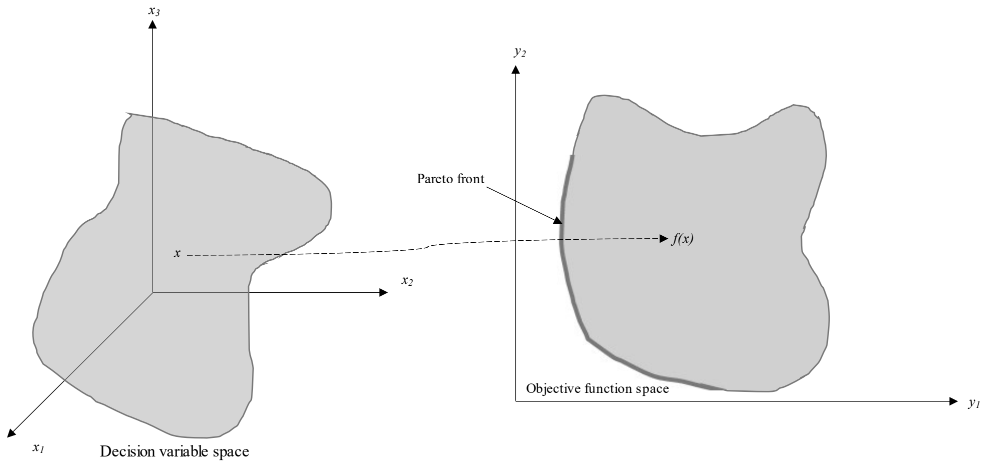


Figure 2.10: Mapping a decision space onto an objective function (Sheikh *et al.*, 2021).

Definition If there exists a solution $\bar{x}^{\wedge} \in X$ that minimizes all objective functions simultaneously, $\bar{x}^{\wedge}$ is an ideal vector. This means a feasible solution ($\bar{x}^{\wedge} \in X$) is an ideal solution (2.14) if there is no other $\bar{x} \in X$ and $i \in \{1,2, . . ., n\}$ such that:

$$f_i(\bar{x}) < f_i(\bar{x}^{\wedge}) \qquad (2.14)$$

2.4.1 Pareto Optimality

There is no unique solution but a set of solutions in the MOP. The set of solutions is found using the Pareto optimality concept (Ehrgott, 2005). The MOP global minimum (or maximum) problem is formally defined as follows:

Definition Given two decision vectors $\bar{x}_1$, $\bar{x}_2 \in X$, $\bar{x}_1$ dominates $\bar{x}_2$ ($\bar{x}_1 \prec \bar{x}_2$).

$$\forall i \in \{1, 2, . . . , n\} : f_1(\bar{x}_1) \leq f_1(\bar{x}_2) \qquad (2.15)$$

$$\exists j \in \{1, 2, . . . , n\} : f_1(\bar{x}_1) \leq f1(\bar{x}_2) \qquad (2.16)$$

Definition Given two decision vectors $\bar{x}_1$, $\bar{x}_2 \in X$, $\bar{x}_1$ is said to cover $\bar{x}_2$ ($\bar{x}_1 \preccurlyeq \bar{x}_2$) if $\bar{x}_1 \prec \bar{x}_2$ or $f(\bar{x}_1) = f(\bar{x}_2)$.

Definition A vector of decision variables $\bar{x}^{\wedge} \in X$ is non-dominated if there is no other $\bar{x} \in X$, such that $\bar{x} = \bar{x}^{\wedge}$

Definition The Pareto optimal set $P^{\wedge}$ is described as $P^{\wedge} = \{\bar{x} \in$ X : $\bar{x}$ is Pareto optimal }

Definition The Pareto front $PF^{\wedge}$ is defined as $PF^{\wedge} = \{\bar{y} = f(\bar{x}) \in Y : \bar{x} \in P^{\wedge}\}$

The main goal of MOP is to obtain the Pareto front (PF). The Pareto front consists of many points from a theoretical point of view. In practice, there is a limited number of usable approximate solutions. It is important to find solutions that are closer to the Pareto front and are uniformly spread. The closeness of approximate solutions to the Pareto front explains the high convergence of that solutions and this means the approximate solutions are closer to the Pareto front. The uniformly spread approximate solutions mean that the approximate solutions have a good exploration of the search space and there are no regions left unexplored.

2.4.2 Multi-objective Evolutionary Algorithm Solution Techniques

There are several ways to classify multi-objective problems. One way is to classify the techniques into three main approaches proposed by Adulbhan P and MT Tabucanon (Adulbhan P, 1980). Their approaches include the conversion of secondary objectives into constraints, the development of a single combined objective function, and treating all objectives as constraints. Another technique is to classify it into four approaches which are proposed by Hwang et al (Hwang, Paidy, Yoon, & Masud, 1980). Their classification is divided into four approaches: (1) no articulation of the preference data of the decision maker, (2) a priori articulation of the preference data, (3) progressive articulation of the preference data, and (4) a posteriori articulation of the preference data.

In multi-objective evolutionary algorithm (MOEA) approaches, it is classified into three main techniques as shown in Table 2.3 (Coello, Lamont, and Veldhuizen, 2006). They are as follows:

Priori techniques (Before the search): These techniques require decision-makers to define the relative importance of the MOP objective before any search.

Typically, this technique involves assigning weights to the aggregated sum of the objectives. The problem with this method is the poor objective prioritization. This happens when the decision maker's weight is greater than necessary, resulting in a more acceptable solution being missed. A priori techniques are divided into 3 main approaches which are listed as follows:

- **Lexicographic ordering:** The objectives are ranked in importance order. This means that the most important objective function is minimized first to get the optimum solution than other objectives with regards to the order of importance. If the order of importance is known, the objective function will be randomly selected. The weakness of this technique is that they prefer certain objectives due to the randomness in the process. This causes an undesirable population to converge to a particular part of the Pareto front rather than completely delineate it. However, this technique is simple to use and computationally efficient.

- **Linear aggregating function**: This linear fitness combination is a scalarizing approach and is easy to use. However, this approach does not find all Pareto front points of interest. These non-supported points are not supported in this approach which is not appear on the convex hull of the Pareto front. Another scalarizing approach is the weighted Tchebycheff model which supports the non-convex hull. Linear aggregating functions are easy to understand, implement, and computationally efficient for an easy problem domain, and the relative worth of each objective is known, quantifiable, and available in a short time of searching. The disadvantages are that if the Pareto front is non-convex, the portion of the front will not be found. However, the linear aggregating function is less common than other approaches namely the Pareto-based approach.

Non-linear aggregating functions: This approach can operate using either multiplicative approaches or target vector approaches. In the multiplicative approach, it is not popular due to the overhead in determining utility function and various conditions under which the objective functions must meet (Keeney and Raiffa, 1993). Simply put, the additional overhead does not warrant quality solutions. Target vector approaches are more popular than multiplicative approaches. It is even more useful if the decision-maker can specify the goals it wanted. Multiplicative approaches are simple, efficient, and maybe troublesome if the definition of a good nonlinear aggregation function is difficult compared to defining a linear aggregation function. In target-vector approaches, it is computationally intensive, may lead to additional problems (generate misleading selection pressure) and limit their applicability if goals are chosen in the feasible domain. Despite the drawbacks, the non-linear aggregating function can give good approximations of the Pareto optimal set.

A priori techniques do not desire general use unless the problems are multi-objective combinatorial optimization problems.

Progressive Techniques (During the search): these techniques require the decision-maker to direct the search. This approach uses the algorithm to prompt the user with questions to decide the search space. The drawbacks are the procedure lies in the decision-maker requirement time. These techniques are affordable if the execution time is not long.

Posteriori techniques (After the search): These techniques perform a regular number of solutions collections in the solution space. Based on the set of solutions, the decision maker will select the preferred ones. The main problem with these techniques is the solutions are difficult to find and computationally intensive. Posterior techniques are divided into five main approaches which are listed as follows:

- **Independent Sampling Techniques**: This sampling has reduced effectiveness. It uses some fitness combination techniques in which the weights assigned to each objective varied over several MOEA runs. These techniques are easy to use and efficient. However, the usefulness is quite limited, as the arbitrary weight combinations prevent the discovery of some solutions.

- **Criterion Selection Techniques**: Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1985) is an example of a criterion selection technique. The vector is defined as a vector of *k* objective functions. These techniques select a fraction of each succeeding population which is based on separate objective performance, and each fraction of the specific objectives is randomly chosen at each generation. Each objective function tends to converge closer to a local optimum. This concept is based on the *k* number of objectives, the number of subpopulations (*k*) of each population size (*M*) over the number of objectives (*k*) is generated. Each sub-population dedicates *k* objective functions for fitness assignment. The mating pool is generated using the proportionate selection operator. The sub-populations are shuffled to obtain a new population of size *M* which is executed by crossover and mutation operators subsequently. These techniques are simple to use and easy to implement. However, it is not able to generate a concave part of the Pareto front.

- **Aggregation Selection Techniques**: These techniques use a variant of other techniques to solve the MOP. These can be weighted sums (Ishibuchi and Murata, 1998), objective combinations (Loughlin and

Ranjithan, 1997), hybrid search approaches (Deb, 2001), among others. It uses different weights on different generations for each function evaluation. The weights may be assigned randomly, or the specified solutions may be evaluated, and even individuals may encode as genes so that the genetic operator acts upon them. The advantage of using these techniques is the ability to generate a set of solutions in a single run. The disadvantage is that certain Pareto front may be missed out if the technique of weighted sum is used and incur significant overhead if it uses both objective or constraints with hybrid search methods.

- **ϵ-Constraint Techniques**: This technique is a primary objective function selection and binds other objective functions with a separate allowable ϵ-constraint. The ϵ-constraint change to generate another point on the Pareto front to find elements in the Pareto optimal set. This technique is easy to implement but computationally intensive.

- **Pareto Sampling Techniques**: This technique uses population to generate several elements of the Pareto optimal set in a single stochastic computational run. It is more effective and robust than other techniques, but it has a scalability issue.

Table 2.3: MOEA Approaches (Coello, Lamont and Veldhuizen, 2006)

	Techniques	**Approaches**
Multi-objective Evolutionary Algorithm Approaches	Priori Techniques	Lexicographic
		Linear fitness combination
		Nonlinear fitness combination
	Progressive Techniques	Progressive techniques
	Posteriori Techniques	Independent sampling
		Criterion selection
		Aggregation selection
		Pareto-based selection
		Pareto rank and niche-based selection
		Pareto deme-based selection
		Pareto elitist-based selection
		Hybrid selection

2.4.3 MOEA Techniques

The first evolutionary algorithm for solving multi-objective optimization was dated in the late 1960s by Rosenberg (Rosenberg, 1970). However, the actual implementation of the multi-objective evolutionary algorithm (MOEA) is performed by David Schaffer which is mainly for solving machine learning problems (Schaffer, 1985).

Multi-Objective Genetic Algorithm (MOGA) was proposed by Carlos M. Fonseca and Peter J. Fleming using a variation of Goldberg's technique (Goldberg, 1989) that ranks certain individuals according to the number of individuals in the current population in which it is dominated. This algorithm is efficient and easy to implement, but sometimes it does not provide a diverse set of solutions. Also, their blocked fitness assignment type has a high likelihood to produce large selection pressure, which causes premature convergence (Goldberg and Deb, 1991).

Nondominated Sorting Genetic Algorithm (NSGA) is another Goldberg's ranking procedure variation that is proposed by N. Srinivas and Kalyanmoy Deb (Srinivas and Deb, 1994). NGSA is based on several layers of classifications in individuals. The population is ranked based on nondominated and these nondominated individuals are grouped into one category. The classified individuals are shared with dummy fitness values to maintain the diversity of the population. This group of classified individuals is then ignored and another layer of nondominated individuals in the population is being processed. These steps continue until all individuals in the population are classified. The NGSA technique uses a stochastic proportionate selection of the remainder. The individual in the first front will get more copies since they have the maximum fitness value, which allows for a better search of the Pareto front regions and results in convergence. NGSA is a highly inefficient algorithm because the classified individuals rapidly converge and are computationally intensive during the fitness sharing mechanism.

Niched-Pareto Genetic Algorithm (NPGA) is an MOEA tournament selection based on Pareto dominance proposed by Jeffrey Horn and his coworkers (Horn, Nafpliotis, and Goldberg, 1994). In NPGA, the two individuals are randomly chosen against a subset of the entire population. If one of the individuals is dominated and the other is not the non-dominated individual wins. If there is a tie, the fitness sharing will decide the tournament.

Pareto Archived Evolution Strategy (PAES) is proposed by Joshua D. Knowles and David W. Corne (Knowles and Corne, 2000). PAES has an evolution strategy that combines a historical archive that records some previously found non-dominated solutions. This archive is used as a reference set and is compared to each mutated individual. This method is used to keep diversity. This algorithm is less computationally intensive than traditional niching methods. Each solution is placed on a grid location based on objectives values. This algorithm is adaptive, and no extra parameters are required except for the number of divisions of the objective space.

Strength Pareto Evolutionary Algorithm (SPEA) was introduced by Eckart Zitzler and Lothar Thiele (Knowles and Corne, 2000). The algorithm uses an external archive (non-dominated solutions) and is copied to an external non-dominated set at each generation. A strength value is calculated for each individual in the external set. In this algorithm, the fitness of each member of the current population is calculated according to the strengths of all the non-dominated external solutions that dominate it.

Multiobjective Messy Genetic Algorithm (MOMGA) was introduced by David A. Van Veldhuizen and Gary B. Lamont (Van Veldhuizen and Lamont, 2000) by extending the messy GA (Deb, 1991) to solve MOP. MOMGA consists of the initialization phase, the primordial phase, and the juxtaposition phase. In the initialization phase, MOMGA uses a deterministic process to produce the building blocks of a certain building size, which is known as partially enumerative initialization. In the primordial phase, it performs tournament selection on the population and minimizes the size of the population if required. Finally, in the juxtaposition phase, it uses a cut and splice recombination operator by building up the population in the messy GA.

Pareto envelope-based selection algorithm (PESA) is proposed by Corne et al. (Corne, Knowles, and Oates, 2000). It consists of a small internal population and a large external population. During execution, it uses a hyper-grid division of phenotype space to maintain selection diversity. Selection diversity uses the crowding measure to enable solutions by using an archive of solutions that evaluate non-dominated vectors into the external population.

Micro-genetic Algorithm (micro-GA) is proposed by Carlos A. Coello Coello & Gregorio Toscano Pulido (C. A. C. C. Coello and Pulido, 2001; C. A. Coello and Pulido, 2001; Coello and Pulido, 2005). They introduced two memories that will be used in micro-GA. The first memory is the population memory which is served as a source of diversity and the second memory is the external memory which is to archives members of the Pareto optimal set. In population memory, it is divided into two parts which are replaceable and non-replaceable memories. Initially, micro-GA starts with the generation of a random population. This population inputs into the replaceable and non-replaceable portions of the population memory. At the beginning of each cycle, the initial population is taken from all population memory to achieve greater diversity. During each cycle, the micro-GA performs conventional genetic operators. After each cycle is completed, two non-dominated vectors from the final population are chosen and compared against the external memory. It replaces the population memory with the vectors. Finally, the replaceable part of population memory will have more non-dominated vectors. Some vectors from the population memory will be used in the initial population to start another new cycle.

Multi-objective Struggle GA (MOSGA) is introduced by Krus et al. (Andersson and Krus, 2001a, 2001b). MOSGA combines Pareto based ranking scheme with a struggle crowding genetic algorithm (Grueninger and Wallace, 1996). These algorithms have a similar pattern to the MOSGA as the two parents are chosen at random from the population. Also, it performs the normal crossover and mutation to create offspring. The offspring competes against the individuals in the population and replaces the individuals if it has a better ranking. This ranking method is like ranking used in MOGA.

Orthogonal Multi-Objective Evolutionary Algorithm (OMOEA) begins by defining a single niche in the decision space *x* (Ding *et al.*, 2003; Zeng *et al.*, 2005). This niche recursively split into a group of sub-niches until a terminating criterion is met.

General Multi-objective Evolutionary Algorithm (GENMOP) is designed at the US Air Force Institute of Technology (AFIT). It uses several operators when performing evolutionary operators (EVOPs) repeatedly to produce better solutions.

2.4.4 Quality Indicator

The MOP goal is to obtain the Pareto optimal front. Many multi-objective optimization problems are difficult to solve, and the results of these optimization problems approximate the Pareto front. The evaluation of approximation quality is also a MOP. There are many popular measures to compare the performance of MOEAs. This measurement is called Quality Indicators (QI), and the term 'performance metric' is referred to as a quantifiable difference between approximation sets. There are many QIs for measuring the approximation set quality (H Ishibuchi, Masuda, Tanigaki, & Nojima, 2015; M. Li, Yang, & Liu, 2014; E Zitzler, Thiele, Laumanns, Fonseca, & Da Fonseca, 2003). However, each QI is designed to take one or more optimization goals. This means that there is no single QI that can measure all approximation goals reliably and the results show the inconsistencies in various approximation sets assessments. The following explains some of the prevalence used QI techniques (Cheng, Zhan, and Shu, 2016; Cremene *et al.*, 2016) :

- **Generational Distance** (GD) measures the distance from the nondominated solution to the Pareto front (Veldhuizen and Veldhuizen, 1999) as shown in Figure 2.11. The $d_i = min_j \; \| f(x_i) - \mathrm{PF}_{\mathrm{true}}(x_j) \|$ represents the distance between the non-dominated solution $f(x_i)$ and the Pareto front ($\mathrm{PF}_{\mathrm{true}}(x_j)$) in the objective space. GD measures the closeness of the solutions to the Pareto front. If the GD value is zero, this means all the nondominated solutions are placed exactly on the Pareto front. Algorithms with low GD values have better performance than algorithms with high GD values. GD emphasizes on convergence when evaluating the quality of the Pareto fronts.

$$\mathrm{GD} = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{2.17}$$

- **Inverted generational distance plus (IGD +)** is weakly Pareto compliant compared to IGD where d_i^{+}= max $\{a_i - z_i, 0\}$ represents the modified distance from z_i to the closest solution in A with the corresponding value a_i. It calculates the distance from each reference point to the dominated region by a solution set. IGD+ incorporates Pareto dominance between a reference point and a solution in their distance calculation.

$$\mathrm{IGD}^{+}(\mathrm{A}) = \frac{1}{\lceil Z \rceil} \left(\sum_{i=1}^{|Z|} d_i^{+2} \right)^{\frac{1}{2}} \tag{2.18}$$

- **Spacing** (S) was introduced by Schott (Schott, 1995). It measures the extent to which the uniformity of the nondominated solution is distributed as shown in Figure 2.11. It is formulated as follows:

$$\mathrm{S} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (d_i - \bar{d})^2} \tag{2.19}$$

The $\bar{d}$ is the average value of d_i. The n is the number of individuals in the non-dominated solution. The algorithm that has a small spacing value dictates better performance than other algorithms with large spacing values.

- **Maximum Pareto Front Error** (MPFE) is introduced by Van Veldhuizen (Veldhuizen and Veldhuizen, 1999) to measure the distance between the Pareto set and non-dominated solutions obtained by a given algorithm as shown in Figure 2.11.

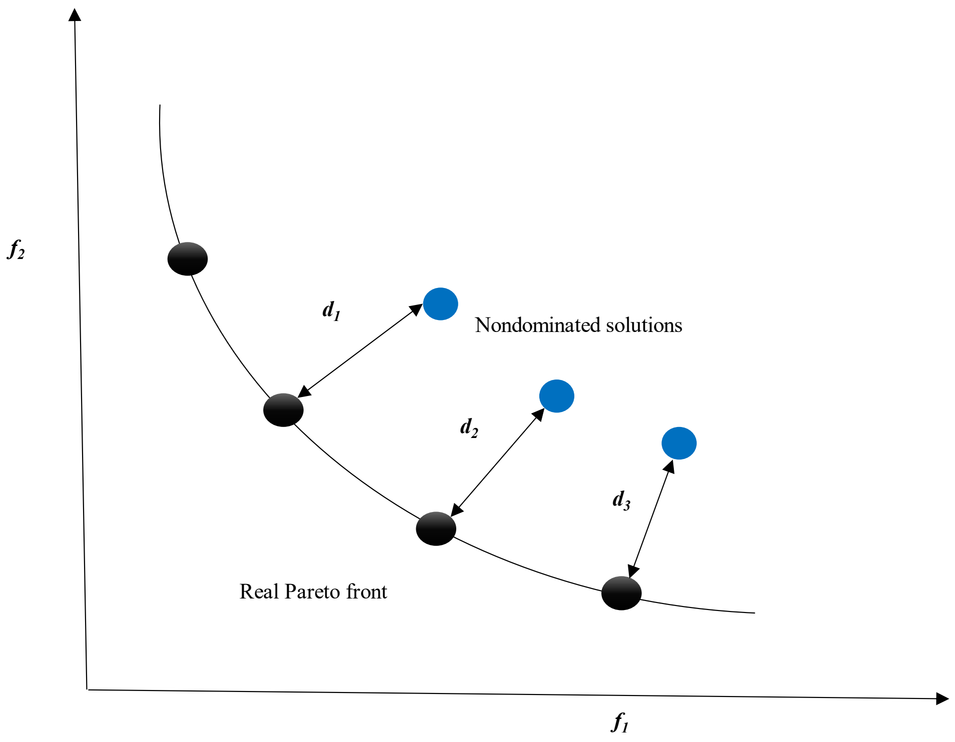


Figure 2.11: GD, Spacing, and MPFE (Yu, Lu, and Yu, 2018)

The algorithm that has a low MPFE value is better performance than algorithms that have high MPFE value.

$$\text{MPFE} = \max d_i \qquad (2.20)$$

The $d_i(min_j \parallel f(x_i) - \text{PF}_{\text{true}}(x_j)\parallel)$ is the distance between the non-dominated solution $f(x_i)$ and the Pareto front ($\text{PF}_{\text{true}}(x_j)$) in the objective space.

- **Inverted generational distance** (IGD) (Hisao Ishibuchi *et al.*, 2015) inverts generational distance as shown in Figure 2.12. An IGD of distance from PF_{true} to P_A is defined as follows:

$$\text{IGD}(\text{P}_\text{A},\text{PF}_{\text{true}}) = \frac{\sum_{v \in \text{PF}_{\text{true}}} d(v,\text{P}_\text{A})}{|\text{PF}_{\text{true}}|} \qquad (2.21)$$

where P_A is the non-dominated solution set output from the algorithm.

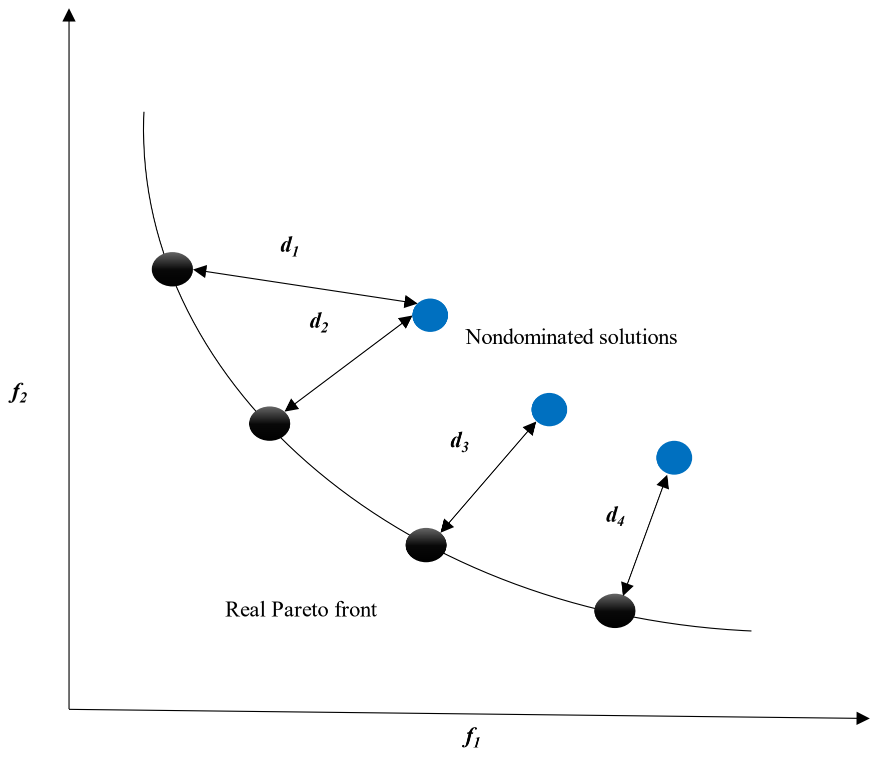


Figure 2.12: IGD (Yu, Lu, and Yu, 2018)

PF_{true} is the Pareto front, $d(v, P_A)$ is the minimum distance calculated using the Euclidean formula between v and the points in P_A. IGD focuses on convergence and diversity when evaluating the quality of Pareto fronts.

- **Hypervolume** (HV) or (*S-metric*) was originally proposed by Zitzler and Thiele (Zitzler and Thiele, 1998) to calculate the area dominated by a set of solutions for a reference point as shown in Figure 2.13.

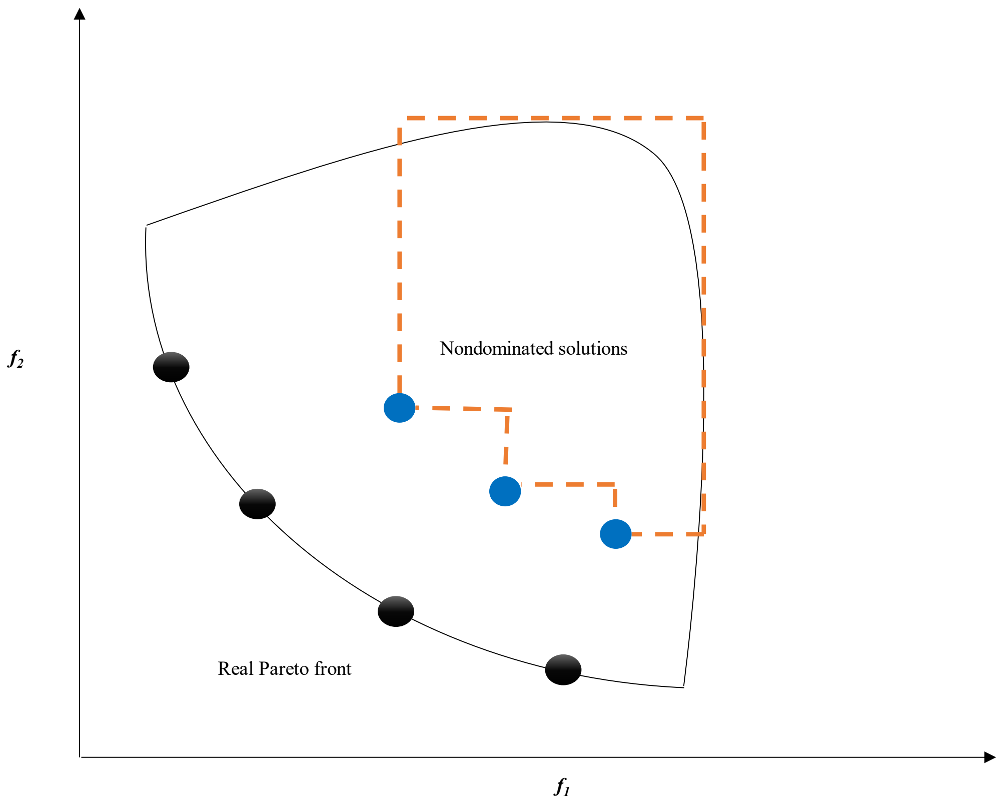


Figure 2.13: Hypervolume (Yu, Lu and Yu, 2018)

- A hypervolume is defined as follows:

$$\text{HV} = \text{volume}\left(\bigcup_{i=1}^{|P_A|} v_i\right) \qquad (2.22)$$

The algorithm proposed that has a larger value in HV than other algorithms dictate better performance. Every single QI has advantages and disadvantages. The evaluation of the quality of approximation sets is a MOP and there is no single QI that reliably assesses all aspects at once. Table 2.4 summarizes the characteristics of the selected indicator.

Table 2.4: Quality indicators and their properties (Ravber, Mernik, and Črepinšek, 2017)

Quality Indicator	**Unary**	**Convergence**	**Uniformity**	**Spread**	**Requires reference set**	**Pareto Compliant**
GD	√	√			√	
HV	√	√	√	√		√
IGD	√	√	√	√	√	
IGD+	√	√	√	√	√	√
Spacing	√		√			
MPFE	√	√			√	

2.5 Literature Review

The following summarizes some of DVRPTW and MODVPTW previous works on authors and their contributions, proposed algorithms, objectives, and datasets used. Table 2.5 shows the previous works of the published journals.

Table 2.5: Previous works

Author (Title –Year)	Contributions	Algorithm	Objectives	Dataset
Feng Wang, Fan shu Liao, Yixuan Li, Xu Chen (An ensemble learning based multi-objective evolutionary algorithm for the dynamic vehicle routing problem with time windows - 2021)	1) A framework of the population is re-initialized to respond to environmental changes effectively. 2) Propose an ensemble learning based dynamic handling method. It can combine different reaction strategies to adapt to different cases, which improves the overall performance. 3) Design a new population-based prediction strategy	Ensemble Learning Dynamic multi-objective optimization evolutionary algorithm	Minimizing the total travel costs and the fixed costs of used vehicles	Hypothetical
Hao Tang and Mingwei Hu (Solution Framework and Computational Experiments - 2005)	1) An efficient chained local search heuristic is embedded in this solution framework. 2) Throughput maximization 3) Improvement only on waiting and travel time objectives	Solution Framework and Computational Experiments	Minimizing the total travel costs and the fixed costs of used vehicles	Hypothetical
S.F. Ghannadpou, S. Noori , R. Tavakkoli-Moghaddam, K. Ghoseiri (solving strategy based on the genetic algorithm (GA) and three basic modules - 2014)	1) A genetic algorithm (GA) solving strategy with its management module, strategy module, and optimization module. 2) A management module for acknowledging information on vehicles and customer every time. 3) The strategy module organized the information reported by the management module and constructed an efficient	Solving strategy based on the genetic algorithm (GA) and 3 basic modules	Minimizing the total travel costs, number of used vehicles, the total waiting time of vehicles, and maximizing total satisfaction rates of customers	Hypothetical

	structure for solving in the subsequent module			
Omprakash Kaiwartya, Sushil Kumarand Ahmed Nazar Hassan (A time seed based solution using particle swarm optimization TS-PSO - 2015)	1) A small granularities DVRP problems 2) A time horizon of each small granularities DVRP is divided into time seeds 3) Each time seed is solved using PSO	A time seed based solution using particle swarm optimization (TS-PSO)	Maximizing the number of vehicles, expected reachability time, profit, and satisfaction level	Hypothetical
Shifeng Chena, Rong Chena, Gai-Ge Wang, Jian Gaoa, Arun Kumar Sangaiah (An adaptive large neighborhood search heuristic for dynamic vehicle routing problems- 2018)	1) Ad hoc destroy/repair heuristics and a periodic perturbation procedure. 2) An efficient feasibility check for customer insertion 3) Problem is broken down into a series of static VRPs to detect new customer requests during in a time slice.	Adaptive Large Neighborhood Search (ALNS)	Minimizing the total travelled distance	Hypothetical
Lianxi Hong. (An improved LNS algorithm for real-time vehicle routing problem with time windows – 2012)	1) Use remove–reinsert process in the LNS, the latest request nodes are regarded as a part of the removed nodes; these nodes can be inserted into current solution during reinsertion process;	An improved LNS algorithm for real-time vehicle routing problems with time windows (LNS)	Minimizing the total travelled distance	Hypothetical
Jesica de Armas, Belén Melián-Batista(Variable Neighborhood Search for a Dynamic Rich Vehicle Routing Problem with time windows- 2015)	1) Use VNS to solve problem. 2) Tackle and change the following constraint 1) Change to fixed heterogeneous fleet of vehicles 2) Manage customers that cannot be served during planning horizon 3) Lengthen drivers work shifts	Variable Neighborhood Search(VNS)	Minimizing the total travel costs and the fixed costs of used vehicles	Hypothetical and unpublished real data

	4) Postponement of remaining customers maximization.			

2.6. Summary

The vehicle routing problem is an important optimization research areas to be studied. This problem can get complex, challenging, and difficult when more constraints are added, and this led to different variants of vehicle routing problems. Some variants are based on static information and the others are based on dynamic or online information. The static information-based variants processes are more straightforward, but the dynamic information-based variants processes are more complex and time-consuming to solve and develop.

There are many algorithms proposed in these studies. The EA and LNS algorithms are among the popular algorithms used to solve vehicle routing problems. These algorithms are even more popular when used in multi-objective optimization problems. In the multi-objective optimization problem, the techniques used for assessing the performance of the algorithms are based on the wideness of the Pareto sets and area coverage based on Pareto sets and reference points. Such techniques as GD, IGD among others are commonly used for performance appraisal.

CHAPTER 3

PROPOSED ALGORITHM

3.1. Introduction

The proposed algorithm consists of two main parts. The first part is the non-fitness evolutionary algorithm (NEA). The second part is the parallelized adaptive large neighbourhood search algorithm (PALNS). The PALNS in the proposed algorithm is designed and deployed as microservices. Microservice is a service-oriented architecture that emphasizes software components that can act independently, which makes them loosely coupled and potentially distributed. Hence, we name our proposed algorithm as a non-fitness evolutionary distributed parallelized adaptive large neighbourhood search (NEDPALNS) algorithm. NEDPALNS is designed and operated differently, as well as independently as a separate business process, service, or entity. Separate entities in NEDPALNS scale, work cohesively and update without interrupting other entities in the wide functionality of the application. In this section, we explain the representation of microservices in the PALNS algorithm, the flow of the NEDPALNS process, and the details of the functions. The NEDPALNS algorithm has three main functions. First, it initializes the solutions in the population. Second, it performs the non-fitness evolutionary step, and lastly, it runs perpetually the distributed PALNS until termination criteria are met.

3.2. NEDPALNS

The following section explains the proposed architecture, framework, system characteristics, and activity sequence interaction among other features characteristics. It also explains how the solution is constructed and represented in NEDPALNS, as well as the lifecycle of NEDPALNS. Each NEDPALNS functionality is explained in the following section.

3.2.1 Architecture

Figure 3.1 shows the NEDPALNS architecture is divided into two parts. The first part is the non-fitness evolutionary algorithm (population initialization, non-fitness selection, solutions intercross, solutions mutation, and interim optimization) and the second part is perpetual optimization which is designed and deployed as a microservice. The microservices can be deployed over different varieties of host types. The host types can be logical or physical. The physical host type is the bare metal that contains plain computing resources such as a server or any commodity hardware. The logical host type can appear as virtualization, containerization, and on the cloud. The purpose of having such an architectural design is to allow the NEDPALNS to scale, ease of integration, and automatic deployment, support multi-tenancy, and ease of latest technologies adoption among other benefits. The non-fitness evolutionary algorithm seamlessly executes the microservice located at different host types.

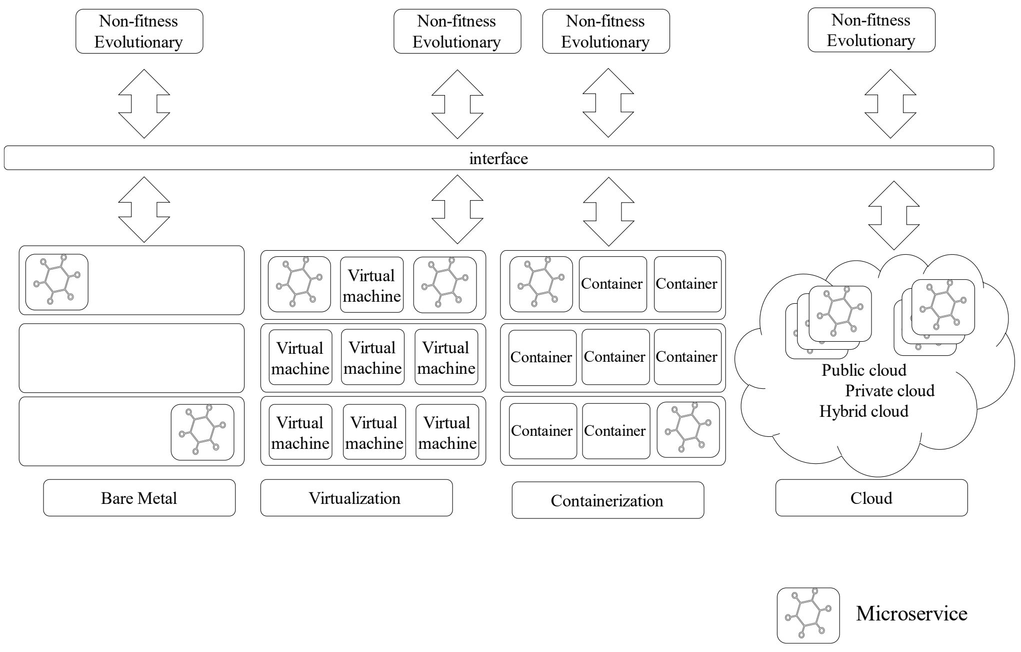


Figure 3.1: NEDPALNS Architecture

3.2.2 Optimization Strategy

The DVRPTW solution must remain optimal and practical if new information is revealed over time. This means that exact algorithms may not seem appropriate for this type of random appearance of new information, which can cause a “curse of dimensionality” and dampen the support of large instances (Powell, 2007). Therefore, a metaheuristic algorithm (Gendreau and Potvin, 1998; Ghiani *et al.*, 2003; Ichoua, Gendreau and Potvin, 2007; Zeimpekis *et al.*, 2007; Jaillet and Wagner, 2008) is a way forward to rapidly generate a new solution given a dynamic problem state. There are many optimization approaches to DVRPTW. In NEDPALNS, we use two optimization approaches. The first one is interim optimization (Psaraftis, 1980b; Kilby, Prosser and Shaw, 1998; Yang, Jaillet and Mahmassani, 2004; Montemanni *et al.*, 2005; Chen and Xu, 2006; Rizzoli *et al.*, 2007) and the second one is the perpetual optimization

(Barceló, Grzybowska, & Pardo, 2007; R. W. Bent & is Van Hentenryck, 2004; Cheung, Choy, Li, Shi, & Tang, 2008; Gendreau, Guertin, Potvin, & Taillard, 1999; Haghani & Jung, 2005; Ichoua, Gendreau, & Potvin, 2000).

3.2.2.1 Interim Optimization

Initially, the route, the customer, and the constraints are initialized, associated, and optimized into the initial solution as shown in Figure 3.2. An event will be triggered once there is a new customer arrival, route changes, or customer state changes at time t_{t+1} which triggers the interim optimization to generate an interim solution that otherwise will be idling waiting for an update event.

The interim optimization is particularly designed for static routing. However, there is a lag when the update is returned to the decision-maker and computational power sitting idling during waiting time. Despite this shortcoming, this optimization approach has already been used in many types of research, especially VRP variants.

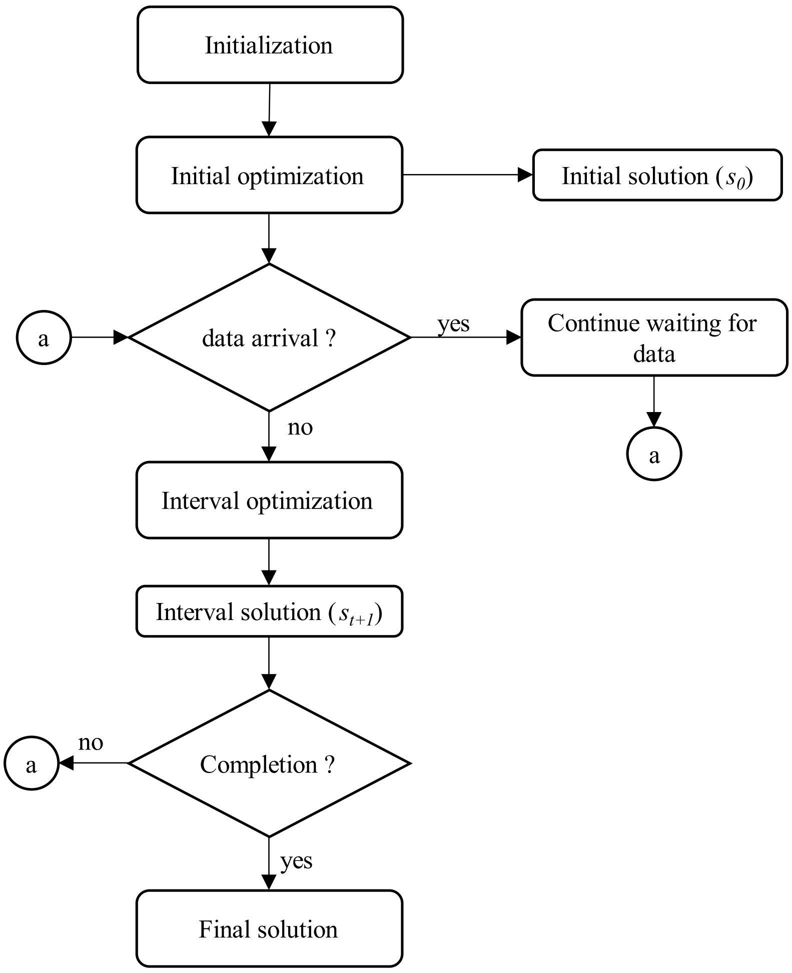


Figure 3.2: Interim optimization

3.2.2.2 Perpetual Optimization

Figure 3.3 shows that perpetual optimization consists of optimization and strategy handlers are part of the DVRPTW core engine, which runs perpetually in the background. The optimization handler performs interim optimization while storing the interim solutions in the adaptive memory. These interim solutions are aggregated from the adaptive memory, and the strategy handler strategically outputs the interim solution from the aggregated interim solutions in the adaptive memory. Perpetual optimization operations run forever while maximizing the utilization of computing power. The parallelized running threads appear at the interim optimization on the left side of this diagram to ensure more near-optimal interim solutions are unfolded. Each thread generates near-optimal interim solutions from all incoming customer arrivals and is cross-checked against all interim solutions (routing strategies or routing plans) in the adaptive memory to decide the viability of the solution before the solution is discarded.

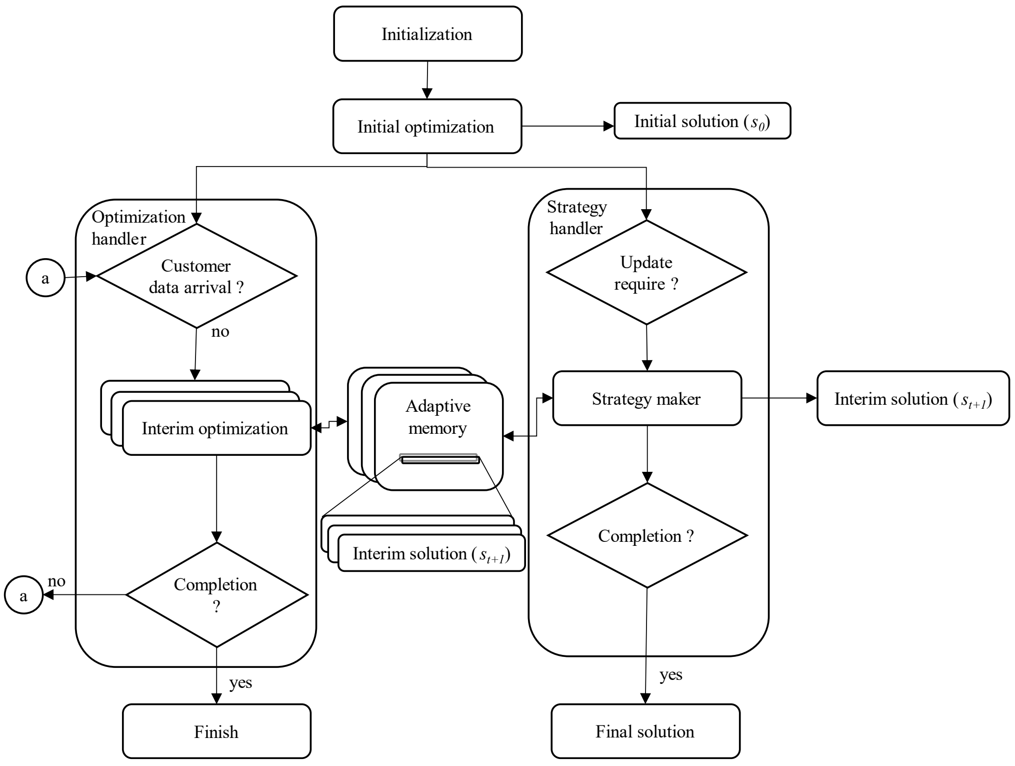


Figure 3.3: Perpetual Optimization

If the solution is unique and has a better result, it will be inserted into adaptive memory. Interim solutions in adaptive memory are updated at intervals (left side of the diagram) whenever a vehicle completes servicing the customer service to ensure that the current state of vehicles and customers is coherent with all solutions in adaptive memory.

3.2.3 System Characteristics

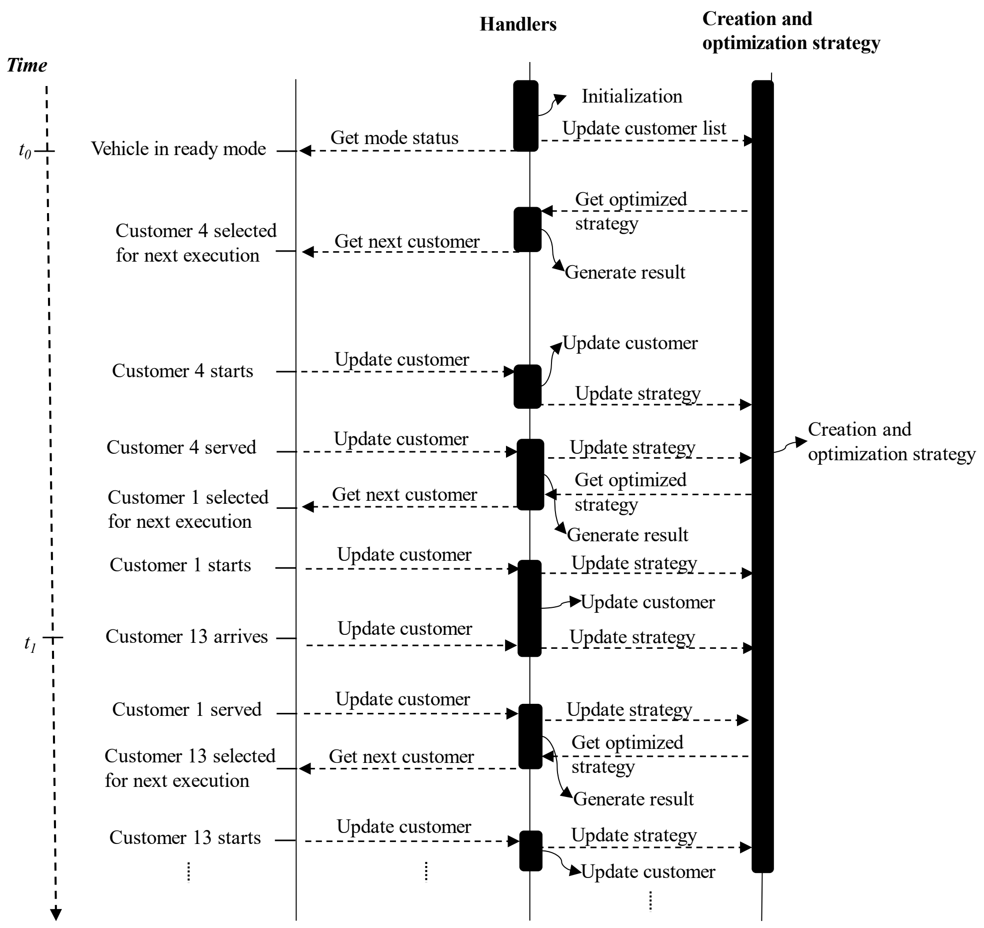
NEDPALNS is developed to support event triggers and parallelism that existed in the DVRPTW environment. This means that in event triggers, NEDPALNS must withstand and handle fast-changing events such as the customer serving event to the customer served event, customer arrival, customer rejection, and strategy selection, among other events.

In parallelism features, NEDPALNS performs multitasking to produce fast decisions periodically and continuously update optimized strategies. It decides the next course of action such as the next customer to be served or rejected. These parallelism features in NDEPALNS are designed and executed in a distributed computing environment.

In this thesis, NEDPALNS are evaluated against two datasets in MODVPRTW. The first dataset that is regularly used to benchmark against other algorithms is the Solomon dataset and dynamic dataset (Chen *et al.*, 2018). The detail of the dynamic dataset is explained in detail in the dynamism dataset under the dataset's subsection in chapter 4. The second dataset that is gaining popularity that uses real-life coordinates and real distance is the MOVRPTW dataset (Castro-Gutierrez, Landa-Silva, and Pérez, 2011) and the dynamic dataset (Chen *et al.*, 2018). The details of both the Solomon dataset and dynamic dataset are explained in a subsequent chapter.

3.2.4 Activity Sequence Interaction

Figures 3.4(a) - 3.4(c) represent the sequence of calls in a DVRPTW system and a set of activities to be performed. These figures represent the flow of messages from one activity to another activity and the state of customers



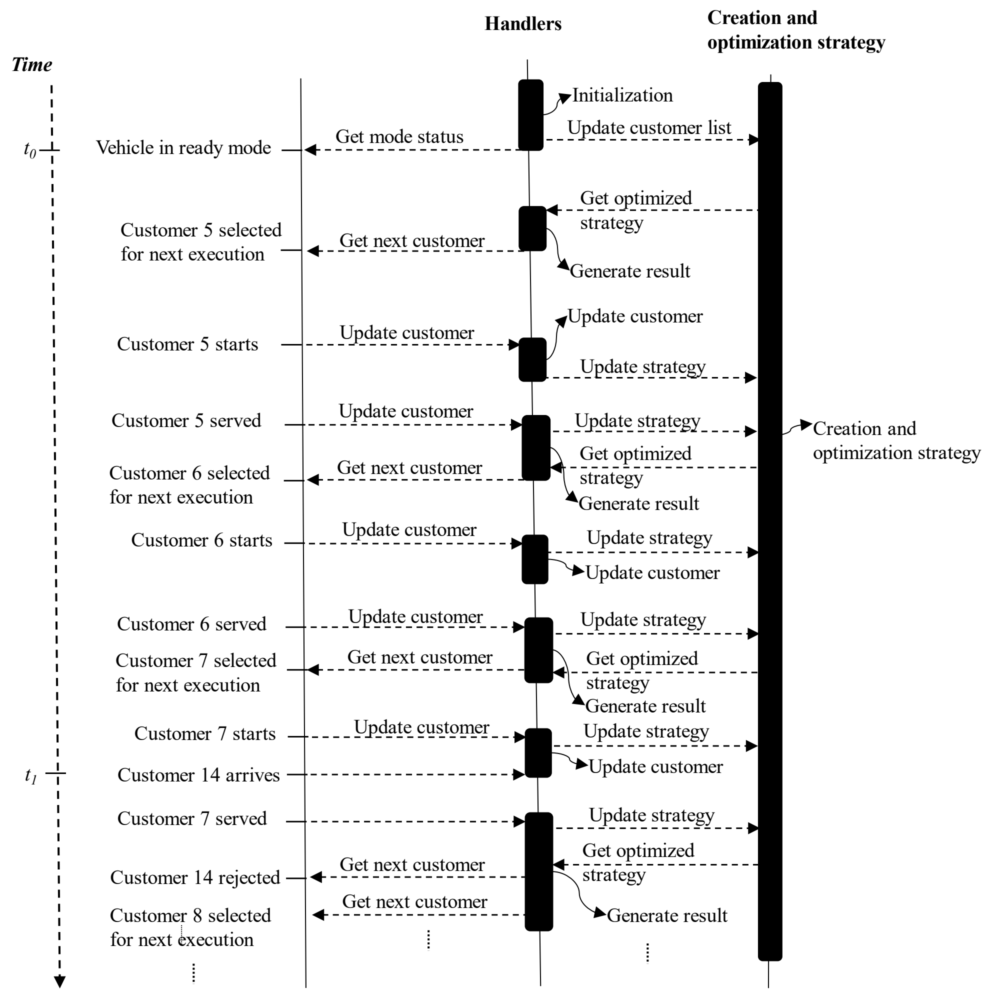
(a)

Figure 3.4: Activity sequence with a timeline of the DVRPTW

during time intervals. First, the vehicle, customers, and constraints are initialized, which is then processed in the creation and optimization strategy, and these vehicles are set in a ready mode at zero time (*t_0*) as shown in Figure 3.4(a).

The first customer (customer 4) is selected from the optimized strategy for execution. This customer starts and ends by updating the creation and optimization strategy. The next customer (customer 1) is selected for execution after obtaining the result of the creation and optimization strategy. At *t_1* time, a new customer request (customer 13) is obtained and updated in the creation and optimization strategy. The executing customer (customer 1) finishes and updates the creation and optimization of the strategy. The earlier new customer (customer 13) is accepted into the strategy creation and optimization after being processed and selected for the next execution. These steps continue until the vehicle returns to the depot.

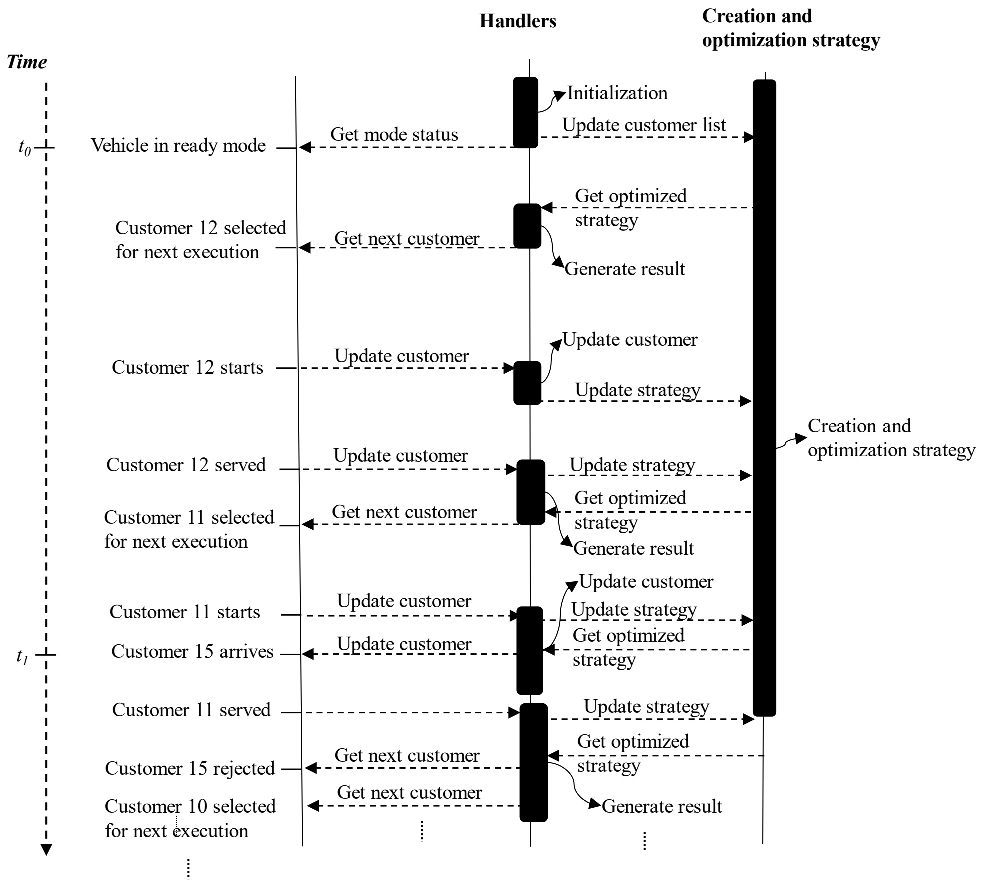
In Figure 3.4(b), at time zero (*t_0*), the vehicle, customers, and constraints are initialized and processed in the creation and optimization strategy before the vehicles are set in a ready mode. The customer accepted earlier (customer 14) in the creation and optimization strategy after being processed is rejected for the next execution. These steps continue with the other customers (customer 8) until the vehicle return to the depot. Similarly, in figure 3.4(c), the vehicle, customers, and constraints are initialized which is then processed in the creation and optimization strategy, and the vehicles are set in a ready mode at time zero (*t_0*). The first customer (customer 5) is selected from the creation and optimized strategy. This customer starts and ends by updating the creation and optimization strategy. The next customer (customer 6) is selected for execution after obtaining the result from the creation and optimization strategy. Once the customer (customer 6) is served, the next customer (customer 7) is selected for execution.



(b)

Figure 3.4: Activity sequence with a timeline of the DVRPTW

After the customer has started to serve, a new customer request (customer 14) is obtained and updated in the creation and optimization strategy at t_1 time. The executing customer (customer 7) finishes and updates the creation and optimization strategy. The first customer (customer 12) is selected from the optimized strategy for execution. This customer starts and ends by updating the creation and optimization strategy. The next customer (customer 11) is selected for execution after obtaining the result of the creation and optimization strategy. Once the customer (customer 11) is served, the next customer (customer 10) is selected for execution.



(c)

Figure 3.4: Activity Sequence with a Timeline of the DVRPTW

After the customer has started to serve, a new customer request (customer 15) is obtained and updated in the creation and optimization strategy at *t_1* time. The executing customer (customer 10) finishes and updates the creation and optimization strategy. The earlier new customer (customer 15) is accepted into the creation and optimization strategy after being processed and rejected for the next execution. This process continues with the other customer until the vehicle returns to the depot.

Dynamism can appear in a variant of forms such as demand for goods (Attanasio *et al.*, 2004; Hvattum, Løkketangen and Laporte, 2006; Goel and Gruhn, 2008), demand for services (Beaudry, Laporte, Melo
& Nickel, 2010; Larsen, Madsen, & Solomon, 2004; Thomas, 2007), travel time (Lorini, Potvin and Zufferey, 2011; Tagmouti, Gendreau and Potvin, 2011; Güner, Murat and Chinnam, 2012), service time, demands a set of known customers (Novoa and Storer, 2009; Secomandi and Margot, 2009) and vehicle availability (Li, Mirchandani, and Borenstein, 2009; Mu *et al.*, 2011).

3.2.5 Framework

NEDPALNS framework is divided into 2 lifecycles which are the evolutionary lifecycle and generation lifecycle as shown in Figure 3.5. The evolutionary lifecycle consists of a set of procedures (population initialization, non-fitness selection, solutions intercross, and solutions morph) that are executed in sequence. The evolutionary lifecycle comprises the generation procedures (population initialization, solutions intercross, and solutions morph) and a re-optimization procedure (PALNS). The generation lifecycle performs two types of optimizations. The first type is non-cyclic optimization in which each generation procedure executes the remote PALNS one time and returns the generated solution. The second type is cyclic optimization, which executes remote PALNS successively until a termination condition is met.

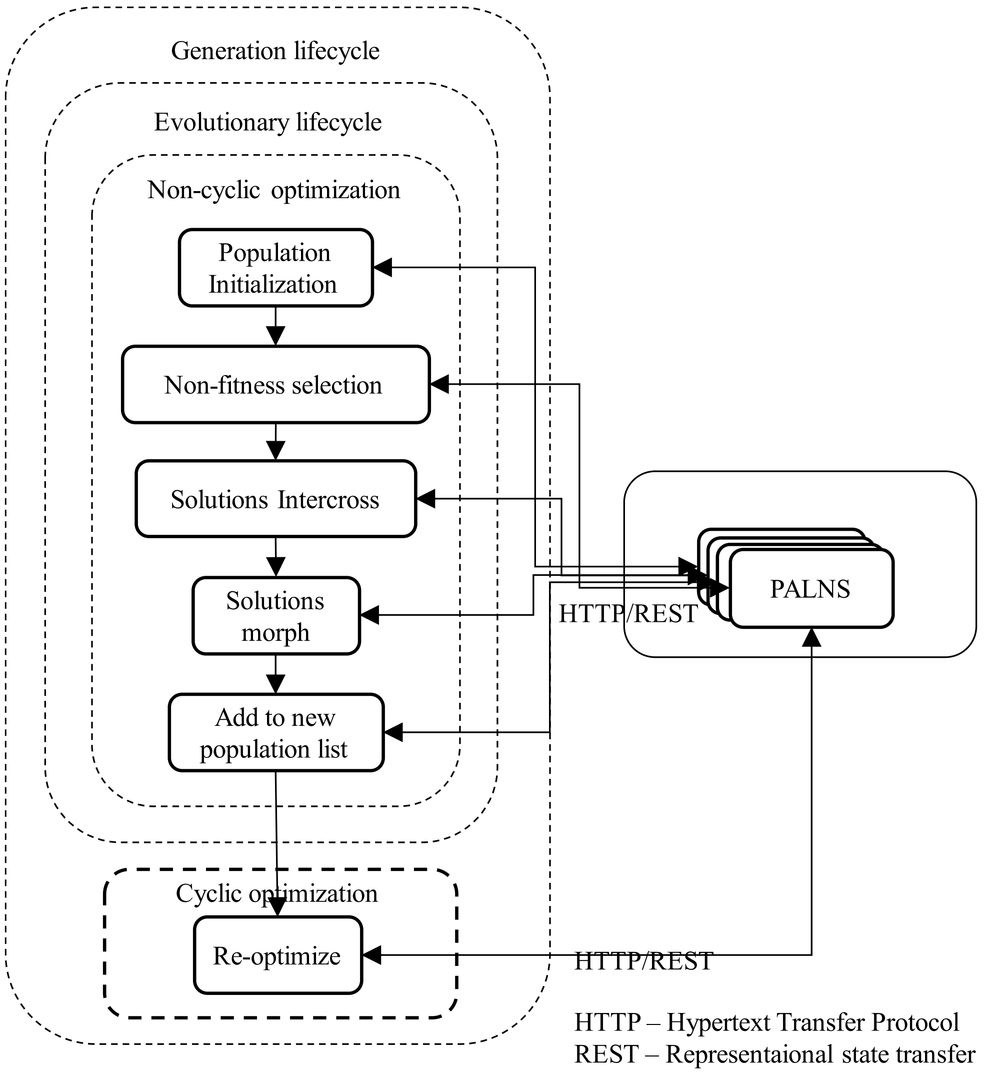


Figure 3.5: NEDPALNS Framework

Figure 3.6 shows that PALNS is designed based on microservice architecture. PALNS has two main responsibility which is to remove the solution and to repair the solution. Four removal procedures and two repair procedures are proposed in this thesis. During the execution of PALNS, each removal procedure and repair procedure is selected randomly to remove existing customers from the solutions and repair the deleted customers into a new solution.

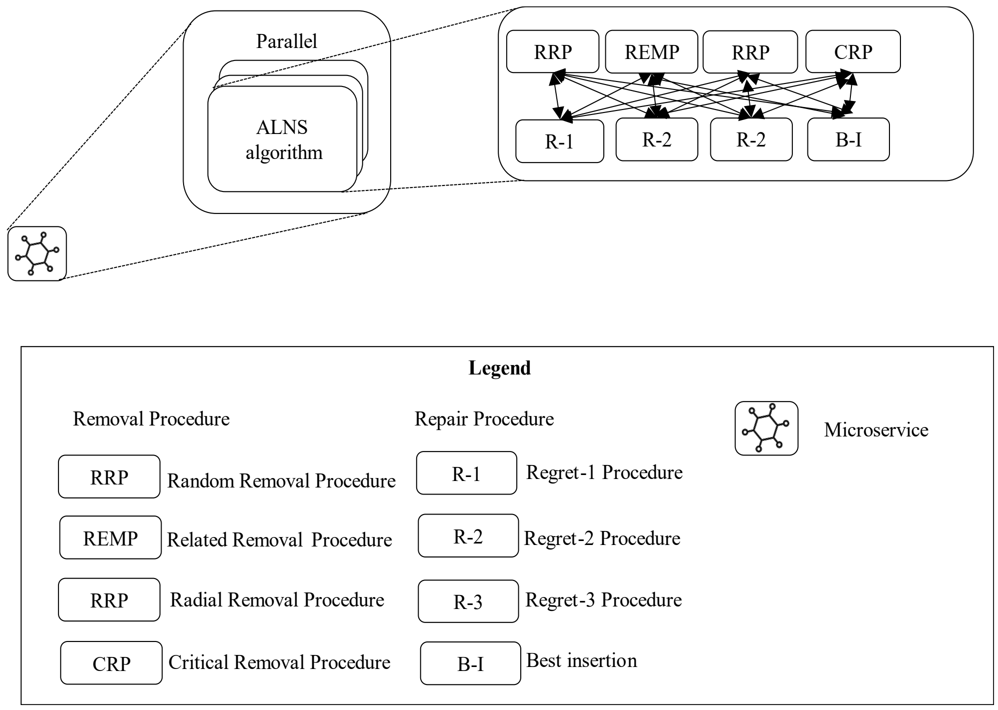


Figure 3.6: Microservice Representation

Initially, the NEDPALNS process starts with the generation of interim solutions during population initialization. Interim solutions are randomly chosen. In NEDPALNS, the non-fitness interim solution is randomly selected to prevent it from falling into local optimal. Two randomly selected interim solutions from the previous procedure are intercrossed to generate a new solution. This new solution morphs into a new interim solution which is added to another population list. These steps continue until the population list is filled up.

The interim solutions in the population list are sorted, compared against the best solution, and replaced if it has a better result. The best interim solution is selected for further refinement in the re-optimization procedure which runs continuously until the termination criteria are met.

The generation lifecycle consists of non-cyclic optimization and cyclic optimization. This division of generation lifecycles into non-cyclic optimization and cyclic optimization escapes local optimal and promotes global optimal. In the re-optimization cycle, only the best solution result from the evolutionary cycle is completely explored and intensely improved. In this way, the optimized solution is produced. In the generation cycle, evolutionary procedures execute the distributed adaptive local neighbourhood search algorithm (DALNS) using synchronous communication. This synchronous communication is performed either using hypertext transfer protocol (HTTP) or representational state transfer (REST) services. HTTP is an application layer protocol for transmitting data. Typically, it enables web browsers to interact with web servers apart from from application communication. REST is an architecture style for developing web services. It is used for exchanging data in a defined format for interoperability purposes.

3.2.6 Solution Representation

Figure 3.7 shows how customers in DVRPTW are represented and encoded as an individual or chromosome that contains the number of routes with the associated customers and the total travelled distance. The terms chromosome or individual are used interchangeably for representing solution and they often refer to the same thing.

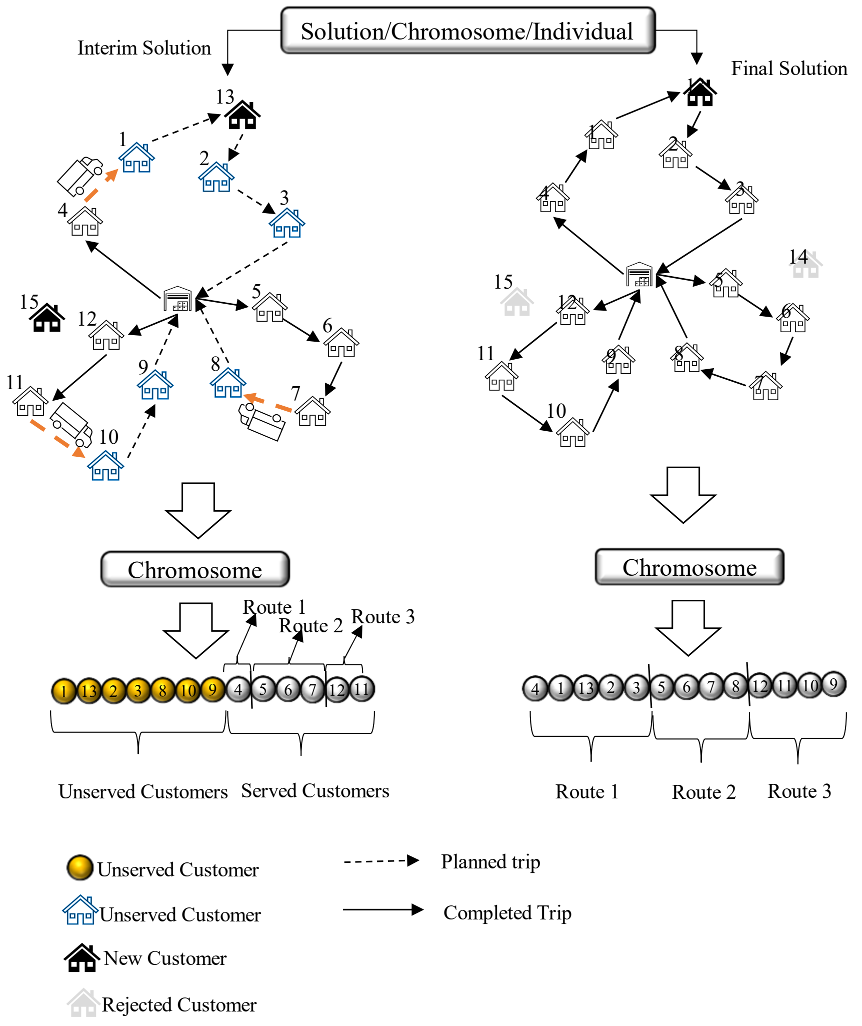


Figure 3.7: Solution Representation

The solution in DVRPTW can be an interim solution or a final solution. In the interim solution, it served customers and unserved customers. Each customer is tagged with an identification that represents a digit. Customer (gene) order on a chromosome is crucial for the formation of the routes, distance calculation, and rejection rates reduction, among other characteristics.

3.2.7 Generation Lifecycle

Algorithm 1 shows the outline of the generation lifecycle of the NEDPALNS algorithm. It consists of two types of optimizations. The first type of optimization is non-cyclic optimization is performed when there is a new customer arrives (lines 3-6).

Algorithm 1 Main

```
Input S: solution
Output BS : best solution
1. begin
2.    //Non-cyclic optimization
3.    P ← population_Initialization(S)
4.    BS ← updateBestSolution(P) // Update the best
      solution
5.    P ← evolutionaryStep(P)
6.    updateBestSolution(P)
7.    //Cyclic optimization
8.    while terminating criteria not met
9.        P` ← palns(P)
10.   end while
11.   BS ← updateBestSolution(P)
12. end
13. return BS
```

Once the non-cyclic optimization is completed, the interim solution is passed to the cyclic optimization for further execution until the termination criteria are met (lines 8 – 10). Finally, the near-optimal interim solution or strategy is returned.

3.2.8 Population Initialization

During population initialization (line 3) as shown in Algorithm 2, the unserved customers of the initial solution are randomly shuffled and sent to remote PALNS for execution. A new interim solution is generated and stored in the population (line 4). These steps continue until it reaches the maximum number of interim solutions allowable in the population size.

```
Algorithm 2 Initialization
Input IS : Initial Solution, PS: Population size
Output P : Population
1. begin
2.   for j ← 1 to PS do //PS - population size
3.     /* $P_j$ - $j^{th}$ individual (interim solution) in the population */
4.     $P_j$ ← palns(shuffle(IS.getUnservedCustomers))
5.   end for
6. end
7. return P
```

3.2.9 Non-fitness Evolutionary Algorithm

The non-fitness evolutionary algorithm run within the evolutionary lifecycle consists of 3 main steps. The first step is to randomly select two parents from the population. The second step is to intercross the parents to generate offspring. Lastly, this offspring is morphed into new interim solutions. The following section explains the evolutionary lifecycle and its operators in detail.

3.2.9.1 Evolutionary Lifecycle

The evolutionary lifecycle has three main steps as shown in Algorithm 3 which are a randomly non-fitness selection, solutions intercross, and solutions morphing. First, in the selection process, two interim solutions are randomly selected from the population (lines 4-7). These two interim solutions (parents) with the unserved customers are intercrossed to generate offspring. These offspring are intercrossed and added to the new population. These steps are repeated until the new population size (lines 3 – 12) is met.

Algorithm 3 Generation Step

```
Input S: Initial Interim solution, P: Population, N: Population size
Output P : Interim solutions
1: begin
2: P` ← P, i =0;
3: while (P'.size() < 2 * P.size()) do
4:    s` ← random[0..1] // A non-fitness value is randomly selected
5:    loc1 ← s' * P.size()
6:    s` ← random[0..1] // A non-fitness value is randomly selected
7:    loc2 ← s' * P.size
8:    $P`_i$ ← solutionsIntercross($P_{loc1}$,$P_{loc2}$)
9:    i ← i +1
10:   $P`_{i+1}$ ← solutionsIntercross ($P_{loc2}$, $P_{loc1}$)
11:   i ← i +1
12: end while
13: P` ← solutionsMorph(P`) //morphing function
14: P` ← sortPopulationAscending(P`) // sorted based on the least
    total travelled distance
15: for i ← 1 to N do
16:    $P_i$ ← $P`_i$
17: end for
18:
19: updateBestSolution(P) //Replace best solution from P
20: end
21: return P
```

The solutions in the new population are morphed (line 13) into new solutions and sorted based on the least total travelled distance. Next, the interim solution in the new population replaces the original population (lines 15 - 17). This indicates that the original population only contains the best interim solutions. Lastly, the best interim solution is updated if the interim solution in the population has a better result (line 19).

3.2.9.2 Solutions Intercross

Algorithm 4 uses the order crossover (OX) algorithm suggested by Goldberg (Goldberg and Holland, 1988). Four parameters are passed to this procedure. They are the first parent, second parent, first cutoff point second cutoff point. First, the customers are copied to the temporary area (slots of the array) depending on the second cut-off point of the first parent. If the second cut-off point of the first parent is similar to the last slot of the first parent (lines 4– 7), the entire first parent fills the array slots, else customers appear in the second cut-off point until the last slot of the first parent fills the array slots (lines 9 – 13). Next, the customers appear in the first slot of the first parent, and the second cut-off point fills the remaining slots in the array (lines 14 – 17).

To initiate the intercross of the first parent and the second parent, customers appear at the first cut-off point, and the second cut-off point is copied from the second parent (lines 19 – 25). These copied customers fill the similar cutoff points in the second parent to the first offspring (lines 26 – 28). Figure 3.8 illustrates the solutions intercross.

Algorithm 4 Solutions_intercross

```
Input Parent1: first parent, Parent2: second parent,
Cutoff_point1: first cutoff point, Cutoff_point2: second cutoff
point
Output offspring : offspring
1:  begin
2:  index ← cutoff_point2+ 1
3:  index2 ← 0
4:  if (index == parent1.unserved_customers.length()) then
5:     for i ← 0 to parent1.unserved_customers.length() do
6:        $arraySlots_i$ = $parent1.unserved\_customers_i$
7:     end for
8:  else
9:     for index ← cutoff_point2 + 1 to
10:        parent1.unserved_customers.length do
11:       $arraySlots_{index2}$ ← $parent1.unserved\_customers_{index}$
12:       index2 ← index2 + 1
13:    end for
14:    for index ← 0 to cutoff_point2 do
15:       $arraySlots_{index2}$ ← $parent1.unserved\_customers_{index}$
16:       index2 ← index2 + 1
17:    end for
18: end if
19: for index ← cutoff_point1 to cutoff_point2 do
20:    for index ← 0 to arraySlots.length() do
21:       if ($arraySlots_{index}$ == $parent2.unserved\_customers_{index}$) then
22:          remove($arraySlots_{index}$)
23:       end if
24:    end for
25: end for
26: for index ← cutoff_point1 to cutoff_point2 do
27:    $offspring_{index}$ ← $parent2.unserved\_customers_{index}$
28: end for
29: index2 ← 0
30: for y ← cutoff_point2 + 1 to offspring.length() do
31:    if (y == offspring.length())
32:       break
33:    end if
34:    $offspring_y$ ← $arraySlots_{index2}$
35:    index2 ← index2 + 1
36: end for
37: for z ←0 to cutoff_point1 do
38:    if (z == offspring.length())
39:       exit for loop
40:    end if
41:    $offspring_z$ ← $arraySlots_{index2}$
42:    index2 ← index2 + 1
43: end for
44: offspring ← palns(offspring)
45: Update_best_solution(P)
46: end
47: return offspring
```

The customers who appear at the first cutoff point and second cutoff point in the second parent are removed from the array slots (lines 20 - 25). The customers in the array slots fill the remaining empty slots in the first offspring (lines 37 – 42). All these steps are illustrated in Figure 3.8. The offspring are passed to remote PALNS for execution to generate a new interim solution (line 44).

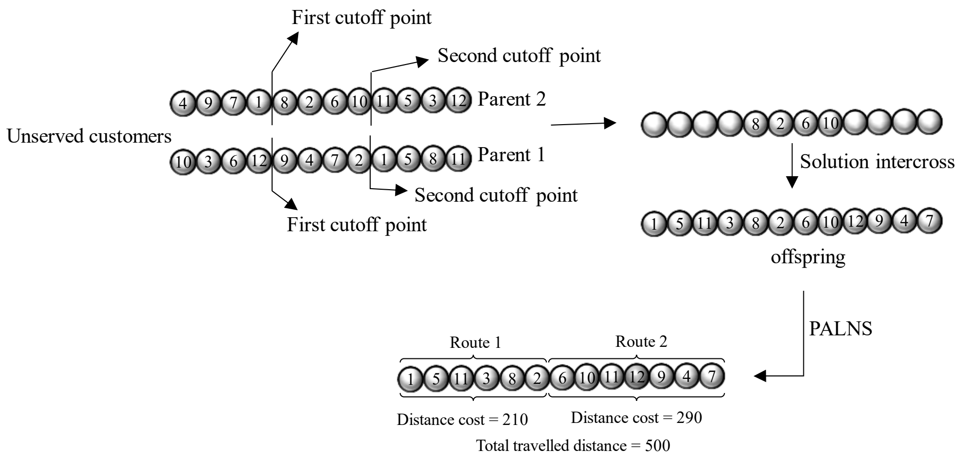


Figure 3.8: Solutions Intercross

3.2.9.3 Solution Morphing

Algorithm 5 adopts a swap mutation (Mihajlović, Živković, and Štrbac, 2007) as illustrated in Figure 3.9. Each solution in the population is visited and checked whether it is a criterion for morphing. The solution of the population is selected for morphing as long as the randomly generated value is lower than

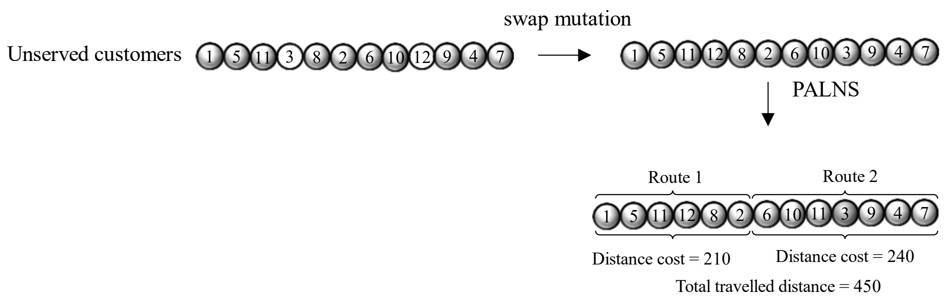


Figure 3.9: Solution Mutation

the morph ratio (line 3). If the ratio of the morph is set higher, the solution in the population has a high chance of being selected for morph, and extensive areas are explored in the search space, but the population may suffer from converging to optimum solutions. Two randomly generated positions in the unserved customer list are generated (lines 5 – 6).

If the generated positions have the same location, it will generate two new positions until they are not equal (lines 7 - 10). The two unserved customers are swapped with one another (line 11). Each solution in the population is visited (line 2). If the morph ratio is higher than the random value (line 3), the two randomly selected genes are swapped in that individual. However, if the random value generated is higher than the morph ratio, there will be fewer selected solutions in the population to be morphed and may trap in local optimal. This means that the morph ratio value must be appropriately selected to ensure that it will not fall into local optimal.

```
Algorithm 5 MorphPopulation
Input P : Population, MR : Morph Ratio
Output P` : Population
 1: begin
 2: for i ← 0 to P.length do
 3:   if (random[0..1].1 < MR) then
 4:      Customer_list ← P_i.get_unserved_customers
 5:      Index1 ← random [0.. Pop.length].1
 6:      Index2 ← random [0.. Pop.length].1
 7:      while (Index1 == Index2) do
 8:        Index1 ← random [0.. Pop.length].1
 9:        Index2 ← random [0.. Pop.length].1
10:      end while
11:   P_i` ← Swap_customer_position(Customer_list, index1,
12: index2)
13:   P_i`← palns(P_i`)
14:  end if
15: end for
16: end
17: return P`
```

3.2.10 Distributed and Parallelized Adaptive Large Neighbourhood Search

The adaptive large neighbourhood search (ALNS) is originated by Pisinger and Ropke (Pisinger and Ropke, 2007). It is an enhanced large neighbourhood search (LNS) (Shaw, 1998). LNS performs two main tasks. The first task is to remove customers from the existing current solution, and the second task is to repair the solutions by reinserting customers back into the solution. There are a variety of repair operators and removal operators that LNS possess. The combination of repair operators and removal operators allows LNS to achieve a better result. The difference between ALNS and LNS is the adaptive layer that is added on top of LNS. The adaptive layer allows the freedom to randomly select removal and repair operators but must be based on past performance.

Algorithm 6 presents the overview of ALNS. Initially, the initial solution *S* is used for processing. It iterates *N* times by executing the removal and repair operators (lines 5 – 16).

Algorithm 6 Adaptive Large Neighborhood Search (ALNS) algorithm

```
Input S: Solution, E: evaluation function, $I^+$: insert operator, $I^-$:removal operator, N: number of iterations
Output $S^*$: solution (best solution)
 1: begin
 2: $S^* \leftarrow S$ //
 3: $S^\pi \leftarrow S$
 4: for $i \leftarrow 1$ to N do
 5:     $i \leftarrow$ getRemovalOperator($I^+$) //select removal operator
 6:     $r \leftarrow$ getInsertionOperator($I^-$) //select insert operator
 7:     $S^\# \leftarrow r(d(S^\pi))$
 8:     if $S^\#$ is an acceptable solution
 9:         $S^\pi \leftarrow S^\#$
10:     end if
11:     if $z(S^\#) < z(S^*)$
12:         $S^* \leftarrow S^\#$
13:     end if
14:     removalAndInsertOperatorUpdate($d$, $r$, $S^`$) //update d and r operator score
15: end for
16: end
17: return $S^*$
```

These removal and repair operators are selected heuristically using a roulette wheel algorithm that reflects their previous performance (lines 5 -6). The new interim solution can only be selected as a current solution by the simulation annealing criterion (line 9).

3.2.10.1 Parallelized procedure

We adopted the parallel version of ALNS which is proposed by Victor Pillac (Pillac, Gueret, and Medaglia, 2013) for fast optimization that efficiently spread out the computational efforts among the processors. Algorithm 7 outlines the PALNS. In this algorithm, a pool *S* of promising solutions (*M*) is optimized in *T* subprocesses. In each "master" iteration, a subset of *T* promising solutions is randomly selected and distributed among the subprocesses. Each subprocess executes I^P ALNS iterations by removing and repairing the current solution S^p (lines 4 – 16). Each subprocess in the final solution is added to the pool (line 15) and filtrated to ensure the pool never exceeds the N solutions boundary (line 17). PALNS stops after performing T^M master iterations. This is equivalent to $I^P * T^M$ ALNS iterations with no synchronization needed between subprocesses thus avoiding deadlocks. The following section explains the removal and repair operators in detail.

Algorithm 7 Parallel Adaptive Large Neighborhood Search (PALNS) algorithm

```
Input S: Solutions, E: evaluation function, $I^+$: insert operator, $I^-$:removal operator, M: maximum pool size (solutions), T: number of subprocesses, $T^M$: number of master iterations, $I^P$: number of parallel iterations
Output $S^*$: solution (best solution)
 1: begin
 2: for i ← 1 to $T^M$ do
 3:     S` ← getSolutionSubset(S, T) // Get $P^S$ solutions subset
 4:     parallel forall $S^\pi$ in S`do
 5:         $S^p$ ← $S^\pi$ //subprocess current solution
 6:         for $I^P$ iterations do
 7:             i ← getRemovalOperator($I^+$) //select removal operator
 8:             r ← getInsertionOperator($I^-$) //select insert operator
 9:             $S^\#$ ← r(d($S^p$))
10:             if $S^\#$ is an acceptable solution
11:                 $S^p$ ← $S^\#$
12:             end if
```

```
13:         removalAndInsertOperatorUpdate($d$, $r$, $S^{\#}$) //update d and r operator score
14:     end for
15:     S ← S ∪ { $S^p$ }
16:   end forall
17: S ← retain(S, M)
18: end for
19: end
20: return $S^* = argmin_{S^\pi \in S} \{E(S^\pi)\}$
```

3.2.10.2 Distributed Procedure

NEDPALNS is executed in a distributed computing environment. The PALNS in NEDPALNS is organized as a microservice that is represented as a container for the PALNS algorithm. The PALNS consists of multiple ALNS running concurrently and collaboratively to produce a near-optimal solution. Each ALNS consists of a removal and repair procedure. In ALNS, random removal, related removal, and critical removal are to be used in the removal procedure whereas, in the repair procedure, best insertion, regret-1, regret-2, and regret-3 are suggested. Each removal and repair procedure is randomly selected during execution to derive a new solution.

3.2.10.3 Adaptive Procedure

This procedure uses a selection roulette to select the removal and repair operators in each iteration. The operator $\theta \in R$ where R^- is the removal operator and R^+ is the repair operator and selection is based on probability p_0. The probabilities are initialized with the value $\frac{1}{|R|}$ and updated at each iteration. Thus, the formula is stated as follows:

$$p_0 = (1-p)\,p_0 + p\frac{s_\theta}{\sum_{\theta \in R} s_\theta} \qquad (3.1)$$

where p [0,1] is the reaction factor that shows how quickly the probabilities are adjusted, and s_θ is operator score in the last l iterations. The scores s_θ are initialized in each iteration. The update on the new solution depends on the last iteration so that a score of σ_1 is granted for a best new solution, σ_2 for an improving solution, σ_3 for a non-improving but an accepted solution, and σ_4 for a rejected solution.

3.2.10.4 Removal Procedure

We suggested that four removal procedures be used in this NEDPALNS algorithm. They are random, related, radial, and critical procedures that were originally proposed by Pisinger and Ropke (Pisinger and Ropke, 2007).

3.2.10.4.1 Random Operator

This operator uses a random selection of the current solution where *f* is the random fraction ([0…1]). It selects customers at random and removes them from the current solution and it has the search diversifying effect.

3.2.10.4.2 Related Operator

This operator removes a set of customers that are related to certain characteristics. In DVRPTW, we measure how related are the two customers. The relatedness r_{ij} of customer i and customer j measure the intensity of the relationship between two customers. It randomly removes a seed customer $i(U = \{i\})$ and enumerative choose a customer i and deletes the most related customer j^* where U is the unserved customer. If the relatedness value (r_{ij}) is lower, the more related the customer i and customer j. Therefore, $j^* = \text{argmin}_{j \in R}\{r_{ij}\}$. In this operator, we proposed a-priori relatedness as originally proposed by Victor Pillac (Pillac, Gueret, and Medaglia, 2013) which is based on distance and time windows between customer i and customer j. Hence, the formula is listed as follows:

$$r_{ij}^{s} = \left(1 + \frac{c_{ij}}{M_c}\right)^{\theta_d} \left(1 + \frac{|b_i - b_j|}{M_t}\right)^{\theta_t} \tag{3.2}$$

Where c_{ij} is the distance between customer i and customer j, b_i and b_j are the ends of the time windows of customer i and customer j, M_c, and M_t are scaling constants, θ_d and θ_t are respective weight on the geographic distance between the two customers, and due dates differences.

3.2.10.4.3 Critical Operator

The critical procedure removes customer *i* that has the least cost. Therefore, the formula for this operator is stated as follows:

$$i^* = \arg max_{i \in R} \{c_{i-1,\, i+1} - c_{i-1,i} - {}_{,i+1}\} \qquad (3.3)$$

where *i - 1* and *i + 1* are the predecessor and successor of *i*

3.2.10.4.4 Radial Operator

This operator selects a random customer from the customer *N*. Select a random customer *c* with $c \leq [f \cdot N\,]$, where *f* is the fraction rate between 0 and 1. Remove customer c from the route. Retrieve the neighbourhood list based on customer *c*. The neighbourhood list is calculated using Euclidean distance. Remove the customers in the neighbourhood list from the routes.

3.2.10.5 Repair Procedure

We suggest using four operators in the repair procedure. They are the regret operators which consist of regret-1, regret-2, regret-3, and the best insertion operator. The details of each are explained in the following section.

3.2.10.5.1 Regret Operators

The repair procedure is based on regret-i heuristics. The three regret levels are used in these repair operators, namely regret-1, regret-2, and regret-3 heuristics (Potvin and Rousseau, 1993). Each unserved customer is iterated and inserted when the insertion value has the least regret value. The regret-I is the desired measurement of how to insert customer *i* in the current iteration when there is no viable best insertion.

The regret-i heuristic is defined as follows:

$$r_i^q = \sum_{h=2}^{q}(\Delta z_i^h - \Delta z_i^1) \tag{3.4}$$

Where Δz_i^h i is the cost of the *q*th best insertion of customer $i \in U$. Note that selecting the customer can be accomplished with the Δz_i^1 value, and therefore *regret-1* is equivalent to the original best insertion heuristic.

3.2.10.5.2 Best Operator

The best insertion strategy (Schrimpf *et al.*, 2000a) uses a randomly generated customer from the removed customer list to perform the best insertion. Each customer is assessed on each vehicle, and minimum cost insertion determines the route. However, a new vehicle and a new route will be allocated if the customer cannot be added to the route.

3.3. Summary

The DVRPTW is different from the VRPTW in that in the latter information is known beforehand, while in the former information is known over time. This makes DVRPTW interesting to be solved because of its close resemblance to the dial-a-ride problem, and food delivery among others. Also, they are ubiquitous in pickup and delivery orders. DVRPTW cannot be solved like the conventional VRPTW because it has to optimize repeatedly to improve the result. There are many optimization strategies available. Among the common ones is interim optimization which performs when there is new arrival of customer request, while perpetual optimization is the continuous optimization giving the state of the existing customers' information. Both work hand in hand and simultaneously to improve the results. The proposed algorithm is based on two core algorithms. The first algorithm is the evolutionary algorithm but modified to target non-fitness solutions. The second algorithm is the distributed and parallel-run adaptive local neighbourhood search algorithm. This algorithm has two lifecycles, the generation lifecycle, and the evolutionary lifecycle. The evolutionary lifecycle performs a non-cyclic optimization while the generation lifecycle performs the non-cyclic and cyclic optimization. The non-fitness evolutionary algorithm targets the less performing solutions that have a higher chance to be selected for intercross and morph. This process continues until the termination criteria are met. The result derived from each genetic operator is executed against the distributed parallelized adaptive local neighbourhood search algorithm. The design of this entire process is to escape local optima while promoting global optima. In the PALNS, it performs the removal and repair process. In the removal process, few removal operators are suggested such as critical, random, radial, and related while in the repair process, the repair operators are the best insertion and regret operators.

CHAPTER 4

RESULTS

4.1. Introduction

This chapter presents the test types and dataset types that are used in the MODVRPTW experiments. It also lists parameters and their values set in NEDPALNS. The results from these experiments are compared against the published algorithms, and best known solutions among others using different metrics. These results will give an insight into the performance of NEDPALNS and its effectiveness.

4.2. Types of Testing

There are two types of tests used to carry out these experiments. The first type of test is a static test. This static test is to test information that is available during planning in MODVRPTW and uses a hypothetical type of dataset. The reason why the static test is performed on MODVRPTW is to showcase the competency of NEDPALNS when the degree of dynamism is zero. The hypothetical type of dataset used is the Solomon dataset (Solomon, 1987).

The second type of test is the dynamic test, which uses two types of datasets. The first type of dataset is hypothetical, and the second type of dataset is a real type. In the hypothetical type, the Lackner dataset (Lackner, 2004) is used. The Lackner dataset combines the Solomon dataset (Solomon, 1987) and the dynamic dataset. This dynamic test is used to test both the information available during the planning and during execution. In real type, the MOVRPTW (Castro-Gutierrez, Landa-Silva, and Pérez, 2011) dataset combined with the dynamic dataset is used for the dynamic test. The details of each dataset are explained in the following section.

4.3. Datasets

Table 4.1 summarizes the datasets used in the hypothetical dataset and the real dataset. The static test uses the hypothetical type of dataset for testing while the dynamic testing uses a hypothetical type of dataset and the real type of dataset for testing.

In the static test, the testing is performed on the static information of the MODVRPTW. This test uses the hypothetical type of dataset which is the Solomon dataset. The reason why the Solomon dataset is used is the availability of the existing published algorithms which also used the Solomon dataset which is conveniently used to gauge NEDPALNS performance.

The dynamic test is performed on the dynamic information of the MODVRPTW. This dynamic information combines two parts of information.

The first part of information consists of the static information which is available during the planning and the second part of information consists of dynamic information which is available during execution. The dynamic test uses two types of datasets. The first type of dataset is the hypothetical type of dataset which uses both the Solomon dataset and dynamic dataset. This is different from static testing, which only uses the Solomon dataset in the hypothetical type of dataset. The second type of dataset is the real dataset.

Table 4.1 Dataset Types

Type of Dataset	Type of Information	Dataset		
		Solomon	MODVRPTW	Dynamic
Hypothetical	Static	√		
	Dynamic	√		√
Real	Dynamic		√	√

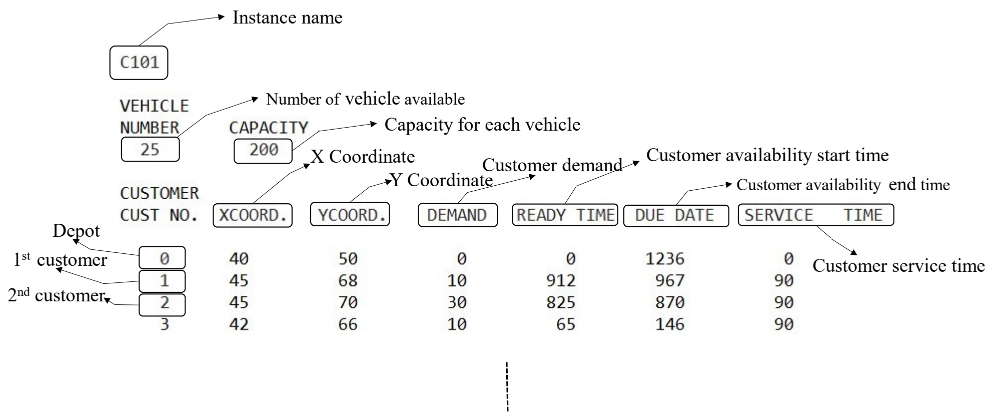
The following section explains each dataset file structure in detail.

4.3.1 Solomon Dataset

These types of datasets are originated from Solomon in 1987. Every instance contains 100 nodes, which are distributed on the Euclidean plane. These datasets are divided into six instance types, namely R1, R2, C1, C2, RC1, and RC2 instance types. There are 56 instances in each category. Each category has a different type of customer distribution, service time, and time windows.

4.3.1.1 File Structure

All dataset files in the Solomon dataset have a similar structure. Figure 4.1 shows the internal structure of one of Solomon's data files, which is the C101 instance. The C101 instance only has 25 vehicles available that are ready to serve customers, and each unit of the vehicle has a loading capacity of 200 units. The identification of the depot is always zero. That is equivalent to zero customer identification. The depot is located at coordinate x, which is 40, and coordinate y, which is 50. It has no unit of demand, no service time, and is available between 0 and 1236.



```
C101

VEHICLE
NUMBER     CAPACITY
  25         200

CUSTOMER
CUST NO.  XCOORD.   YCOORD.    DEMAND   READY TIME  DUE DATE   SERVICE   TIME

    0      40        50          0          0        1236          0
    1      45        68         10        912         967         90
    2      45        70         30        825         870         90
    3      42        66         10         65         146         90
```

Instance name
Number of vehicle available
Capacity for each vehicle
X Coordinate
Y Coordinate
Customer demand
Customer availability start time
Customer availability end time
Customer service time
Depot
1st customer
2nd customer

Figure 4.1: Typical Structure of Solomon Datafile (Solomon, 1987)

In this data file, the first customer is located at the coordinate x, which is 45, and the coordinate y, which is 68. It has 10 demand units that are ready to be loaded into the service vehicle. This first customer can only be served between the time units of 912 and 967, and the serving time is 90 units.

The second customer is located at the coordinate x, which is 45, and the coordinate y, which is 68. It has 30 units of demand that are ready to be loaded in the serving vehicle. This first customer can only be served between the time units of 825 and 870, and the serving time is 90 units.

4.3.1.2 Characteristics of Geographical Distribution

In the customers' geographical distribution, there are 3 types of datasets. The first type is the cluster type whereby the customers are grouped in clusters and there are 17 instances in this type. The second type is the random type, and the customers are randomly distributed which has 23 instances. Lastly, the combination of random and cluster-type distribution of customers. This final combination has customers distributed randomly and clustered in groups and it has 16 instances. The C1 and C2 types belong to cluster types of customers distribution as shown in Figures 4.2 and Figure 4.3. In the C1 type, we can observe the 10 customers.

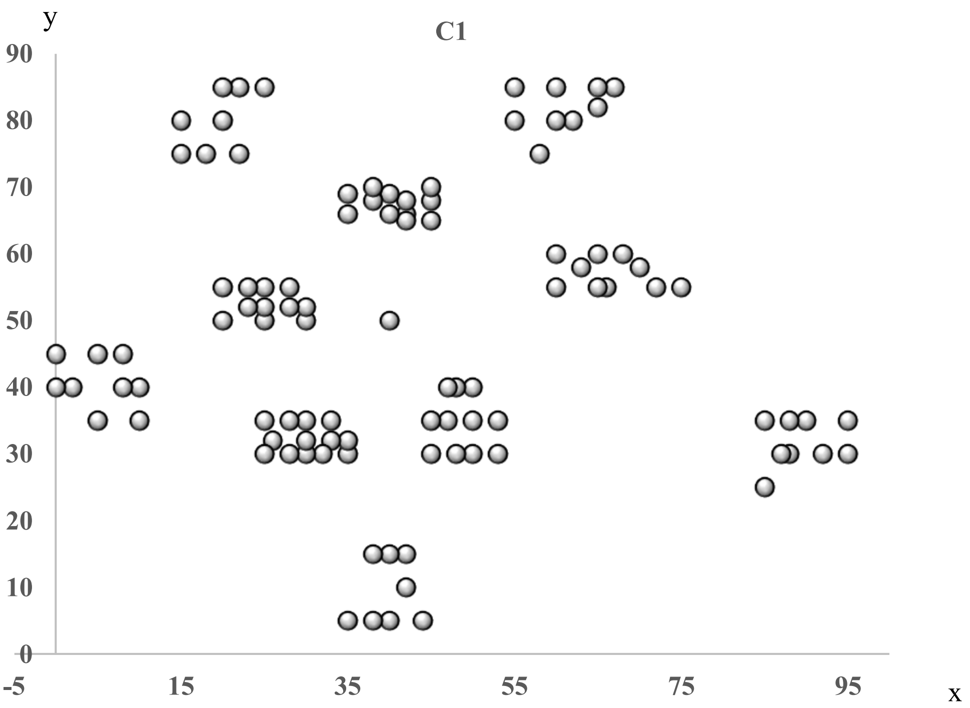


Figure 4.2: C1 Type (Solomon, 1987)

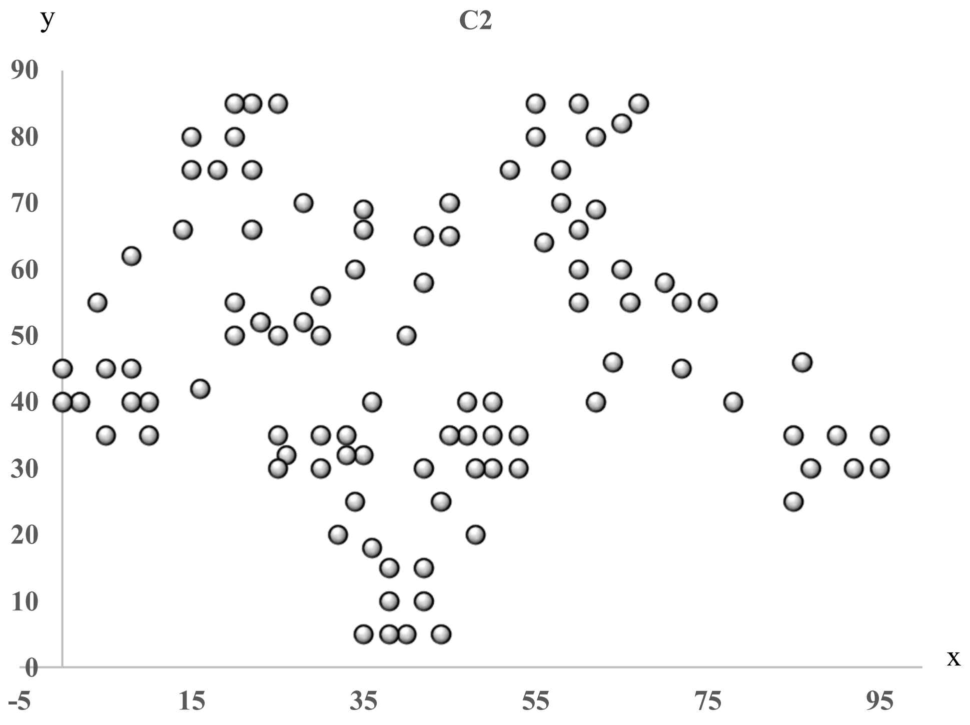


Figure 4.3: C2 Type (Solomon, 1987)

that clustered together. However, in the C2 type, cluster formation is not clear, and some customers are hard to differentiate if they are clustered together. Customers in both C1 type and C2 type spread across the range of 0 to 100 in the x coordinate and 0 to 90 in the y coordinate. This means the best solution can be obtained from this cluster.

In the R1 and R2 types, customers are randomly distributed as shown in Figures 4.4 and Figure 4.5. Both R1 and R2 types have a similar customer geographical distribution but different time window and their customers are distributed within the range of 0 to 70 in x coordinate and 0 to 80 in y coordinate. Finally, the RC1 and RC2 types have customers randomly distributed in a cluster, as shown in Figures 4.6 and 4.7. Both RC1 and RC2 types have their customers distributed within the range of 0 to 100 in x coordinate and 0 to 90 in y coordinate.

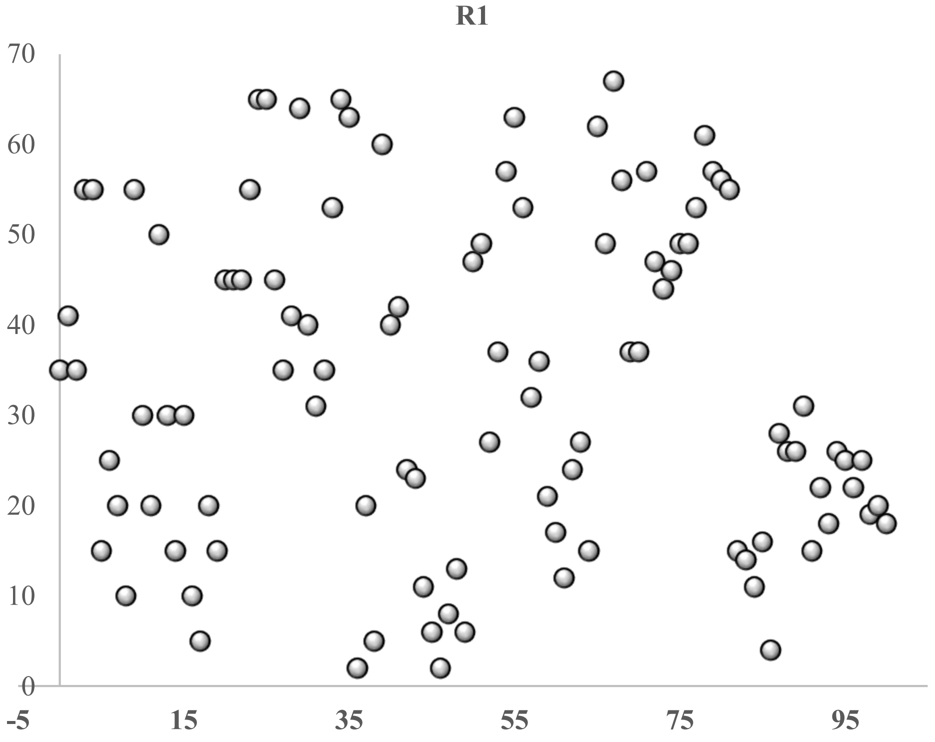


Figure 4.4: R1 Type (Solomon, 1987)

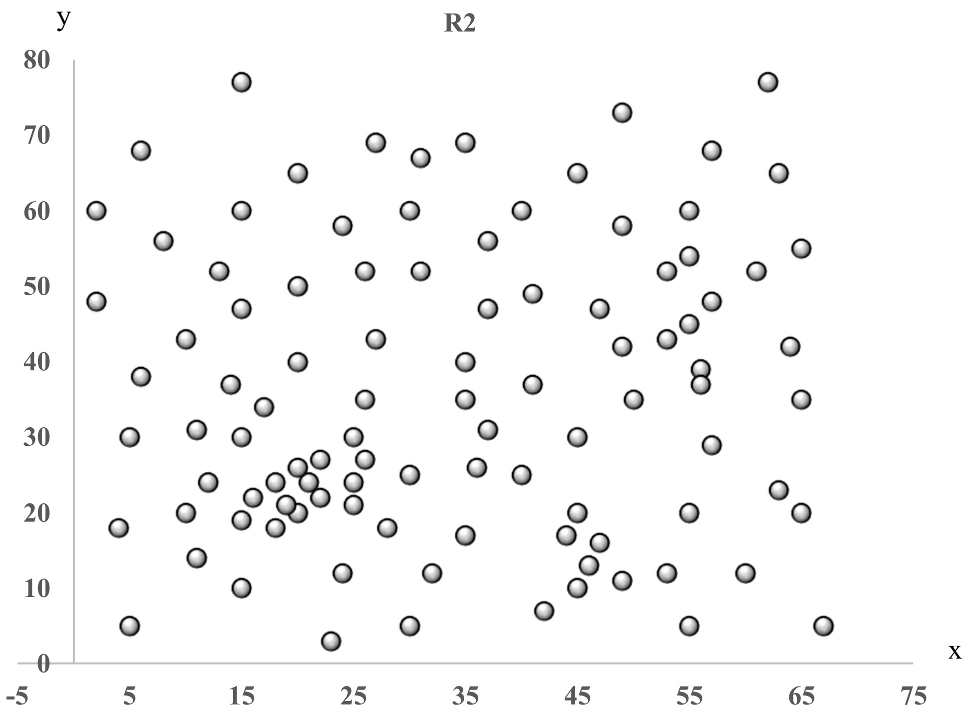


Figure 4.5: R2 Type (Solomon, 1987)

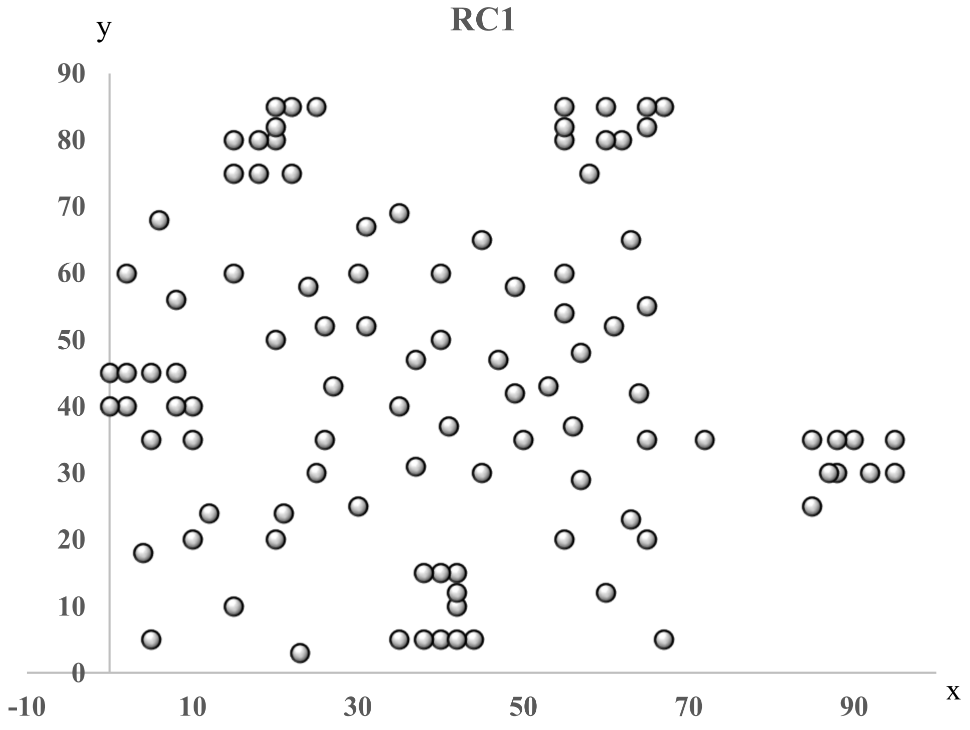


Figure 4.6: RC1 Type (Solomon, 1987)

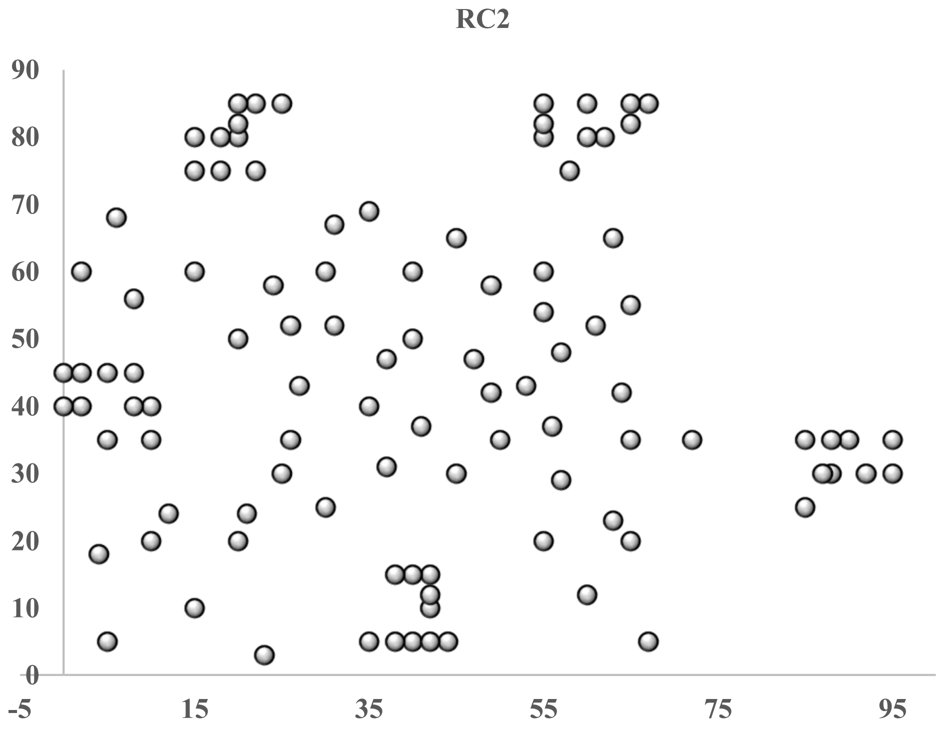


Figure 4.7: RC2 Type (Solomon, 1987)

4.3.1.3 Characteristics of Time Windows

The Solomon datasets contain two types of time windows. The first type has a narrow time window, and the second type has a broader time window. Instance type C1, R1, and RC1 have narrow time windows compared to instance type C2, R2, and RC2, which has a broader time window. In the narrow time windows, the C1 type has a time window ranging from 0 to 1300 while the R1 and RC1 have time windows ranging from 0 to 250. In the broader time window, the C2 type has time windows ranging from 0 to 3400 and the R2 and RC2 types have a time window ranging from 0 to 1000. The narrow time windows have less feasible solutions and longer waiting times than the broader time windows. In Solomon's dataset, there are 3 types of time windows. Instances in Solomon's dataset can be narrow time windows, broader time windows, and a combination of the narrow time window and broader time windows.

In service time, it is categorized into two types. The first type of service time has 90 units, and the second type has 10 units. The C and RC types have 90 units, and the R type has 10 units. The demand unit that appears in discrete values is in a multiple of 10 and has a minimum value of 10 and a maximum value of 40. The demand units vary accordingly to the instances.

4.3.2 MOVRPTW Dataset

Various datasets are used to evaluate the performance of VRPTW (Ahmmed *et al.*, 2008; Kaiwartya, Kumar, D K Lobiyal, *et al.*, 2015; Saint-Guillain, Deville and Solnon, 2015; Jacobsen-Grocott *et al.*, 2017). Realistically speaking, the VRPTW datasets used for performance evaluation are not real-life if we calculate their travel distance, travel time, time windows, demand, and service time. This is due to the orograph[1] of the location, customer activities, and urban or rural areas among others. Therefore, a real-life dataset is needed to address this problem.

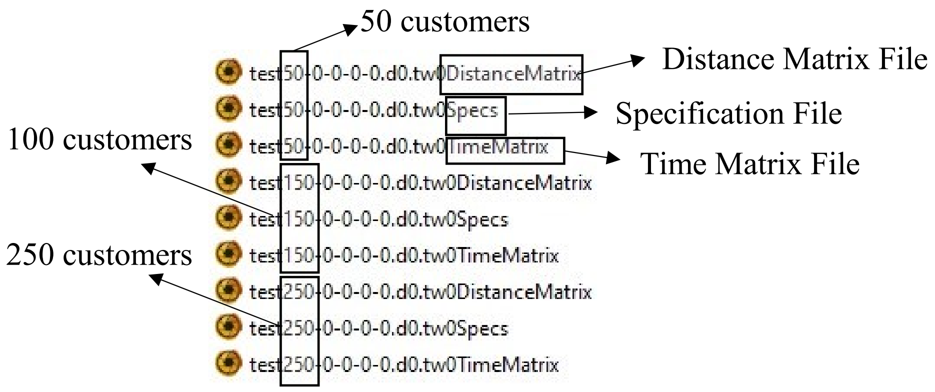
The MOVRPTW (Castro-Gutierrez, Landa-Silva and Pérez, 2011) dataset is a new set of MOVRPTW benchmarking. It obtains real data from a distribution company in Tenerife, Spain whose core business is to provide food products delivery which serves around 150 customers per day which is equivalent to around 1000 customers per week. The travel distance and travel time between the customers are based on the Google Maps database. This means that the travel distance and travel time matrices are unique and non-symmetrical, which provides a true representation of travel distance and travel time. Hence, the travelling time in urban areas is bound to be more time-consuming than the travelling time in rural areas. Also, the travel distance and travel time are different, and this is not the case in Solomon datasets. Therefore, different scenarios can be formed by having differences in travel distance and

[1] https://www.thefreedictionary.com/Orograph

travel time, time windows, and demand specifications, and this reflects the real information provided by the company.

4.3.2.1 File Structure

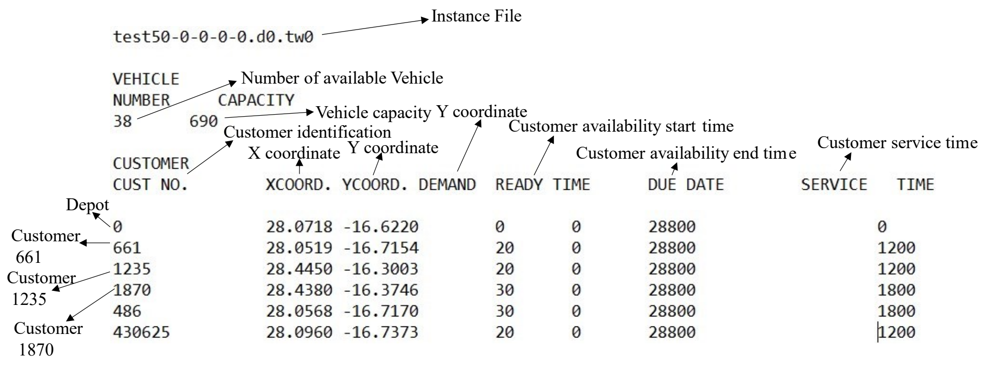
Currently, MOVRPTW datasets only contain 3 customer sizes. They are 50, 150, and 250 customers, as shown in Figure 4.8. Each customer size has 15 different sets of datasets. Each set has three associated files. They are the distance matrix, time matrix, and specification files.



test50-0-0-0-0.d0.tw0DistanceMatrix
test50-0-0-0-0.d0.tw0Specs
test50-0-0-0-0.d0.tw0TimeMatrix
test150-0-0-0-0.d0.tw0DistanceMatrix
test150-0-0-0-0.d0.tw0Specs
test150-0-0-0-0.d0.tw0TimeMatrix
test250-0-0-0-0.d0.tw0DistanceMatrix
test250-0-0-0-0.d0.tw0Specs
test250-0-0-0-0.d0.tw0TimeMatrix

Figure 4.8: Structure of MOVRPTW Files (Castro-Gutierrez, Landa-Silva and Pérez, 2011)

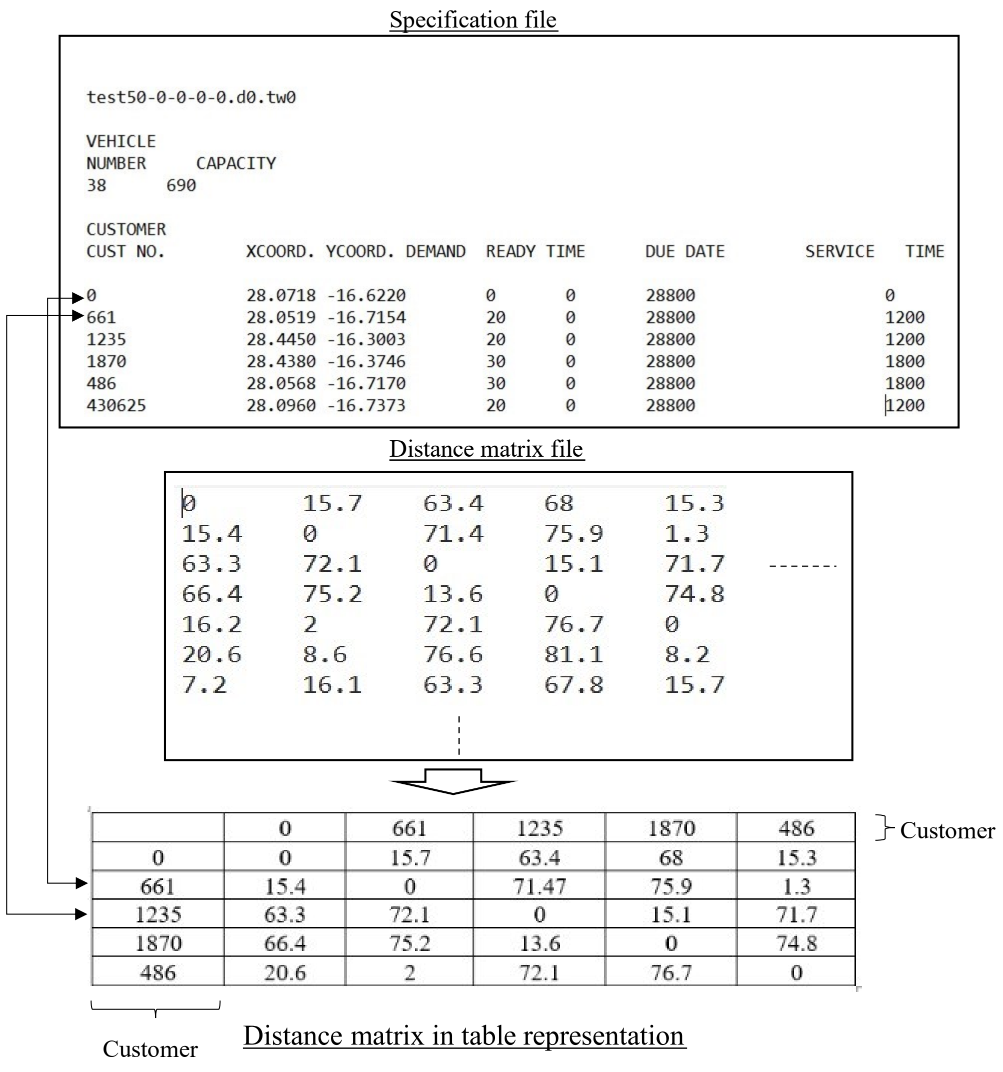
First, the distance matrix contains the distance information between different customers. Second, the time matrix represents the travel time information between different customers, and third, the specification file contains information on fleet maximum size, vehicle capacity, customer location, customer time windows (availability time), demand unit, ready time, service time, and customer identification. The specification file has a similar structure to the Solomon instances. Figure 4.9 shows the internal MOVRPTW specification file.



```
test50-0-0-0-0.d0.tw0

VEHICLE
NUMBER      CAPACITY
38          690

CUSTOMER
CUST NO.        XCOORD. YCOORD. DEMAND  READY TIME      DUE DATE        SERVICE   TIME

0               28.0718 -16.6220        0       0       28800                   0
661             28.0519 -16.7154        20      0       28800                   1200
1235            28.4450 -16.3003        20      0       28800                   1200
1870            28.4380 -16.3746        30      0       28800                   1800
486             28.0568 -16.7170        30      0       28800                   1800
430625          28.0960 -16.7373        20      0       28800                   1200
```

Figure 4.9: Structure of MOVRPTW specification file (Castro-Gutierrez, Landa-Silva and Pérez, 2011)

In the distance matrix, the structure of the file is represented in two dimensions. Figure 4.10 shows the distance matrix is derived from the specification file and the distance matrix file. In this table, one can observe that the distance from customer 661 to the depot (0) is not symmetrical with the distance from the depot(0) to customer 661. The distance from customer 661 to the depot (0) is 15.7 and the distance from the depot(0) to customer 661 is 15.4. Similarly, in the time matrix file, the structure of the file is represented in two dimensions. Figure 4.11 shows that the time matrix is derived from the specification file and the time matrix file. In this table, the distance from customer 661 to the depot (0) is not symmetrical with the distance from the depot (0) to customer 661. The time to travel from customer 661 to the depot(0) is 1020 and the distance from the depot(0) to customer 661 is 960.



```
test50-0-0-0-0.d0.tw0

VEHICLE
NUMBER     CAPACITY
38      690

CUSTOMER
CUST NO.        XCOORD. YCOORD. DEMAND  READY TIME      DUE DATE        SERVICE   TIME

0               28.0718 -16.6220        0       0       28800                   0
661             28.0519 -16.7154        20      0       28800                   1200
1235            28.4450 -16.3003        20      0       28800                   1200
1870            28.4380 -16.3746        30      0       28800                   1800
486             28.0568 -16.7170        30      0       28800                   1800
430625          28.0960 -16.7373        20      0       28800                   1200
```

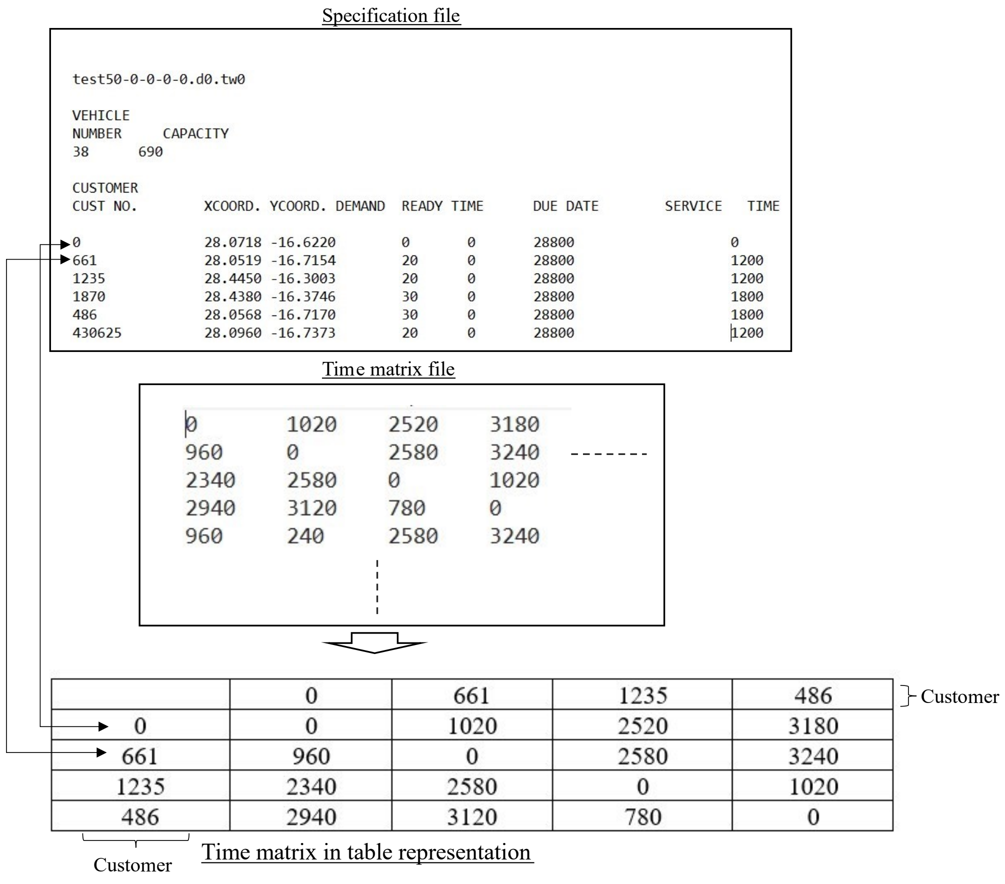
```
0       15.7    63.4    68      15.3
15.4    0       71.4    75.9    1.3
63.3    72.1    0       15.1    71.7    -------
66.4    75.2    13.6    0       74.8
16.2    2       72.1    76.7    0
20.6    8.6     76.6    81.1    8.2
7.2     16.1    63.3    67.8    15.7
```

	0	661	1235	1870	486
0	0	15.7	63.4	68	15.3
661	15.4	0	71.47	75.9	1.3
1235	63.3	72.1	0	15.1	71.7
1870	66.4	75.2	13.6	0	74.8
486	20.6	2	72.1	76.7	0

Figure 4.10: Structure of MOVRPTW distance matrix file (Castro-Gutierrez, Landa-Silva and Pérez, 2011)

4.3.2.2 Characteristics of Geographical Distribution

Unlike Solomon datasets, which have three types of geographical distribution of customers, such as clustering, randomly distributed, and a combination of clusters and randomly distributed. The MOVRPTW type of customer distribution only has customers randomly distributed and clustered.



```
test50-0-0-0-0.d0.tw0

VEHICLE
NUMBER     CAPACITY
38      690

CUSTOMER
CUST NO.        XCOORD. YCOORD. DEMAND  READY TIME      DUE DATE        SERVICE   TIME

0               28.0718 -16.6220        0       0       28800                   0
661             28.0519 -16.7154        20      0       28800                   1200
1235            28.4450 -16.3003        20      0       28800                   1200
1870            28.4380 -16.3746        30      0       28800                   1800
486             28.0568 -16.7170        30      0       28800                   1800
430625          28.0960 -16.7373        20      0       28800                   1200
```

```
0       1020    2520    3180
960     0       2580    3240   -------
2340    2580    0       1020
2940    3120    780     0
960     240     2580    3240
```

	0	661	1235	486
0	0	1020	2520	3180
661	960	0	2580	3240
1235	2340	2580	0	1020
486	2940	3120	780	0

Figure 4.11: Structure of MOVRPTW time matrix file (Castro-Gutierrez, Landa-Silva and Pérez, 2011)

In the Solomon dataset, there are four different types of the geographical distribution of customers such as C1XX, C2XX, RXXX, and RCXXX. However, in the MOVRPTW dataset, there are two types of layout. These two layouts are generated using two random seeds which are 0 and 10 in a custom dataset generator (Ghoseiri and Farid, 2010) as shown in Figures 4.12 and Figure 4.13.

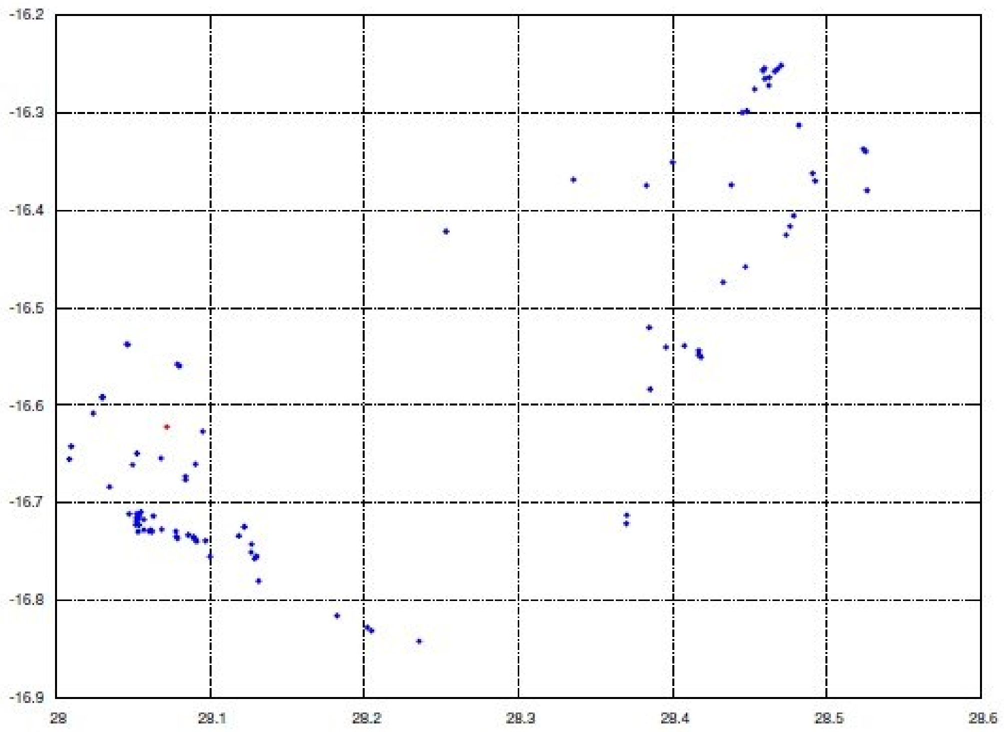


Figure 4.12: Customers distribution using seed 0 (Castro-Gutierrez et al. 2011)

In MOVRPTW, there are three unique features for the location of customers. Firstly, there are two clusters located at the capital (upper right corner) and touristic areas (lower left corner). These two areas have different travel distances and travel times which is due to congestion and speed limits. Secondly, The customers are distributed unevenly within the latitudes of 28 and 28.6 and longitudes of -16.9 and -16.2. The depot is located at latitude and longitude of -16.78 and 28.07 instead of central.

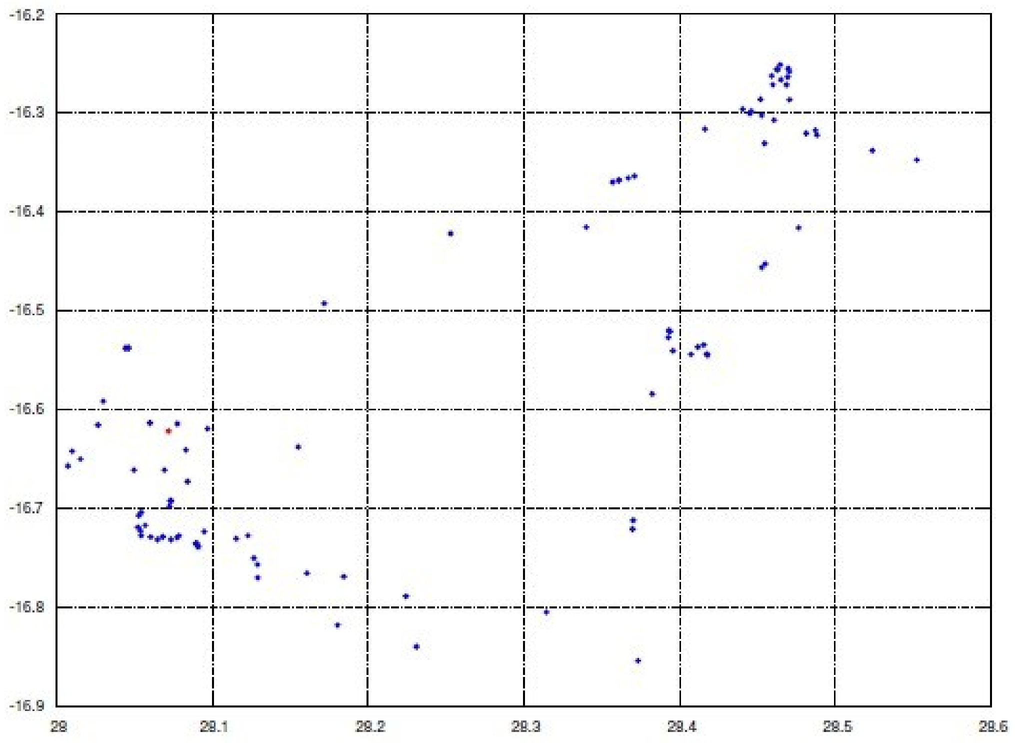


Figure 4.13: Customers distribution using seed 10 (Castro-Gutierrez, Landa-Silva, and Pérez, 2011)

Thirdly, the travel distance is obtained using the Google Maps database. This mimics the real road distance. The distance is not symmetrical between customers and depends on factors such as average transit vehicle speeds. These scenarios present a real-life assessment in multi-objective algorithms.

4.3.2.3 Characteristics of Time Windows

The time windows specify the availability time of the customers, and in the MOVRPTW dataset, their window specifications are crafted to follow the real scenario of commercial activities, as shown in Figure 4.14. There are 5 different time window profiles and 3 types of customers with different time windows. The three types of customers are the early type of customers, midday type of customers, and late type of customers. The early type of customer

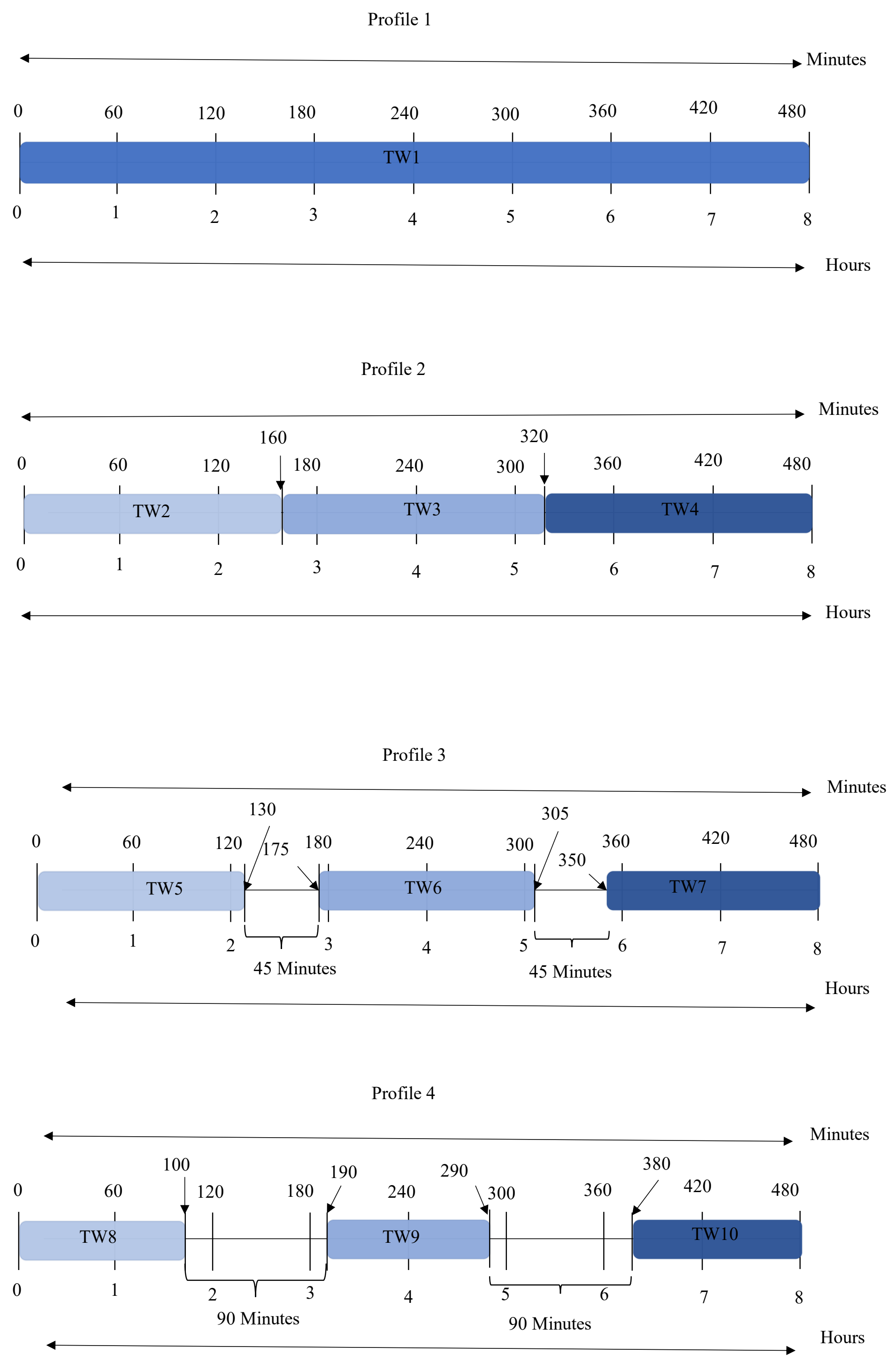


Figure 4.14: Time Windows Profiles (Castro-Gutierrez, Landa-Silva, and Pérez, 2011)

is the customers to be served in the morning. The midday type of customers are the customers to be served at midday and the latest type of customers are the customers to be served at the latest. To allocate the time range for each type of customer, the total operating hour of the depot is divided into 3 parts of time windows to reflect the 3 types of customers. The total operating hours of a depot is 8 hours (480 minutes), if it is divided into 3 parts, each type of customer will have 160 minutes availability times. This means that the early customer is available in the range of 0 to 160 minutes, the midday customer is available in the range of 160 minutes to 320 minutes, and the latest type of customer is available in the range of 320 minutes to 480 minutes.

In this profile, the customers are available for 8 hours. There is no customer unavailability gap in this profile. The first-time windows profiles allow the customers to be always available for 8 hours (480 minutes).

In the second profile, the time windows are divided into 3 parts. The first part of the time window is the early type of customer which can be served from 0 to 160 minutes. The second part is the midday customer, which can be served within the range of 160 to 320 minutes, and the final part is the latest customer, who can be served within the range of 320 to 480 minutes. This means that the length of the time windows is 160 minutes.

In the third profile, the customer unavailability gap is introduced and set to 45 minutes between each type of customer, and the time windows length is 130 minutes. The early type of customers is available between 0 to 130 minutes.

For the midday type of customers, it is available between the range of 175 minutes to 305 minutes, and the latest type of customer is available between the range of 350 minutes to 480 minutes. In this profile, the time window length is 130 minutes.

In the fourth profile, the customer unavailability gap is set to 90 minutes between each type of customer. The early type of customers is available between 0 to 100 minutes. For the midday type of customers, it is available between the range of 190 minutes to 290 minutes, and the latest type of customers is available between the range of 380 minutes to 480 minutes. This profile has 100 minutes time windows length.

There are 10 types of time windows if we add all the time window types from profile 1 to profile 4. The fifth profile contains the time windows from profiles 1, 2, 3, and 4. The purpose of having these types of time windows is to cover a wider range and realistic scenarios.

In the travel times calculations, these figures are obtained from the Google Maps database. Travel times between customers are not the same and are subject to traffic and speed limits. Therefore, the first customer's travel times to the second customer are not the same as the other way round.

4.3.2.4 Formation of Dataset

The dataset is formed based on the following combinations:

- Number of Customers: MOVRPTW uses 100 customers.
- Time windows: Time windows 1 (TW1) is assigned to types of instances in profile 1. Time windows 2, 3, and 4 are randomly selected and applied to types of instances in profile 2. Time windows 4, 5, and 6 are randomly selected and applied to types of instances in profile 3. Time windows 7, 8, and 9 are randomly selected and applied to types of instances in profile 4, and time windows 1 to 10 are randomly selected and apply types of instances in profile 5.
- Customer Demand: The customer demand is not based on real data. It is randomly selected based on three demand values which are 10, 20, and 30, and the three types of slack margin which are 60, 20, and 5. The three types of slack margin reflect a high slack margin (60), a normal low slack margin (20), and a very tight slack margin (5).
- Service times: These service times are based on real data provided by the distribution company and are assigned based on several factors, such as customer activity, location, and time of day. There are three service times, 10 minutes, 20 minutes, and 30 minutes. The three service times are randomly selected.
- Seeds: Seeds are randomly selected from two sets of seeds. The first seed is 0,0,0,0 and the second is 10,7,5,1.

A total of 30 MOVRPTW instances are generated (1 size * 5 types of time windows profiles * 3 deltas and 2 groups of seeds).

4.3.3 Dynamism Dataset

We use two types of datasets to evaluate the performance of NEDPALNS against other algorithms. One is the Solomon dataset, and the other is the MOVRPTW dataset. Both datasets do not have dynamism features. To qualify for the dynamism scenario, we associate the Lackner (Lackner, 2004) dynamic test datasets with the Solomon dataset and the MOVRPTW dataset. Lackner dataset has five types of dynamic degrees, which are 90%, 70%, 50%, 30%, and 10%. This means that if the instance degree of dynamism (DoD) is 10%, 10% of the customers in the instance are dynamic and the remaining are static customers. However, if the instance DoD is 90%, this means 90% of the customers in the instance are dynamic and the remaining are static customers. Hence, the degree of dynamism is measured as follows:

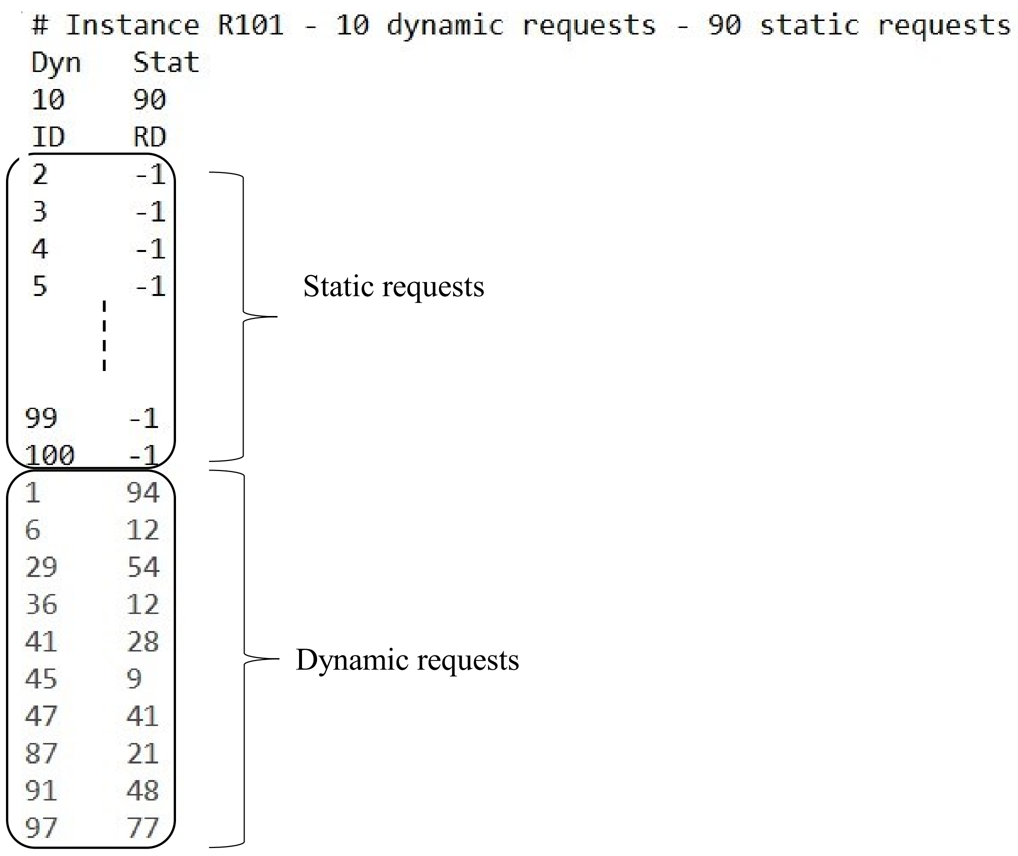
$$\text{DoD (Dynamic of demand)} = \frac{N_d}{N_d + N_s} * 100\% \qquad (4.1)$$

where N_d is the dynamic customer demand and N_s is the static customer demand.

4.3.3.1 File Structure

In the Solomon dataset, the dynamic instance (R101) with a 10% degree of dynamism is shown in Figure 4.15. The static requests have the customers associated with -1 and the dynamic requests have the customers associated with ready times accordingly. Customer 1 is released at time 94. The dynamic instance R101 is named R101_rd_10.txt. This means the R101 instance has 10% dynamic requests (10 customers have information available during execution) or dynamic information and 90% static requests (90 customers have information available during planning) or static information. If we combine the instances with the five degrees of dynamism, the total instances are 280 instances.

For MOVRPTW datasets, there are no dynamic instances. To support dynamism, we develop a program to read all customers from the instance and randomly select the customers based on the degree of dynamism. For example, if the instance contains 50 customers and the degree of dynamism is 10%, 5 customers are randomly selected, and each customer is assigned a random time based on the customer availability time. For example, if customer 661 is chosen as a dynamic request, the program will refer to the instance file and check the customer availability time. If the customer availability time is between 0 and 28800, the dynamic time will be randomly generated between the customer availability time. In this case, the generated dynamic time is 12,800 units.

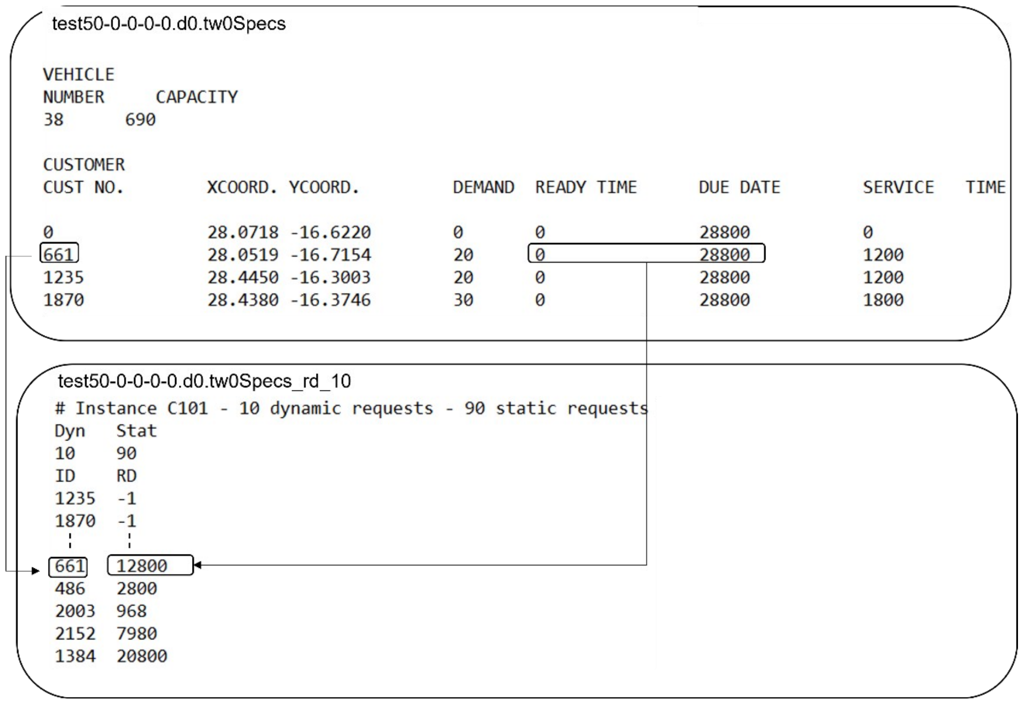


```
# Instance R101 - 10 dynamic requests - 90 static requests
Dyn   Stat
10    90
ID    RD
2      -1
3      -1
4      -1
5      -1

99     -1
100    -1
1      94
6      12
29     54
36     12
41     28
45     9
47     41
87     21
91     48
97     77
```

Static requests
Dynamic requests

Figure 4.15: Solomon instance R101 dynamic data file (Lackner, 2004)

Figure 4.16 shows the MOVRPTW instance (test50-0-0-0-0.d0.tw0Spec.txt) and the corresponding generated dynamic instance (test50-0-0-0-0.d0.tw0Specs_rd_10.txt). This data file (test50-0-0-0-0.d0.tw0Specs_rd_10.txt) contains 50 customers using profile 1 with 10% dynamic requests. The dynamic dataset used for testing NEDPALNS can be found at https://github.com/tskhoo/MODVRPTW.



```
VEHICLE
NUMBER     CAPACITY
38      690

CUSTOMER
CUST NO.            XCOORD. YCOORD.         DEMAND  READY TIME      DUE DATE        SERVICE   TIME

0                   28.0718 -16.6220        0       0               28800           0
661                 28.0519 -16.7154        20      0               28800           1200
1235                28.4450 -16.3003        20      0               28800           1200
1870                28.4380 -16.3746        30      0               28800           1800
```

```
# Instance C101 - 10 dynamic requests - 90 static requests
Dyn   Stat
10    90
ID    RD
1235  -1
1870  -1
 ¦     ¦
661   12800
486   2800
2003  968
2152  7980
1384  20800
```

Figure 4.16: MOVRPTW Dynamic Datafile

(Test50-0-0-0-0.d0.tw0Spec_rd_10.txt)

4.4. Parameter settings

These experiments are carried out at the University of Tunku Abdul Rahman's Apple iMac lab, Investment and Trading Strategies Lab, and Numerical and High-Performance Computing Lab. NEDPALNS algorithm is developed in Java programming and capable of running on any operating system (i.e., Mac OS, Linux, and Windows). The parameters set in the NEDPALNS algorithm are listed in Table 4.2. The parameter values are randomly set high in value are the generation parameter, the number of runs parameter, size of the population, generation, destroy, and repair iteration. This high value is to enable a thorough execution. In the mutation ratio, this value is set low to reduce the chance of exploring more search space.

Table 4.2: Parameters Setting

Parameter	Value
Population size	750
Mutation ratio	0.1
Number of runs or independent runs	10
Number of threads	8
Number of parallel iterations	100
Maximum promising solution pool size	40
Penalization for unserved customers	0.10
Minimum proportion of customers to be removed	0.10
Maximum proportion of customers to be removed	0.40
Reference objective degradation	0.05
Initial probability of accepting a degrading solution	0.5
Fraction of the initial temperature to be reached at the end	0.002
Reaction factor	0.40
Score for new best solution	1.00
Score for improving solution	0.25
Score for non-improving accepted solution	0.40
Score for rejected solution	0.00
Operator probability (w_) update frequency	100
Reference points – Static dataset	(1.5,1.5)
Reference points – Dynamic dataset	(1.5,1.5,1.5)

4.5. Results

To compare our results with the published papers related to our study, we adopt the quantitative and qualitative metrics for comparations and what to compare.

In the MODVRPTW static assessment, each instance is run 30 times using the Solomon dataset. In the MODVRPTW dynamic assessment, we perform two assessments. The first assessment is based on a hypothetical type of dataset which uses the Solomon dataset with the dynamic dataset. The second assessment is based on the real type of dataset which uses the MODVRPTW dataset with the dynamic dataset. In the second assessment, each instance runs 10 times.

The results are assessed based on qualitative and quantitative. In the quantitative assessment, the metrics used are the average, the best, and the worst result of the total travelled distance, the number of vehicles, insertion rates, and rejection rates, as well as counting the highest number of non-dominated solutions.

In qualitative metrics, we use hypervolume (Zitzler and Thiele, 1998) to assess the performance. The hypervolume result is a single element indicator metric and is rigorous monotonic with Pareto dominance. First, an approximation set A is obtained and a given reference point is used to calculate the hypervolume. A higher hypervolume value indicates better convergence and diversity in minimum multi-objective optimization problem context,

Second, we normalized the objective values to a range of 0–1 to obtain the hypervolume values. This is because to a large extent, objective values may differ. We perform the normalization based on 2 criteria. The first criterion is the static dataset that focuses on static information in MOVRPTW, and the reference points are set at (1,5, 1.5). The second criterion is the dynamic data set that focuses on the dynamic information in MODVRPTW, and the reference points are set at (1.5,1.5,1.5).

4.5.1 Comparisons with Published Algorithms Using the Static Dataset (Solomon Dataset)

In these comparisons, we use the static dataset (Solomon dataset) to compare the published algorithms. The published algorithms are the multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Qi *et al.*, 2015a), multi-objective goal programming, and genetic algorithm (MOGPGA) (Ghoseiri and Farid, 2010), multi-objective evolutionary algorithm (MOEA) (Najera, 2010). We also compare the best-known solutions (BKS) and the BKS by minimum average results after 30 independent runs. The "NumNs" is the number of non-dominated solutions. The "N/A" is no result from the comparing algorithm. The "NV" is the number of used vehicles, and the "TD" is the total travelled distance. Both "NV" and "TD" are extracted from the 30 independent runs and the lowest number is extracted. In this section, the customer size used for testing is 100 customers. Our analysis is assessed based on the quantities of the non-dominated solutions in instance types C1, C2, R1, R2, RC1, and RC2. The “MNV” is the average number of used vehicles after 30 independent runs. The “MTD” is the average total travelled distance after 30 independent runs. These metrics measure the best results on average after 30 independent runs. The %MNV and %MTD represent the different values from the comparing algorithm in percentages on the respective average number of vehicles and average total travelled distance.

In Appendix A1, NEDPALNS slightly underperforms other published algorithms. The difference results in instance C109 (829.71) compares to other published algorithms (828.94), which is insignificant (0.008%).

The results in Appendix A2 show that NEDPALNS (8) has similar non-dominated solutions to other published algorithms (8) except for MOGPGA which only has 6 non-dominated results. Appendix A3 – A6 show NEDPALNS (R1 – 20, R2 – 27, RC1 – 14, and RC2 – 21) outperform other published algorithms in instance type R1 (M-MOEA/D – 2, MOGPGA – 5, MOEA – 6), R2 (M-MOEA/D – 7, MOGPGA – 2, MOEA – 1), RC1 (M-MOEA/D – 2, MOGPGA – 4, MOEA – 1) and RC2 (M-MOEA/D – 8, MOGPGA – 1, MOEA – 3) with the highest number of non-dominated solutions. These results show that NEDPALNS has better diversity and convergence results than other published algorithms in the instance type R1, R2, RC1, and RC2 on 100 customers.

Table 4.3 shows the hypervolume comparisons in which NEDPALNS (44 instances) has better results than the M-MOEA/D (27 instances), MOGPGA (16 instances), and MOEA (19 instances). NEDPALNS results accounted for 78.6% of the total instances has better hypervolume than other published algorithms. This indicates that NEDPALNS has significant hypervolume performance, better convergence, and a wide diversity than other published algorithms. Table 4.4 shows the comparison of the least average result with other published solutions. The “Min TD” represents the lowest average of the total travelled distance after 30 independent runs. In these comparisons, the results are not available in M-MOEA/D and MOEA. NEDPALNS outperforms MOGPGA results by 35 instances, which accounted for 62.5% of the total instances. The remaining 21 instances (37.5%) show non-dominated solutions like MOGPGA. This shows that NEDPALNS has more than 50% instances with better least average results against the MOGPGA.

TABLE 4.3: A comprehensive comparison of the obtained non-dominated solutions using hypervolume indicator

Inst-ance	**M-MOEA/D (Qi et al., 2015)**	**MOGPGA (Ghoseiri & Farid, 2010)**	**MOEA (Najera, 2010)**	**NED-PALNS**	**Inst-ance**	**M-MOEA/D (Qi et al., 2015)**	**MOGPGA (Ghoseiri & Farid, 2010)**	**MOEA (Najera, 2010)**	**NED-PALNS**
C101	**0.2505**	**0.2505**	**0.2505**	**0.2505**	C201	**0.25**	**0.25**	**0.25**	**0.25**
C102	**0.2505**	**0.2505**	**0.2505**	**0.2505**	C202	**0.25**	**0.25**	**0.25**	**0.25**
C103	**0.251**	**0.251**	**0.251**	**0.251**	C203	**0.2503**	**0.2503**	**0.2503**	**0.2503**
C104	**0.253**	**0.253**	**0.253**	**0.253**	C204	**0.2508**	0.2429	**0.2508**	**0.2508**
C105	**0.2505**	**0.2505**	**0.2505**	**0.2505**	C205	**0.2523**	**0.2523**	**0.2523**	**0.2523**
C106	**0.2505**	**0.2505**	**0.2505**	**0.2505**	C206	**0.2526**	**0.2523**	**0.2526**	**0.2526**
C107	**0.2505**	**0.2505**	**0.2505**	**0.2505**	C207	**0.2528**	0.25	**0.2528**	**0.2528**
C108	**0.2505**	**0.2505**	**0.2505**	**0.2505**	C208	**0.2527**	**0.2527**	**0.2527**	**0.2527**
C109	**0.2505**	**0.2505**	**0.2505**	0.25					
Average	**0.2508**	**0.2508**	**0.2508**	**0.2508**	Average	**0.2514**	0.2501	**0.2514**	**0.2514**

R101	0.2854	0.2827	**0.286**	**0.286**	R201	0.5442	0.4564	0.5433	**0.5473**
R102	0.4034	0.3647	0.4033	**0.4035**	R202	0.6288	0.6082	0.6302	**0.6377**
R103	**0.6557**	0.5962	0.6543	**0.6557**	R203	0.8651	0.7955	0.8573	**0.8701**
R104	0.9083	**0.9191**	0.9127	0.9175	R204	0.9759	0.8707	0.9707	**0.9807**
R105	0.5468	0.5055	0.5485	**0.5504**	R205	**0.8207**	0.6952	0.8109	0.8155
R106	0.6791	0.6311	0.683	**0.6841**	R206	**0.8724**	0.839	0.8637	0.8708
R107	0.8507	0.7969	0.8519	**0.8589**	R207	0.9304	0.8638	0.9242	**0.9374**
R108	0.9243	0.9204	0.9732	**0.9864**	R208	**1.1676**	0.9557	1.1624	1.1526
R109	0.7299	0.7021	0.7303	**0.7318**	R209	**0.884**	0.6568	0.7355	0.8782
R110	0.7983	0.7293	0.8081	**0.8173**	R210	**0.8483**	0.8401	0.8385	0.8456
R111	0.8167	0.7951	0.8237	**0.8697**	R211	0.9598	0.7006	0.9474	**0.9677**
R112	0.9148	0.852	0.9151	**0.9287**					
Average	0.7094	0.6746	0.7158	**0.7242**	Average	0.8634	0.7529	0.844	**0.864**
RC101	0.3482	0.3015	0.3238	**0.3574**	RC201	**0.6258**	0.5386	0.6154	0.5795
RC102	0.4499	0.4029	0.4517	**0.4593**	RC202	0.7621	0.5781	0.7578	**0.767**
RC103	0.6296	0.5569	0.6281	**0.6357**	RC203	**0.9776**	0.8286	0.8829	0.9016
RC104	0.7423	0.6727	0.743	**0.7474**	RC204	1.1093	1.0189	1.1062	**1.1113**
RC105	0.3934	0.333	0.3966	**0.3969**	RC205	**0.6988**	0.5484	0.6854	0.6481
RC106	0.5264	0.4751	0.5359	**0.5375**	RC206	0.8603	0.7066	**0.8666**	0.8061
RC107	0.6471	0.652	0.6565	**0.6576**	RC207	**0.9412**	0.8197	0.9275	0.8695
RC108	0.7441	0.6822	0.7524	**0.755**	RC208	1.0954	1.0213	1.1123	**1.1144**
Average	0.5601	0.5095	0.561	**0.5684**	Average	**0.8838**	0.7575	0.8693	0.8497

TABLE 4.4: Comparison of the least average result with other published algorithms

Instance	**Min TD**									
	M-MOEA/D (Qi *et al.*, 2015b)		**MOGPGA** (Ghoseiri and Farid, 2010)				**MOEA (Najera, 2010)**		**NEDPALNS**	
	MNV	**MTD**	**MNV**	**MTD**	**%MNV**	**%MTD**	**MNV**	**MTD**	**MNV**	**MTD**
C101	N/A	N/A	10	828.94	0.00	0.00	N/A	N/A	**10**	**828.94**
C102	N/A	N/A	10	839.41	0.00	0.01	N/A	N/A	**10**	**828.94**
C103	N/A	N/A	10	849.17	0.00	0.02	N/A	N/A	**10**	**828.06**
C104	N/A	N/A	10	845.56	0.00	0.02	N/A	N/A	**10**	**824.78**
C105	N/A	N/A	10	828.94	0.00	0.00	N/A	N/A	**10**	**828.94**
C106	N/A	N/A	10	828.94	0.00	0.00	N/A	N/A	**10**	**828.94**
C107	N/A	N/A	10	828.94	0.00	0.00	N/A	N/A	**10**	**828.94**
C108	N/A	N/A	10	839.16	0.00	0.01	N/A	N/A	**10**	**828.94**
C109	N/A	N/A	10	828.94	0.00	0.00	N/A	N/A	**10**	**828.94**
Average			10	835.33	0.00	0.01			**10**	**828.38**
C201	N/A	N/A	3	591.56	0.00	0.00	N/A	N/A	**3**	**591.56**
C202	N/A	N/A	3	593.24	0.00	0.00	N/A	N/A	**3**	**591.56**
C203	N/A	N/A	3	614.15	0.00	0.04	N/A	N/A	**3**	**591.17**
C204	N/A	N/A	3	603.94	0.00	0.02	N/A	N/A	**3**	**590.6**
C205	N/A	N/A	3	590.74	0.00	0.00	N/A	N/A	**3**	**588.88**
C206	N/A	N/A	3	592.42	0.00	0.01	N/A	N/A	**3**	**588.49**
C207	N/A	N/A	3	593.24	0.00	0.01	N/A	N/A	**3**	**588.29**
C208	N/A	N/A	3	597.7	0.00	0.02	N/A	N/A	**3**	**588.32**
Average			3	597.12	0.00	0.01			**3**	**589.86**

R101	N/A	N/A	19.4	1673.9	-0.03	0.02	N/A	N/A	20	1642.88
R102	N/A	N/A	18.7	1510.6	0.04	0.02	N/A	N/A	**18**	**1473**
R103	N/A	N/A	14.3	1291.1	0.02	0.06	N/A	N/A	**14**	**1213.81**
R104	N/A	N/A	11.2	1043.3	0.02	0.06	N/A	N/A	**10.93**	**977.05**
R105	N/A	N/A	15.6	1409.6	0.04	0.03	N/A	N/A	**15**	**1360.78**
R106	N/A	N/A	14	1315.9	0.07	0.06	N/A	N/A	**13**	**1239.37**
R107	N/A	N/A	11.8	1134.8	0.07	0.05	N/A	N/A	**11**	**1073.04**
R108	N/A	N/A	10.3	1014.3	0.03	0.07	N/A	N/A	**10.03**	**946.26**
R109	N/A	N/A	13	1220.1	0.00	0.06	N/A	N/A	**13**	**1151.84**
R110	N/A	N/A	12.2	1160.7	0.02	0.07	N/A	N/A	**12**	**1074.81**
R111	N/A	N/A	11.9	1149.1	-0.01	0.08	N/A	N/A	12	1053.5
R112	N/A	N/A	10.5	1051.7	0.05	0.09	N/A	N/A	**10**	**962.02**
Average			13.58	1247.93	0.02	0.05			**13.25**	**1180.7**
R201	N/A	N/A	4	1358.7	-1.00	0.16	N/A	N/A	8	1147.8
R202	N/A	N/A	4	1173.1	-0.50	0.12	N/A	N/A	6	1035.66
R203	N/A	N/A	4.8	1022.3	-0.25	0.14	N/A	N/A	6	874.87
R204	N/A	N/A	5.4	839.82	0.07	0.12	N/A	N/A	**5**	**735.8**
R205	N/A	N/A	3.4	1188.5	-0.47	0.20	N/A	N/A	5	954.32
R206	N/A	N/A	3	1004	-0.66	0.12	N/A	N/A	4.97	880.73
R207	N/A	N/A	3	907.9	-0.33	0.12	N/A	N/A	4	797.99
R208	N/A	N/A	3	778.25	-0.29	0.09	N/A	N/A	3.86	706.69
R209	N/A	N/A	4	1009.9	-0.25	0.15	N/A	N/A	5	860.13
R210	N/A	N/A	3.2	1020.3	-0.88	0.11	N/A	N/A	6	905.21
R211	N/A	N/A	3.6	1191	-0.10	0.36	N/A	N/A	3.97	759.78
Average			3.76	1044.89	-0.40	0.16			5.25	**878.09**
RC101	N/A	N/A	15.3	1693.2	0.02	0.04	N/A	N/A	15	**1623.59**
RC102	N/A	N/A	14.5	1521	0.02	0.04	N/A	N/A	14.21	**1466.02**
RC103	N/A	N/A	12.2	1357.4	0.08	0.06	N/A	N/A	11.17	**1273.58**
RC104	N/A	N/A	11	1213.5	0.09	0.06	N/A	N/A	10	**1136.4**
RC105	N/A	N/A	15.9	1610.5	-0.01	0.06	N/A	N/A	16	**1518.58**
RC106	N/A	N/A	13.5	1437.1	0.04	0.04	N/A	N/A	13	**1376.99**
RC107	N/A	N/A	12.2	1287.9	0.02	0.06	N/A	N/A	12	**1216.78**
RC108	N/A	N/A	11.3	1197.9	0.03	0.06	N/A	N/A	11	**1121.21**
Average			13.24	1414.81	0.03	0.05			12.8	**1341.64**
RC201	N/A	N/A	4	1457	-1.25	0.13	N/A	N/A	9	**1266.15**
RC202	N/A	N/A	4	1381.9	-0.98	0.21	N/A	N/A	7.93	**1095.87**
RC203	N/A	N/A	4.9	1196.7	-0.02	0.23	N/A	N/A	5	**926.82**
RC204	N/A	N/A	3	926.74	-0.31	0.15	N/A	N/A	3.93	**787.74**
RC205	N/A	N/A	4	1411.3	-0.75	0.18	N/A	N/A	7	**1157.55**
RC206	N/A	N/A	4	1195.5	-0.72	0.12	N/A	N/A	6.86	**1057.54**
RC207	N/A	N/A	4	1070.3	-0.50	0.09	N/A	N/A	6	**971.05**
RC208	N/A	N/A	3.7	905.07	-0.08	0.14	N/A	N/A	4	**779.22**
Average			3.95	1193.06	-0.57	0.16			6.22	**1005.24**
Total Sum			8.14	1064.77	-0.06	0.08			8.62	**978.16**
Total Average			455.8	59626.87	-0.06	0.08			482.86	**54776.69**
Count				0						35

Table 4.5 shows the comparison against the best-known solutions. The "Ref" indicates the reference used for the best-known solution in that instance.

The “Min NV” represents the least number of used vehicles, and the “Min TD” represents the least total travelled distance. The solution appears in bold font, indicating that NEDPALNS has a similar or better result than the best-known solution. The “*” shown next to the result indicates that NEDPALNS has a better result than the best-known solution (BKS).

In the “Min NV” comparison, NEDPALNS achieve 18 instances better than the BKS. This represents 32% of the total instances that have results like the BKS. However, in the comparison of “Min TD”, NEDPALNS has 42 instances or 75% of the total instances that have similar or better results than BKS. Of which, there are 8 instances in which NEDPALNS outperforms BKS.

TABLE 4.5: Compare with the best-known solutions (Min NV)

Instance	Min NV						
	Best-known Solution (BKS)			NEDPALNS			
	NV	TD	Ref	NV	TD	%NV	%TD
C101	10	828.94	(Rochat and Taillard, 1995)	**10**	**828.94**	0.00	0.00
C102	10	828.94	(Rochat and Taillard, 1995)	**10**	**828.94**	0.00	0.00
C103	10	828.06	(Rochat and Taillard, 1995)	**10**	**828.07**	0.00	0.00
C104	10	824.78	(Rochat and Taillard, 1995)	**10**	**824.78**	0.00	0.00
C105	10	828.94	(Rochat and Taillard, 1995)	**10**	**828.94**	0.00	0.00
C106	10	828.94	(Rochat and Taillard, 1995)	**10**	**828.94**	0.00	0.00
C107	10	828.94	(Rochat and Taillard, 1995)	**10**	**828.94**	0.00	0.00
C108	10	828.94	(Rochat and Taillard, 1995)	**10**	**828.94**	0.00	0.00
C109	10	828.94	(Rochat and Taillard, 1995)	**10**	**828.94**	0.00	0.00
C201	3	591.56	(Rochat and Taillard, 1995)	**3**	**591.56**	0.00	0.00
C202	3	591.56	(Rochat and Taillard, 1995)	**3**	**591.56**	0.00	0.00
C203	3	591.17	(Rochat and Taillard, 1995)	**3**	**591.17**	0.00	0.00
C204	3	590.6	(Rochat and Taillard, 1995)	**3**	**590.6**	0.00	0.00
C205	3	588.88	(Rochat and Taillard, 1995)	**3**	**588.88**	0.00	0.00
C206	3	588.49	(Rochat and Taillard, 1995)	**3**	**588.49**	0.00	0.00
C207	3	588.29	(Rochat and Taillard, 1995)	**3**	**588.29**	0.00	0.00
C208	3	588.32	(Rochat and Taillard, 1995)	**3**	**588.32**	0.00	0.00
R101	18	1613.59	(Tan, Chew and Lee, 2006)	19	1650.8	0.06	0.02
R102	17	1486.12	(Rochat and Taillard, 1995)	17	1494.15	0.00	0.01
R103	13	1292.68	(Li and Lim, 2003)	13	1351.98	0.00	0.05
R104	9	1007.24	(Mester, 2002)	10	981.23	0.11	-0.03
R105	14	1377.11	(Rochat and Taillard, 1995)	14	1377.33	0.00	0.00
R106	12	1251.98	(Mester, 2002)	12	1263.98	0.00	0.01

R107	10	1104.66	(Shaw, 1997)	10	1131.67	0.00	0.02
R108	9	960.88	(Berger, Barkaoui and Bräysy, 2003)	9	978.33	0.00	0.02
R109	11	1194.73	(Homberger and Hermann, 1999)	12	1153.02	0.09	-0.03
R110	10	1118.59	(Mester, 2002)	11	1078.8	0.10	-0.04
R111	10	1096.72	(Rousseau, Gendreau and Pesant, 2002)	10	1123.37	0.00	0.02
R112	9	982.14	(Gambardella, Taillard and Agazzi, 1999)	10	958.03	0.11	-0.02
R201	4	1252.37	(Homberger and Hermann, 1999)	4	1331.25	0.00	0.06
R202	3	1191.7	(Rousseau, Gendreau and Pesant, 2002)	4	1079.39	0.33	-0.09
R203	3	939.54	(Mester, 2002)	3	972.58	0.00	0.04
R204	2	825.52	(Bent and Hentenryck, 2004)	3	751.06	0.50	-0.09
R205	3	994.42	(Rousseau, Gendreau and Pesant, 2002)	3	1064.71	0.00	0.07
R206	3	906.14	(Schrimpf *et al.*, 2000b)	3	942.08	0.00	0.04
R207	2	837.2	(Bouthilliera and Crainic, 2005)	3	811.51	0.50	-0.03
R208	2	726.75	(Mester, 2002)	2	864.3	0.00	0.19
R209	3	909.16	(Homberger, 2000)	3	1002.82	0.00	0.10
R210	3	938.58	(Ghoseiri and Farid, 2010)	3	1038.78	0.00	0.11
R211	2	892.71	(Bent and Hentenryck, 2004)	3	770.19	0.50	-0.14
RC101	14	1696.94	(Taillard *et al.*, 1997)	14	1708.05	0.00	0.01
RC102	12	1554.75	(Taillard *et al.*, 1997)	13	1477.54	0.08	-0.05
RC103	11	1261.67	(Taillard *et al.*, 1997)	**11**	**1261.67**	0.00	0.00
RC104	10	1135.48	(Cordeau, Laporte and Mercier, 2001)	10	1135.52	0.00	0.00
RC105	13	1629.44	(Berger, Barkaoui and Bräysy, 2003)	14	1540.18	0.08	-0.05
RC106	11	1424.73	(Berger, Barkaoui and Bräysy, 2003)	12	1379.08	0.09	-0.03
RC107	11	1222.1	(Ghoseiri and Farid, 2010)	11	1232.26	0.00	0.01
RC108	10	1139.82	(Taillard *et al.*, 1997)	10	1147.2	0.00	0.01
RC201	4	1406.91	(Mester, 2002)	5	1321.93	0.25	-0.06
RC202	3	1365.65	(Repoussis, Tarantilis and Ioannou, 2009)	4	1214.17	0.33	-0.11
RC203	3	1049.62	(Czech and Czarnas, 2002)	4	947.95	0.33	-0.10
RC204	3	798.41	(Mester, 2002)	3	798.46	0.00	0.00
RC205	4	1297.19	(Mester, 2002)	5	1247.85	0.25	-0.04
RC206	3	1146.32	(Homberger, 2000)	4	1087.93	0.33	-0.05
RC207	3	1061.14	(Bent and Hentenryck, 2004)	4	996.94	0.33	-0.06
RC208	3	828.14	(Ibaraki *et al.*, 2005)	3	829	0.00	0.00
	Count				18		

TABLE 4.5: Compare with the best-known solutions (Min TD)

(continued)

Instance	Min TD						
	Best-known Solution (BKS)			NEDPALNS			
	NV	TD	Ref	NV	TD	%NV	%TD
C101	10	828.94	(Rochat & Taillard, 1995)	**10**	**828.94**	0	0
C102	10	828.94	(Rochat & Taillard, 1995)	**10**	**828.94**	0	0
C103	10	828.06	(Rochat & Taillard, 1995)	**10**	**828.07**	0	0
C104	10	824.78	(Rochat & Taillard, 1995)	**10**	**824.78**	0	0
C105	10	828.94	(Rochat & Taillard, 1995)	**10**	**828.94**	0	0
C106	10	828.94	(Rochat & Taillard, 1995)	**10**	**828.94**	0	0
C107	10	828.94	(Rochat & Taillard, 1995)	**10**	**828.94**	0	0
C108	10	828.94	(Rochat & Taillard, 1995)	**10**	**828.94**	0	0

C109	10	828.94	(Rochat & Taillard, 1995)	**10**	**828.94**	0	0
C201	3	591.56	(Rochat & Taillard, 1995)	**3**	**591.56**	0	0
C202	3	591.56	(Rochat & Taillard, 1995)	**3**	**591.56**	0	0
C203	3	591.17	(Rochat & Taillard, 1995)	**3**	**591.17**	0	0
C204	3	590.6	(Rochat & Taillard, 1995)	**3**	**590.6**	0	0
C205	3	588.88	(Rochat & Taillard, 1995)	**3**	**588.88**	0	0
C206	3	588.49	(Rochat & Taillard, 1995)	**3**	**588.49**	0	0
C207	3	588.29	(Rochat & Taillard, 1995)	**3**	**588.29**	0	0
C208	3	588.32	(Rochat & Taillard, 1995)	**3**	**588.32**	0	0
R101	18	1613.59	(Tan et al., 2006)	20	1642.88	0.11	0.02
R102	18	1454.68	(Tan et al., 2006)	18	1472.82	0	0.01
R103	14	1213.62	(Rochat & Taillard, 1995)	**14**	**1213.62**	0	0
R104	10	974.24	(Tan et al., 2006)	11	976.61	0.1	0
R105	15	1360.78	(Soonchul Jung & Moon, 2002)	**15**	**1360.78**	0	0
R106	13	1240.47	(Soonchul Jung & Moon, 2002)	**13**	**1239.37***	0	0
R107	11	1073.34	(Soonchul Jung & Moon, 2002)	**11**	**1072.12***	0	0
R108	10	947.55	(Soonchul Jung & Moon, 2002)	**10**	**938.2***	0	-0.01
R109	13	1151.84	(Soonchul Jung & Moon, 2002)	**13**	**1151.84**	0	0
R110	12	1072.41	(Soonchul Jung & Moon, 2002)	**12**	**1072.41**	0	0
R111	12	1053.5	(Soonchul Jung & Moon, 2002)	**12**	**1053.5**	0	0
R112	10	953.63	(Rochat & Taillard, 1995)	10	958.03	0	0
R201	9	1144.48	(Alvarenga, Mateus, & de Tomi, 2007)	8	1147.8	-0.11	0
R202	8	1034.35	(Soonchul Jung & Moon, 2002)	6	1034.97	-0.25	0
R203	6	874.87	(Soonchul Jung & Moon, 2002)	**6**	**874.87**	0	0
R204	4	736.52	(Soonchul Jung & Moon, 2002)	**5**	**735.8***	0.25	0
R205	5	954.16	(Ombuki et al., 2006)	**5**	**954.16**	0	0
R206	5	879.89	(Soonchul Jung & Moon, 2002)	5	884.85	0	0.01
R207	4	799.86	(Soonchul Jung & Moon, 2002)	**4**	**797.99***	0	0
R208	4	705.45	(Soonchul Jung & Moon, 2002)	**4**	**705.33***	0	0
R209	5	859.39	(Soonchul Jung & Moon, 2002)	**5**	**860.11**	0	0
R210	5	910.7	(Soonchul Jung & Moon, 2002)	**6**	**905.21**	0.2	-0.01
R211	4	755.96	(Soonchul Jung & Moon, 2002)	**4**	**753.15***	0	0
RC101	15	1623.58	(Rochat & Taillard, 1995)	**15**	**1623.58**	0	0
RC102	14	1461.23	(Soon chul Jung & Moon, 2015)	**14**	**1461.23**	0	0
RC103	11	1261.67	(Taillard et al., 1997)	**11**	**1261.67**	0	0
RC104	10	1135.48	(Cordeau et al., 2001)	**10**	**1135.52**	0	0
RC105	16	1518.58	(Soonchul Jung & Moon, 2002)	**16**	**1518.58**	0	0
RC106	13	1371.69	(Tan et al., 2006)	**13**	**1376.99**	0	0
RC107	12	1212.83	(Soonchul Jung & Moon, 2002)	**12**	**1212.83**	0	0
RC108	11	1117.53	(Soonchul Jung & Moon, 2002)	**11**	**1118.07**	0	0
RC201	6	1134.91	(Tan et al., 2006)	**9**	**1265.56**	0.5	0.12
RC202	8	1095.64	(Soonchul Jung & Moon, 2002)	**8**	**1095.64**	0	0
RC203	5	928.51	(Soonchul Jung & Moon, 2002)	**5**	**926.82***	0	0
RC204	4	786.38	(Soonchul Jung & Moon, 2002)	**4**	**788.66**	0	0
RC205	7	1157.55	(Soonchul Jung & Moon, 2002)	**7**	**1157.55**	0	0
RC206	7	1054.61	(Soonchul Jung & Moon, 2002)	**7**	**1054.61**	0	0
RC207	6	966.08	(Soonchul Jung & Moon, 2002)	**6**	**969.8**	0	0
RC208	4	779.31	(Soonchul Jung & Moon, 2002)	**4**	**778.93**	0	0
Count			_	42			

Table 4.6 shows the least average number of used vehicles and the least total travelled distance compared to the BKS. The "min MNV" refers to the

average of the 30 least number of the used vehicle while the "min MTD" refers to the average of the 30 least total travelled distance. The "min MNV" in BKS results is not made available. However, in the comparison of the least average total travelled distance (min MTD) comparison, NEDPALNS achieves 52 instances that have similar or better results than BKS. This is equivalent to 93% of the total instances. Of the 52 instances, 35 instances (62.5%) have results better than the BKS. This shows that NEDPALNS has achieved significant results in the least average total travelled distance compared to BKS.

TABLE 4.6: Comparison with the least average best-known solution

	Min MNV						
Instance	**Best-known Solution**			**NEDPALNS**			
	MNV	**MTD**	**Ref**	**MNV**	**MTD**	**%MNV**	**%MTD**
C101	N/A	N/A	(Rochat & Taillard, 1995)	10	828.94	N/A	N/A
C102	N/A	N/A	(Rochat & Taillard, 1995)	10	828.94	N/A	N/A
C103	N/A	N/A	(Rochat & Taillard, 1995)	10	828.06	N/A	N/A
C104	N/A	N/A	(Rochat & Taillard, 1995)	10	824.78	N/A	N/A
C105	N/A	N/A	(Rochat & Taillard, 1995)	10	828.94	N/A	N/A
C106	N/A	N/A	(Rochat & Taillard, 1995)	10	828.94	N/A	N/A
C107	N/A	N/A	(Rochat & Taillard, 1995)	10	828.94	N/A	N/A
C108	N/A	N/A	(Rochat & Taillard, 1995)	10	828.94	N/A	N/A
C109	N/A	N/A	(Rochat & Taillard, 1995)	10	828.94	N/A	N/A
Average				10	828.38		
C201	N/A	N/A	(Rochat & Taillard, 1995)	3	591.56	N/A	N/A
C202	N/A	N/A	(Rochat & Taillard, 1995)	3	591.56	N/A	N/A
C203	N/A	N/A	(Rochat & Taillard, 1995)	3	591.17	N/A	N/A
C204	N/A	N/A	(Rochat & Taillard, 1995)	3	590.6	N/A	N/A
C205	N/A	N/A	(Rochat & Taillard, 1995)	3	588.88	N/A	N/A
C206	N/A	N/A	(Rochat & Taillard, 1995)	3	588.49	N/A	N/A
C207	N/A	N/A	(Rochat & Taillard, 1995)	3	588.29	N/A	N/A
C208	N/A	N/A	(Rochat & Taillard, 1995)	3	588.32	N/A	N/A
Average				3	589.86		
R101	N/A	N/A	(Tan et al., 2006)	19	1650.86	N/A	N/A
R102	N/A	N/A	(Rochat & Taillard, 1995)	17	1494.15	N/A	N/A
R103	N/A	N/A	(Li & Lim, 2003)	13.03	1479.93	N/A	N/A
R104	N/A	N/A	(Mester, 2002)	10	981.23	N/A	N/A
R105	N/A	N/A	(Rochat & Taillard, 1995)	14	1378.57	N/A	N/A
R106	N/A	N/A	(Mester, 2002)	12.03	1266.98	N/A	N/A
R107	N/A	N/A	(Shaw, 1997)	10	1190.96	N/A	N/A
R108	N/A	N/A	(Berger et al., 2003)	9.07	991.51	N/A	N/A
R109	N/A	N/A	(Homberger & Hermann, 1999)	12	1153.74	N/A	N/A
R110	N/A	N/A	(Mester, 2002)	11	1082.56	N/A	N/A
R111	N/A	N/A	(Rousseau et al., 2002)	11	1060.33	N/A	N/A

R112	N/A	N/A	(Gambardella et al., 1999)	10	962.32	N/A	N/A
Average				12.34	1224.43		
R201	N/A	N/A	(Homberger & Hermann, 1999)	4.52	1282.66	N/A	N/A
R202	N/A	N/A	(Rousseau et al., 2002)	4	1081.54	N/A	N/A
R203	N/A	N/A	(Mester, 2002)	3.14	997.8	N/A	N/A
R204	N/A	N/A	(Bent & Hentenryck, 2004)	3	756.89	N/A	N/A
R205	N/A	N/A	(Rousseau et al., 2002)	3	1161.32	N/A	N/A
R206	N/A	N/A	(Schrimpf et al., 2000)	3	943.09	N/A	N/A
R207	N/A	N/A	(Bouthilliera & Crainic, 2005)	3	813.25	N/A	N/A
R208	N/A	N/A	(Mester, 2002)	2.59	777.1	N/A	N/A
R209	N/A	N/A	(Homberger, 2000)	3.17	1036.05	N/A	N/A
R210	N/A	N/A	(Ghoseiri & Farid, 2010)	4	920.3	N/A	N/A
R211	N/A	N/A	(Bent & Hentenryck, 2004)	3	779.73	N/A	N/A
Average				3.31	959.07		
RC101	N/A	N/A	(Taillard et al., 1997)	14.03	1710.52	N/A	N/A
RC102	N/A	N/A	(Taillard et al., 1997)	13	1497.8	N/A	N/A
RC103	N/A	N/A	(Taillard et al., 1997)	11	1290.94	N/A	N/A
RC104	N/A	N/A	(Cordeau et al., 2001)	10	1136.4	N/A	N/A
RC105	N/A	N/A	(Berger et al., 2003)	14	1541.59	N/A	N/A
RC106	N/A	N/A	(Berger et al., 2003)	12	1385.56	N/A	N/A
RC107	N/A	N/A	(Ghoseiri & Farid, 2010)	11	1236.08	N/A	N/A
RC108	N/A	N/A	(Taillard et al., 1997)	10	1151.65	N/A	N/A
Average				11.88	1368.82		
RC201	N/A	N/A	(Mester, 2002)	5	1324.15	N/A	N/A
RC202	N/A	N/A	(Repoussis et al., 2009)	4.1	1255.07	N/A	N/A
RC203	N/A	N/A	(Czech & Czarnas, 2002)	4	947.95	N/A	N/A
RC204	N/A	N/A	(Mester, 2002)	3	798.46	N/A	N/A
RC205	N/A	N/A	(Mester, 2002)	5	1269.78	N/A	N/A
RC206	N/A	N/A	(Homberger, 2000)	4	1105.29	N/A	N/A
RC207	N/A	N/A	(Bent & Hentenryck, 2004)	4	1000.29	N/A	N/A
RC208	N/A	N/A	(Ibaraki et al., 2005)	3	834.16	N/A	N/A
Average				4.01	1066.89		
Total Sum			-	466.22	61873.38		
Total Average				7.6	1016.12		
Count							

TABLE 4.6: Compare the least average with the best-known solutions

(*continued*)

Instance	**Min MTD**						
	Best-known Solution			**NEDPALNS**			
	MNV	**MTD**	**Ref**	**MNV**	**MTD**	**%MNV**	**%MTD**
C101	10	828.94	(Soonchul Jung & Moon, 2002)	**10**	**828.94**	0	0
C102	10	828.94	(Soonchul Jung & Moon, 2002)	**10**	**828.94**	0	0
C103	10	828.06	(Soonchul Jung & Moon, 2002)	**10**	**828.06**	0	0
C104	10	824.96	(Soonchul Jung & Moon, 2002)	**10**	**824.78**	0	-0.02
C105	10	828.94	(Soonchul Jung & Moon, 2002)	**10**	**828.94**	0	0
C106	10	828.94	(Soonchul Jung & Moon, 2002)	**10**	**828.94**	0	0
C107	10	828.94	(Soonchul Jung & Moon, 2002)	**10**	**828.94**	0	0

C108	10	828.94	(Soonchul Jung & Moon, 2002)	**10**	**828.94**	0	0
C109	10	828.94	(Soonchul Jung & Moon, 2002)	**10**	**828.94**	0	0
Average	10	828.4		**10**	**828.38**	0	0
C201	3	591.56	(Soonchul Jung & Moon, 2002)	**3**	**591.56**	0	0
C202	3	591.56	(Soonchul Jung & Moon, 2002)	**3**	**591.56**	0	0
C203	3	591.17	(Soonchul Jung & Moon, 2002)	**3**	**591.17**	0	0
C204	3	591.18	(Soonchul Jung & Moon, 2002)	**3**	**590.6**	0	-0.1
C205	3	588.88	(Soonchul Jung & Moon, 2002)	**3**	**588.88**	0	0
C206	3	588.49	(Soonchul Jung & Moon, 2002)	**3**	**588.49**	0	0
C207	3	588.29	(Soonchul Jung & Moon, 2002)	**3**	**588.29**	0	0
C208	3	588.32	(Soonchul Jung & Moon, 2002)	**3**	**588.32**	0	0
Average	3	589.93		**3**	**589.86**	0	-0.01
R101	20	1643.53	(Soonchul Jung & Moon, 2002)	**20.00**	**1642.88***	0	-0.04
R102	18.5	1479.19	(Soonchul Jung & Moon, 2002)	**18.00**	**1473***	-2.7	-0.42
R103	14.81	1222.29	(Soonchul Jung & Moon, 2002)	**14.00**	**1213.81***	-5.47	-0.69
R104	11.7	1001.44	(Soonchul Jung & Moon, 2002)	**10.93**	**977.05***	-6.57	-2.44
R105	15.91	1371.52	(Soonchul Jung & Moon, 2002)	**15.00**	**1360.78***	-5.72	-0.78
R106	13.59	1252.44	(Soonchul Jung & Moon, 2002)	**13.00**	**1239.37***	-4.34	-1.04
R107	11.73	1083.1	(Soonchul Jung & Moon, 2002)	**11.00**	**1073.04***	-6.22	-0.93
R108	10.74	959.65	(Soonchul Jung & Moon, 2002)	**10.03**	**946.26***	-6.57	-1.4
R109	12.97	1157.27	(Soonchul Jung & Moon, 2002)	13.00	1151.84	0.23	-0.47
R110	12	1082.72	(Soonchul Jung & Moon, 2002)	**12.00**	**1074.81***	0	-0.73
R111	12	1063.21	(Soonchul Jung & Moon, 2002)	**12.00**	**1053.5***	0	-0.91
R112	10.77	971.89	(Soonchul Jung & Moon, 2002)	**10.00**	**962.02***	-7.15	-1.02
Average	13.73	1190.69		**13.25**	**1180.7**	-3.71	-0.91
R201	8.29	1153.04	(Soonchul Jung & Moon, 2002)	**8.00**	**1147.8***	-3.5	-0.45
R202	7.4	1038.4	(Soonchul Jung & Moon, 2002)	**6.00**	**1035.66***	-18.92	-0.26
R203	6	875.87	(Soonchul Jung & Moon, 2002)	**6.00**	**874.87***	0	-0.11
R204	4.46	741.41	(Soonchul Jung & Moon, 2002)	**5.00**	**735.8***	12.11	-0.76
R205	6.05	964.69	(Ombuki et al., 2006)	**5.00**	**954.32***	-17.36	-1.07
R206	5.33	892.55	(Soonchul Jung & Moon, 2002)	**4.97**	**880.73***	-6.84	-1.32
R207	4.66	814.05	(Soonchul Jung & Moon, 2002)	**4.00**	**797.99***	-14.16	-1.97
R208	3.5	714.37	(Soonchul Jung & Moon, 2002)	3.86	706.69	10.34	-1.07
R209	5.26	867.52	(Soonchul Jung & Moon, 2002)	**5.00**	**860.13***	-4.94	-0.85
R210	6.1	918.37	(Soonchul Jung & Moon, 2002)	**6.00**	**905.21***	-1.64	-1.43
R211	4.7	765.64	(Soonchul Jung & Moon, 2002)	**3.97**	**759.78***	-15.63	-0.76
Average	5.61	885.99		**5.25**	**878.09**	-5.5	-0.91
RC101	16.46	1658.34	(Soonchul Jung & Moon, 2002)	**15.00**	**1623.59***	-8.87	-2.1
RC102	14.65	1480.82	(Soonchul Jung & Moon, 2002)	**14.21**	**1466.02***	-3.02	-1
RC103	12.11	1313.73	(Soonchul Jung & Moon, 2002)	**11.17**	**1273.58***	-7.74	-3.06
RC104	10.56	1154.26	(Soonchul Jung & Moon, 2002)	**10.00**	**1136.4***	-5.3	-1.55
RC105	15.96	1540.66	(Soonchul Jung & Moon, 2002)	16.00	1518.58	0.25	-1.43
RC106	13.39	1397.45	(Soonchul Jung & Moon, 2002)	**13.00**	**1376.99***	-2.91	-1.46
RC107	12.03	1227.81	(Soonchul Jung & Moon, 2002)	**12.00**	**1216.78***	-0.25	-0.9
RC108	11	1135.81	(Soonchul Jung & Moon, 2002)	**11.00**	**1121.21***	0	-1.29
Average	13.27	1363.61		**12.80**	**1341.64**	-3.48	-1.6
RC201	9	1269.94	(Soonchul Jung & Moon, 2002)	**9.00**	**1266.15***	0	-0.3
RC202	7.84	1101.03	(Soonchul Jung & Moon, 2002)	**7.93**	**1095.87***	1.16	-0.47
RC203	5.29	943.81	(Soonchul Jung & Moon, 2002)	**5.00**	**926.82***	-5.48	-1.8
RC204	4.05	799.19	(Soonchul Jung & Moon, 2002)	**3.93**	**787.74***	-2.94	-1.43
RC205	7.8	1164.43	(Soonchul Jung & Moon, 2002)	**7.00**	**1157.55***	-10.26	-0.59
RC206	6.39	1067.49	(Soonchul Jung & Moon, 2002)	6.86	1057.54	7.39	-0.93
RC207	6.07	975.24	(Soonchul Jung & Moon, 2002)	**6.00**	**971.05***	-1.15	-0.43
RC208	4.98	791.35	(Soonchul Jung & Moon, 2002)	**4.00**	**779.22***	-19.68	-1.53
Average	6.43	1014.06		**6.22**	**1005.24**	-3.87	-0.94

Total Sum	543.760089.2			**482.86**	**54776.69**	-3.05	-0.82
Total Average	8.89	986.26		**8.62**	**978.16**	-3.04	-0.82
Count			52				

4.5.2 Comparison with other Published Algorithms (Dynamic dataset and Solomon dataset)

In this comparison, the algorithms published used are the adaptive local neighbourhood algorithm (ALNS) (S. Chen et al., 2018), the improved local neighbourhood algorithm (ILNS) (Hong, 2012), and the general variable neighbourhood search algorithm (GVNS) (de Armas & Melián-Batista, 2015). These comparisons compare the least number of vehicles used (VN), the least total travelled distance (TD), and the least rejection ratio (RR). The insertion time is provided for information purposes and not for comparison with other published algorithms since different hardware is used for recording the results.

Table 4.7 shows the comparison with other published algorithms. In this table, the columns from the left to right represent the instance type, degree of vehicles dynamism, the number of used vehicles (VN), the total travelled distance (TD), the insertion time (IT), and the rejection ratio (RR). To obtain the result, each instance belonging to the customer size and specific degree of dynamism is executed 10 times and the best result is obtained. These steps are repeated with the other instances for that customer size and specific DoD. The "Average <<DoD type>> based on all instance types" is the average result for all instance types for that specific DoD. The "Average <<instance type>>" based on all DoD is the average result for all DoD of that specific type of instance. The “Overall Average” is the average result of all “Average <<instance type>>” based on all DoD. The bold font indicates the best result.

In the average C1 instance type based on all DoDs, NEDPALNS (NV - 10.89, TD - 964.90, and RR - 0.02) has achieved a non-dominated solution compared to ALNS (VN - 10.44, TD - 1077.80 and RR - 0.11), ILNS (VN - 10.71, TD - 986.10 and RR - 0.24) and GVNS (VN - 11.02, TD - 962.80, RR - 0.00). This result indicates that the NEDPALNS result is competitive with other published algorithms in instance type C1.

In the average C2 instance type based on all DoDs, NEDPALNS has a non-dominated solution in the least average of best NV (3.13), best TD (618.01), and best RR (0) in 10% DoD and best NV (3.38), best TD (607.04), and best RR (0.5) in 50% DoD compared to ILNS in 10% DoD (best NV - 3, best TD - 594.67 and best RR - 0) and 50% DoD (best NV - 3.13, best TD - 604.98 and best RR - 0). However, NEDPALNS (best NV - 3.20, best TD - 614.83 and best RR - 0.35) in the C2 instance type has an overall average result best VN and best TD than ALNS (best NV - 3.35, best TD - 650.79 and best RR - 0.00), ILNS (best NV - 3.23, best TD - 624.87 and best RR - 0.00) and GVNS (best NV - 3.28, best TD - 641.16 and best RR - 0.00). This shows that NEDPALNS has the best NV and the best TD on the least average in the C2 instance type compared to other published algorithms.

NEDPALNS has the least average of best RR in R1 (0.04), R2 (0.00), RC1 (0.00) and RC2 (0.00) compared to ALNS (R1 – 2.20, R2 – 0.00, RC1 –

1.15 and RC2 – 0.00), ILNS (R1 – 1.10, R2 – 0.05, RC1 – 1.50 and RC2 – 0.10) and GVNS (R1 – 2.18, R2 – 0.00, RC1 – 1.45 and RC2 – 0.00). In general, NEDPALNS (0.07) still has the least overall average of the best RR compared to other published algorithms (ALNS – 0.58, ILNS – 0.50, and GVNS – 0.61). This shows that overall, NEDPALNS can accommodate more customer requests compared to other published algorithms on all DoDs.

If we compare the average for a specific DoD in all the instances, NEDPALNS consistently achieve the least average of best RR in 10% DoD (0.00), 30% DoD (0.06), 50% DoD (0.08), 70% DoD (0.16) and 90% DoD – 0.04) compared to ALNS (10% DoD – 0.16, 30% DoD – 0.37, 50% DoD – 0.61, 70% DoD – 0.78 and 90% DoD – 0.96) , ILNS (10% DoD – 0.27, 30% DoD – 0.38, 50% DoD – 0.40, 70% DoD – 0.66 and 90% DoD – 0.80) and GVNS (10% DoD – 0.17, 30% DoD – 0.43, 50% DoD – 0.61, 70% DoD – 0.87 and 90% DoD – 0.95). This means that NEDPALNS can still achieve the least average of best RR based on specific DoDs for all instance types.

Nevertheless, NEDPALNS has the overall least average of best RR (0.07) compared to other published algorithms (ALNS – 0.58, ILNS – 0.50, and GVNS – 0.61).

. **Table 4.7: Comparison with other published algorithms on the least average of best VN, best TD, and best RR**

		NEDPALNS				**ALNS** (Chen *et al.*, 2018)				**ILNS (Hong, 2012)**				**GVNS** (de Armas and Melián-Batista, 2015)			
Instance Type	**Degree of Dynamic**	**VN**	**TD**	**IT**	**RR**	**VN**	**TD**	**IT**	**RR**	**VN**	**TD**	**IT**	**RR**	**VN**	**TD**	**IT**	**RR**
	90	10.89	995.78	18.56	**0**	**10.44**	974.41	9.96	0.11	10.78	1039.77	6.6	0.22	10.67	**963.33**	7.81	**0**
	70	11	**995.21**	19.56	0.11	**10.33**	1088.38	12.45	0.11	10.78	1031.68	10.79	0.22	11.33	1009.47	7.67	**0**
C1	50	11	**979.88**	20.22	**0**	**10.44**	1096.58	17.37	0.11	10.89	1001.18	19.01	0.22	11	992.97	6.22	**0**
	30	10.89	**948.63**	21.78	**0**	**10.56**	1126.84	23.62	0.11	**10.56**	962.08	28.03	0.33	11.56	949.95	9.13	**0**
	10	10.67	905.01	28.89	**0**	**10.44**	1102.8	30.89	0.11	10.56	**895.77**	15.4	0.22	10.56	898.3	13.74	**0**
	Average	10.89	964.90	21.80	0.02	**10.44**	1077.80	18.86	0.11	10.71	986.10	15.97	0.24	11.02	**962.80**	8.91	**0.00**
	90	**3.25**	**626.91**	19.88	0.13	3.38	668.45	11.86	**0**	3.25	636.79	6.12	**0**	3.38	668.99	16.67	**0**
	70	**3.13**	**616.56**	20.38	0.75	3.25	652.63	13.32	**0**	**3.13**	636.47	10.01	**0**	3.38	672.95	14.03	**0**
C2	50	3.38	607.04	20.88	0.5	3.25	650.7	18.49	**0**	**3.13**	**604.98**	16.8	**0**	**3.13**	623.1	20.25	**0**
	30	**3.13**	**605.65**	22.88	0.38	3.63	658.09	24.22	**0**	3.63	651.42	29.87	**0**	3.25	624.81	34.82	**0**
	10	3.13	618.01	41.88	**0**	3.25	624.06	40.25	**0**	**3**	**594.67**	59.7	**0**	3.25	615.93	80.78	**0**
	Average	**3.20**	**614.83**	25.18	0.35	3.35	650.79	21.63	**0.00**	3.23	624.87	24.50	**0.00**	3.28	641.16	33.31	**0.00**
	90	15.25	1422.29	23.67	**0.08**	13.67	**1270.2**	8.61	3.92	14.25	1335.94	17.43	2.33	14.67	**1250.38**	14.5	3.83
	70	14.2	1350.94	22.6	**0.1**	13.42	1298.86	11.31	2.83	14.33	1331.34	21.73	1.75	14.75	**1267.78**	10.95	3.08
R1	50	14.92	1390.38	22.42	**0**	**13.58**	1313.35	15.22	2.17	14.08	1295.81	28.27	0.67	14.58	**1267.47**	11.84	1.92
	30	15.08	1385.09	23.83	**0**	**13.33**	1310.23	21.99	1.5	13.92	1286.63	46.59	0.58	14.25	**1256.04**	15.7	1.58
	10	15.17	1394.08	33.42	**0**	**13.33**	1309.1	34.19	0.58	13.5	1257.08	67.99	0.17	14.17	**1250.16**	15.29	0.5
	Average	14.92	1388.56	25.19	**0.04**	**13.47**	1300.35	18.26	2.20	14.02	1301.36	36.40	1.10	14.48	**1258.37**	13.66	2.18
	90	4.27	1075.17	16.64	**0**	**3.45**	1076.77	21.82	**0**	3.55	**1047.82**	13.2	0.09	4	1086.78	16.47	**0**
	70	4.27	1048.81	17.73	**0**	**3.36**	1088.97	29.25	**0**	3.64	**1032.04**	20.15	0.09	4.36	1078.03	12.74	**0**
R2	50	4.09	1035.87	18.45	**0**	**3.45**	1086.09	39.3	**0**	3.82	**1016.52**	30.03	0	4.55	1071.83	11.96	**0**
	30	4.55	1017.73	21	**0**	**3.45**	1087.52	59.93	**0**	4.91	**985.59**	57.07	**0**	4.73	1035.6	10.18	**0**
	10	**4.27**	1004.02	35.45	**0**	5.09	1080.5	75.21	**0**	6.36	**950**	68.58	0.09	5.27	1000	9.48	**0**
	Average	4.29	1036.32	21.85	**0.00**	**3.76**	1083.97	45.10	0.00	4.46	**1006.39**	37.81	0.05	4.58	1054.45	12.17	**0.00**

RC1	90	14.88	1600.39	22.75	**0**	**13.63**	1501.76	5.04	1.75	14	1513.94	17.31	2	14.63	**1470.45**	15.39	1.88
	70	14.75	1591.19	22.13	**0**	**13.13**	1510.21	6.65	1.75	13.88	1511.29	25.32	1.88	14.88	**1489.28**	13.43	2.13
	50	15	1575.78	22	**0**	**13.63**	1520.47	8.92	1.38	**13.63**	1514.72	48.78	1.38	14.5	**1484.01**	13.72	1.75
	30	15.13	1622.91	22.88	**0**	**13**	1484.89	12.71	0.63	13.88	1492.22	45.26	1.13	14.38	**1471**	16.51	1
	10	15.38	1617.36	32.38	**0**	**12.88**	1473.69	16.94	0.25	13.38	1436.23	83.52	1.13	13.5	**1417.07**	23.01	0.5
	Average	15.03	1601.53	24.43	**0.00**	**13.25**	1498.20	10.05	1.15	13.75	1493.68	44.04	1.50	14.38	**1466.36**	16.41	1.45
RC2	90	5	1270.18	17.38	**0**	**3.88**	1264.94	11.31	**0**	4	**1257.19**	11.34	0.13	4.63	1275.93	28.05	**0**
	70	5.25	**1183.63**	18.13	**0**	**3.88**	1261.81	14.74	**0**	**3.88**	1239.46	19.26	0	5.13	1234.36	16.07	**0**
	50	5.13	**1165.78**	19.38	**0**	**3.88**	1260.66	20.67	**0**	4.25	1190.54	27.84	0.13	5.88	1200.26	11.46	**0**
	30	5.38	1172.01	21.75	**0**	**3.88**	1238.82	30.76	**0**	5.38	**1166.04**	41.51	0.25	5.88	1172.33	11.68	**0**
	10	**5.38**	1165.31	34.63	**0**	5.75	1253.11	42.89	**0**	6.75	**1103.3**	55.55	0	6.13	1153.43	13.27	**0**
	Average	5.23	1191.38	22.25	**0.00**	**4.25**	1255.87	24.07	**0.00**	4.85	**1191.31**	31.10	0.10	5.53	1207.26	16.11	**0.00**
Average (10% DoD in C1, C2, R1, R2, RC1 and RC2 instance type)		9.00	1117.30	34.44	**0.00**	**8.46**	1140.54	40.06	0.16	8.93	**1039.51**	58.46	0.27	8.81	1055.82	25.93	0.17
Average (30% DoD in C1, C2, R1, R2, RC1 and RC2 instance type)		9.03	1125.34	22.35	**0.06**	**7.98**	1151.07	28.87	0.37	8.71	1090.66	41.39	0.38	9.01	**1084.96**	16.34	0.43
Average (50% DoD in C1, C2, R1, R2, RC1 and RC2 instance type)		8.92	1125.79	20.56	**0.08**	**8.04**	1154.64	20.00	0.61	8.30	**1103.96**	28.46	0.40	8.94	1106.61	12.58	0.61
Average (70% DoD in C1, C2, R1, R2, RC1 and RC2 instance type)		8.77	1131.06	20.09	**0.16**	**7.90**	1150.14	14.62	0.78	8.27	1130.38	17.88	0.66	8.97	**1125.31**	12.48	0.87
Average (10% DoD in C1, C2, R1, R2, RC1 and RC2 instance type)		8.92	1165.12	19.81	**0.04**	**8.08**	1126.09	11.43	0.96	8.31	1138.58	12.00	0.80	8.66	**1119.31**	16.48	0.95
Average (C1 instance type		10.89	964.9	21.8	0.02	**10.44**	1077.8	18.86	0.11	10.71	986.1	15.97	0.24	11.02	**962.8**	8.91	**0**

in 10%, 30%, 50% 70% and 90% DoD)																
Average (C2 customers in 10%, 30%, 50% 70% and 90% DoD)	**3.2**	**614.83**	25.18	0.35	3.35	650.79	21.63	**0**	3.23	624.87	24.5	**0**	3.28	641.16	33.31	**0**
Average (R1 customers in 10%, 30%, 50% 70% and 90% DoD)	14.92	1388.56	25.19	**0.04**	**13.47**	1300.35	18.26	2.2	14.02	1301.36	36.4	1.1	14.48	**1258.37**	13.66	2.18
Average (R2 customers in 10%, 30%, 50% 70% and 90% DoD)	4.29	1036.32	21.85	**0**	**3.76**	1083.97	45.1	0	4.46	**1006.39**	37.81	0.05	4.58	1054.45	12.17	**0**
Average (RC1 customers in 10%, 30%, 50% 70% and 90% DoD)	15.03	1601.53	24.43	**0**	**13.25**	1498.2	10.05	1.15	13.75	1493.68	44.04	1.5	14.38	**1466.36**	16.41	1.45
Average (RC2 customers in 10%, 30%, 50% 70% and 90% DoD)	5.23	1191.38	22.25	**0**	**4.25**	1255.87	24.07	**0**	4.85	**1191.31**	31.1	0.1	5.53	1207.26	16.11	**0**
Overall Average	8.93	1132.92	23.45	**0.07**	**8.09**	1144.50	23.00	0.58	8.50	1100.62	31.64	0.50	8.88	**1098.40**	16.76	0.61

Table 4.8 shows the best, worst, and average the least average rejection rates, the least average number of used vehicles, and the least average total travelled distance against ALNS.

In the type of C1 instance, NEDPALNS has the least average rejection rates (best - 0.02, and average - 0.02) and total travelled distance (best - 963.2, worst - 1212.98, average - 1064.98) than ALNS's least average rejection rates (best - 0.11, and average - 0.11) and total travelled distance (best - 1077.8, worst - 1348.22, average - 1227.96). This result shows that NEDPALNS has the least average results in RR and TD than ALNS in the narrow type of clustered customer distribution.

However, for instance, type C2, NEDPALNS (best - 789.02, worst - 1056.7 and average - 882.82) has the least average total travelled distance than ALNS (best - 864.29, worst - 1083.85 and average - 987.73). This shows that NEDPALNS has a better result in the least total travelled distance than ALNS in the wider type of clustered customer distribution.

In R1 and RC1 instance types. NEDPALNS (R1 best - 0.05, R1 worst - 0.23 and R1 average - 0.05, RC1 best - 0, RC1 worst - 0.37, and RC1 average RR – 0) has the least average rejection rates than ALNS (R1 best - 2.2, R1 worst - 1.91, R1 average - 2.05, RC1 best - 1.15, RC1 worst - 0.98 and RC1 average - 1.03). This result indicates NEDPALNS has the least average rejection rates than ALNS in the narrow type of random customer distribution and narrow type of random and clustered customer distribution regardless of DoD.

In the R2 instance type, NEDPALNS has the least average total travelled distance (best - 999.36 and worst - 1090.22) than ALNS (best– 1028.23 and worst - 118.74). However, in the RC2 instance type, NEDPALNS underperforms ALNS in the least rejection rates, least number of used vehicles, and least total travelled distance. This means NEDPALNS is susceptible to a wider type of random and clustered customer distribution. Nevertheless, on overall average, NEDPALNS (best– 0.06, worst– 0.25, and average– 0.06) still has the least rejection rates than ALNS (best– 0.68, worst - 0.6, and average– 0.63).

If we compare all instance types on their specific DoD, NEDPALNS has the least average rejection rates in 10% DoD (best– 0, worst– 0.04 and average – 0.00), 30% DoD (best– 0.06, worst– 0.19, and average – 0.06), 50% DoD (best– 0.08, worst– 0.30 and average – 0.08), 70% DoD (best– 0.16, worst– 0.33, and average – 0.16) and 90% DoD (best– 0.05, worst– 0.48, and average – 0.05) than ALNS in 10% DoD (best– 0.16, worst– 0.16 and average– 0.16), 30% DoD (best - 0.37, worst - 0.35, and average - 0.38), 50% DoD (best - 0.61, worst - 0.49, and average - 0.51), 70% DoD (best - 0.78, worst - 0.64, and average - 0.74) and 90% (best - 0.96, worst - 0.87, and average - 0.87) respectively.

Table 4.8: Comparison with ALNS on the least average of best, the least average of worst, and the least average of average (VN, TD, and RR)

Instance	Degree of Dynamic	Average rejection rates (NEDPALNS)			Average rejection rates (ALNS) (S. Chen et al., 2018)			Average number of used vehicles (NEDPALNS)			Average number of used vehicles (ALNS) (S. Chen et al., 2018)			Average total travelled Distance (NEDPALNS)			Average total travelled distance (ALNS) (S. Chen et al., 2018)		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Avg.	Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
	90	**0**	0.37	**0**	0.11	**0.11**	0.11	10.89	12.22	11.44	**10.44**	**11.22**	**10.56**	995.78	1213.9	**1099.01**	**974.41**	**1182.01**	1105.24
	70	**0.11**	**0.11**	**0.11**	**0.11**	**0.11**	**0.11**	11	12.22	11.46	**10.33**	**11.56**	**10.76**	**986.69**	**1224.97**	**1087.79**	1088.38	1322.81	1207.47
C1	50	**0**	**0.1**	**0**	0.11	0.11	0.11	11	12.44	11.53	**10.44**	**11.56**	**10.82**	**979.88**	**1206.54**	**1075.67**	1096.58	1390.96	1262.96
	30	**0**	**0.02**	**0**	0.11	0.11	0.11	10.89	12.22	11.36	**10.56**	**11.44**	**10.99**	**948.63**	**1240.2**	**1048.04**	1126.84	1436.46	1294.45
	10	**0**	**0.01**	**0**	0.11	0.11	0.11	10.67	12.11	11.29	**10.44**	**11.67**	**11.12**	**905.01**	**1179.27**	**1014.4**	1102.8	1408.86	1269.66
Average		**0.02**	0.12	**0.02**	0.11	0.11	0.11	10.89	12.24	11.42	**10.44**	**11.49**	**10.85**	**963.2**	**1212.98**	**1064.98**	1077.8	1348.22	1227.96
	90	0.13	1.41	0.13	**0**	**0**	**0**	**3.25**	**4.25**	**3.71**	3.38	4.63	3.79	**626.91**	953.44	748.48	668.45	**815.11**	**742.73**
	70	0.75	1.15	0.75	**0**	**0**	**0**	**3.13**	**4.13**	**3.51**	3.25	4.5	3.7	**616.56**	884.51	**696.19**	652.63	**837.39**	745.04
C2	50	0.5	0.78	0.5	**0**	**0**	**0**	3.38	**4.13**	**3.59**	**3.25**	4.75	3.76	**607.04**	820.43	**654.2**	650.7	**809.99**	761.64
	30	0.38	0.89	0.38	**0**	**0**	**0**	**3.13**	**4.25**	**3.51**	3.63	4.38	3.69	**605.65**	842.68	681.56	658.09	834.73	**741.69**
	10	**0**	0.1	**0**	**0**	**0**	**0**	**3.13**	4	3.54	3.25	**3.75**	3.38	**618.01**	1001.04	**722.85**	624.06	800.13	746.43
Average		0.19	0.49	0.19	**0.06**	**0.06**	**0.06**	7.04	8.2	7.5	**6.9**	**7.95**	**7.26**	**789.02**	**1056.7**	**882.82**	864.29	1083.85	987.73
	90	**0.17**	**0.55**	**0.17**	3.92	3.58	3.65	**14.83**	16.67	15.8	**13.67**	**15.58**	14.41	1394.91	1523.86	1446.27	**1270.2**	**1387.89**	**1333.11**
	70	**0.1**	**0.27**	**0.1**	2.83	2.33	2.79	14.2	16.3	15.23	**13.42**	**14.83**	**13.95**	1350.94	1505.17	1422.41	**1298.86**	**1386.23**	**1345.85**
R1	50	**0**	**0.21**	**0**	2.17	1.83	1.88	14.92	17	15.75	**13.58**	**14.83**	**14.04**	1390.38	1538.54	1452.01	**1313.35**	**1421.24**	**1383.66**
	30	**0**	**0.06**	**0**	1.5	1.25	1.4	15.08	17.17	15.89	**13.33**	**14.58**	**13.56**	1385.09	1529.58	1451.25	**1310.23**	**1413.81**	**1366.6**
	10	**0**	**0.06**	**0**	0.58	0.58	0.54	15.17	17.5	16.38	**13.33**	**14.33**	**13.78**	1394.08	1543.52	1465.08	**1309.1**	**1418.95**	**1369.74**
Average		**0.05**	**0.23**	**0.05**	2.2	1.91	2.05	14.84	16.93	15.81	**13.47**	**14.83**	**13.95**	1383.08	1528.13	1447.4	**1300.35**	**1405.62**	**1359.79**
R2	90	**0**	**0**	**0**	**0**	**0**	**0**	4.27	5.45	4.66	**3.45**	**4.64**	**3.67**	**1075.17**	1298.21	1164.19	1076.77	**1145.14**	**1133.29**

	70	**0**	**0**	**0**	**0**	**0**	**0**	4.27	5.55	4.84	**3.36**	**4.36**	**3.55**	**1048.81**	1305.56	1162.15	1088.97	**1155.53**	**1138.44**
	50	**0**	0.09	**0**	**0**	**0**	**0**	4.09	5.36	4.76	**3.45**	**4.73**	**3.69**	**1035.87**	1304.39	1150.88	1086.09	**1201.51**	**1146.93**
	30	**0**	**0**	**0**	**0**	**0**	**0**	4.55	5.55	4.98	**3.45**	**4**	**3.35**	**1017.73**	1308.96	**1132.54**	1087.52	**1182.04**	1142.1
	10	**0**	**0**	**0**	**0**	**0**	**0**	**4.27**	5.64	**4.9**	5.09	**5.5**	5.31	**1004.02**	1314.36	**1129.51**	1080.5	**1193.41**	1137.68
Average		**0**	0.01	**0**	**0**	**0**	**0**	8.3	9.71	8.91	**7.76**	**8.84**	**8.09**	**999.36**	1236.96	**1090.22**	1028.23	**1187.21**	1118.74
	90	**0**	**0.56**	**0**	1.75	1.5	1.48	15	16.88	15.81	**13.63**	**15.25**	**14.38**	1604.29	1788.05	1689.05	**1501.76**	**1651.64**	**1578.27**
	70	**0**	**0.44**	**0**	1.75	1.38	1.53	14.75	16.38	15.63	**13.13**	**14.63**	**14.1**	1591.19	1777.13	1673.54	**1510.21**	**1643.41**	**1585.8**
RC1	50	**0**	**0.64**	**0**	1.38	1	1.04	15	16.88	15.73	**13.63**	**14.75**	**13.89**	1575.78	1805.74	1675.96	**1520.47**	**1650.57**	**1586.02**
	30	**0**	**0.15**	**0**	0.63	0.75	0.78	15.13	17.13	16.06	**13**	**14.25**	**13.56**	1622.91	1826.36	1714.22	**1484.89**	**1620.66**	**1564.33**
	10	**0**	**0.06**	**0**	0.25	0.25	0.31	15.38	17.13	16.18	**12.88**	**14**	**13.28**	1617.36	1815.3	1699.38	**1473.69**	**1610.58**	**1539.29**
Average		**0**	**0.37**	**0**	1.15	0.98	1.03	15.05	16.88	15.88	**13.25**	**14.58**	**13.84**	1602.31	1802.51	1690.43	**1498.2**	**1635.37**	**1570.74**
	90	**0**	**0**	**0**	**0**	**0**	**0**	5	6.25	5.56	**3.88**	**4.63**	**3.96**	1270.17	1568.79	1414.31	**1264.94**	**1353.99**	**1327.66**
	70	**0**	**0**	**0**	**0**	**0**	**0**	5.25	6.5	5.73	**3.88**	**5**	**4**	**1183.62**	1539.01	**1334.11**	1261.81	**1400.27**	1339.28
RC2	50	**0**	**0**	**0**	**0**	**0**	**0**	5.13	6.5	5.88	**3.88**	**5**	**4.08**	1165.78	1466.81	**1323.53**	**1260.66**	**1387.33**	1347.17
	30	**0**	**0**	**0**	**0**	**0**	**0**	5.38	6.63	5.96	**3.88**	**4.63**	**3.96**	**1172.01**	1503.48	**1318.3**	1238.82	**1372.79**	1342.04
	10	**0**	**0**	**0**	**0**	**0**	**0**	**5.38**	6.63	**5.85**	5.75	**6.5**	5.86	**1165.31**	1465.5	**1294.71**	1253.11	**1355.36**	1328.66
Average		**0**	**0**	**0**	**0**	**0**	**0**	5.23	6.5	5.8	**4.25**	**5.15**	**4.37**	**1191.38**	1508.72	1336.99	1255.87	**1373.95**	**1336.96**
Average (10% DoD on C1,C2,R1,R2,RC1 and RC2 instance type		**0**	**0.04**	**0.00**	0.16	0.16	0.16	9.00	10.50	9.69	8.46	9.29	8.79	1117.30	1386.50	**1220.99**	1140.54	1297.88	1231.91
Average (30% DoD on C1,C2,R1,R2,RC1 and RC2 instance type		**0.06**	**0.19**	**0.06**	0.37	0.35	0.38	9.03	10.49	9.63	7.98	8.88	8.19	1125.34	1375.21	**1224.32**	1151.07	1310.08	1241.87
Average (50% DoD on C1,C2,R1,R2,RC1 and RC2 instance type		**0.08**	**0.30**	**0.08**	0.61	0.49	0.51	8.92	10.39	9.54	8.04	9.27	8.38	1125.79	1357.08	**1222.04**	1154.64	1310.27	1248.06
Average (70% DoD on C1,C2,R1,R2,RC1 and RC2 instance type		**0.16**	**0.33**	**0.16**	0.78	0.64	0.74	8.77	10.18	9.40	7.90	9.15	8.34	1129.64	1372.73	1229.37	1150.14	1290.94	1226.98
Average (90% DoD		**0.05**	**0.48**	**0.05**	0.96	0.87	0.87	8.87	10.29	9.50	8.08	9.33	8.46	1161.21	1391.04	1260.22	1126.09	1255.96	1203.38

on C1,C2,R1,R2,RC1 and RC2 instance type)																		
Average (C1 instance type on 10%, 30%, 50% 70% and 90% DoD)	**0.02**	0.12	**0.02**	0.11	0.11	0.11	**10.89**	12.24	11.42	**10.44**	11.49	**10.85**	963.2	**1212.98**	**1064.98**	**1077.8**	1348.22	1227.96
Average (C2 instance type on 10%, 30%, 50% 70% and 90% DoD)	0.19	0.49	0.19	**0.06**	**0.06**	**0.06**	7.04	8.2	7.5	**6.9**	**7.95**	**7.26**	**789.02**	**1056.7**	**882.82**	864.29	1083.85	987.73
Average (R1 instance type on 10%, 30%, 50% 70% and 90% DoD)	**0.05**	**0.23**	**0.05**	2.2	1.91	2.05	14.84	16.93	15.81	**13.47**	**14.83**	**13.95**	1383.08	1528.13	1447.4	**1300.35**	**1405.62**	**1359.79**
Average (R2 instance type on 10%, 30%, 50% 70% and 90% DoD)	**0**	**0.01**	**0**	**0**	**0**	**0**	8.3	9.71	8.91	**7.76**	8.84	**8.09**	**999.36**	1236.96	**1090.22**	1028.23	**1187.21**	1118.74
Average (RC1 instance type on 10%, 30%, 50% 70% and 90% DoD)	**0**	**0.37**	**0**	1.15	0.98	1.03	15.05	16.88	15.88	**13.25**	**14.58**	**13.84**	1602.31	1802.51	1690.43	**1498.2**	**1635.37**	**1570.74**
Average (RC2 instance type on 10%, 30%, 50% 70% and 90% DoD)	**0**	**0**	**0**	**0**	**0**	**0**	5.23	6.5	5.8	**4.25**	**5.15**	**4.37**	**1191.38**	1508.72	1336.99	1255.87	**1373.95**	**1336.96**
Overall Average	**0.04**	**0.20**	**0.04**	0.59	0.51	0.54	10.23	11.74	10.89	**9.35**	**10.47**	**9.73**	**1154.73**	1391.00	**1252.14**	1170.79	**1339.04**	1266.99

Overall, NEDPALNS has the best (0.04), worst (0.20) and average (0.04) on the least average rejection rates and the best (1154.73) and average (1252.14) on the least average total travelled distance than ALNS's least average rejection rates (best – 0.59, worst – 0.51 and average – 0.54) and the least average of total travelled distance (best – 1170.79 and average - 1266.99).

Appendix A7 – A12 show a comparison between NEDPALNS and ALNS hypervolume and the number of non-dominated solutions based on all instance types on 10% DoD. NEDPALNS has better hypervolume (C1 – 0.4530, C2 – 0.4635, R1 – 0.4710, R1 – 0.4170, R2 – 0.5563, RC1 – 0.4097 and RC2 – 0.5369) and number of dominated solutions (C1 – 11, C2 – 11, R1 – 19, R2 – 18, RC1 – 10 and RC2 – 12) than ALNS hypervolume (C1 – 0.4029, C2 - 0.4080, R1 – 0.3991, R2 – 0.4810, RC1 – 0.4067 and RC2 – 0.4748) and number of dominated solutions (C1 – 11, C2 - 8, R1 –15, R2 – 15, RC1 – 9 and RC2 – 10).

In general, in all instance types using 10% DoD, NEDPALNS has a better overall count (81) and an overall hypervolume average (0.4727) than ALNS's overall count (68) and an overall hypervolume average (0.4288). This indicates NEDPALNS has better convergence and diversity in all instances types using 10% DoD than ALNS.

Appendix A13-A18 show a comparison between NEDPALNS and ALNS hypervolume based on all instance types using 30% DoD. NEDPALNS has better hypervolume (C1 – 0.4176, C2 – 0.4427, R1 – 0.3817, R1 – 0.4464, R2 – 0.5051, RC1 – 0.4376 and RC2 – 0.4959) than ALNS (C1 – 0.4065, C2 - 0.3817, R1 – 0.4160, R2 – 0.5090, RC1 – 0.4195 and RC2 – 0.4937).

Overall, in all instance types using 30% DoD, NEDPALNS has an overall hypervolume average (0.4576) than ALNS (0.4377). This indicates NEDPALNS has better convergence and diversity in all instance types using 30% DoD.

In all the instance types using 50% DoD as shown in Appendix A19-A24, NEDPALNS hypervolume in C1(0.4565), R1(0.4451), R2(0.6215), RC1(0.4451), and RC2(0.5745) outperform ALNS hypervolume in C1(0.4357), R1(0.4376), R2(0.5393), RC1(0.4129) and RC2(0.5104). Overall average, NEDPALNS hypervolume (0.4948) and the number of non-dominated solutions (90) outperform the overall average in ALNS hypervolume (0.4642) and the number of non-dominated solutions (85). This shows that NEDPALNS has better convergence and diversity in all instance types using 50% DoD.

Appendix A25-A30 shows the 70% DoD benchmark. NEDPALNS hypervolume in instance type C1 (0.4061), C2 (0.4670), R1 (0.4500), R2 (0.5398), RC1 (0.4715), and RC2 (0.5802) has better results than ALNS hypervolume in instance type C1 (0.3948), C2 (0.4229), R1 (0.4200), R2 (0.4963), RC1 (0.4153) and RC2 (0.5198). The overall average in NEDPALNS hypervolume (0.4858) and the number of non-dominated solutions (89) outperform the overall average in ALNS's hypervolume (0.4449) and the number of non-dominated solutions (81). This shows that NEDPALNS has better convergence and diversity based on all instance types using 70% DoD.

In 90% DoD as shown in Appendix A31-A36, the hypervolume of NEDPALNS in instance type C1 (0.5060), C2 (0.4952), R1 (0.4213), R2 (0.4570) and RC1 (0.4578), as well as in the number of non-dominated solutions in instance type C1 (16), C2 (14), R1 (25) and RC2 (13) outperform the hypervolume of ALNS in instance type C1(0.4379), C2 (0.4881), R1(0.4052), R2(0.4341) and RC1(0.4234) and the number of non-dominated solutions in C1 (13)), C2 (11) and R1(18) and RC2(12). The overall average based on all instance types using 90% DoD show NEDPALNS (0.4725) and the number of non-dominated solutions (97) is better than the overall average in ALNS's hypervolume (0.4534) and the number of non-dominated solutions (87). This shows that NEDPALNS has better convergence and diversity based on all instance types using 90% DoD.

In summary, NEDPALNS has better convergence and diversity than ALNS in all instance types (C1, C2, R1, R2, RC1, and RC2) and all DoDs (10%, 30%, 50%, 70%, and 90%).

4.5.3 Comparison with ALNS algorithm (MOVRPTW and Dynamic dataset)

In this comparison, we evaluate NEDPALNS performances against ALNS using MOVRPTW and a dynamic dataset that is used for solving MODVRPTW. ALNS is compared based on 50, 150, and 250 customers. Each customer size is run against each type of DoD. In this comparison, we use 5 categories of DoD which are 10%, 30%, 50%, 70%, and 90%. Each of the instances is executed 10 times. The VN, TD, IT, and RR have a similar definition in Section 4.2.2 and, therefore, it requires no further introduction. In Table 4.9, these comparisons are based on the number of vehicles used, the total travelled distance, and the rejection ratio.

Compared to 50 customers, NEDPALNS has the least average in the best NV, the best TD, and the best RR in 30% DoD (NV – 8.2, TD - 1097.3 and RR - 0.6) and 50% DoD (NV – 8, TD - 1205.7 and RR - 0.6) than ALNS in 30% DoD (NV – 8.47, TD – 1080.24, RR – 0.2) and 50% DoD (NV – 8, TD – 1222.15, RR – 0.67). This shows that NEDPALNS has the least average of best NV, best TD, and best RR compared to ALNS.

In 10% DoD, NEDPALNS has the least average of the best VN (8.4) and the least average of the best RR (0.07) than ALNS (VN - 8.47 and RR - 0.2). In 70% DoD, NEDPALNS has the least average of the best VN (7.6) and the least average of the best TD (1178.65) than ALNS (best VN – 7.67 and best

TD – 1243.09). Except for 90% DoD where NEDPALNS only has the least average of best TD (1260.87) than ALNS (best TD – 1284.42).

On average, NEDPALNS has the least average of best NV - 7.99, best TD - 1166.924, and best RR -1.44 than ALNS (best NV - 7.99, best TD - 1189.79, and best RR -1.44). This shows that on average, NEDPALNS outperforms ALNS with the least average of best VN, best TD, and best RR in the 50 customers comparison.

In the 150 customers comparison, NEDPALNS has least average results in 10% DoD (best NV- 13.47, best TD - 1881.28 and best RR - 0.6), 30% DoD (best NV – 13, best TD – 2062 and best RR - 2.8) and 90% DoD (best NV - 14.13, best TD - 2497.33 and best RR – 5.53) than ALNS in 10% DoD (best NV - 13.47, best TD - 1888.82 and best RR - 0.67), 30% DoD (best NV - 13.07, best TD - 2029.92 and best RR - 2.6) and 90% DoD (best NV - 14.13, best TD - 2586.38 and best RR – 5.73) respectively. These results show that NEDPALNS has the least average result of the best VN, the best TD, and the best RR in 10% DoD, 30% DoD, and 90% DoD.

Table 4.9: Comparison with ALNS the least average of best VN, best TD, and best RR

Customer	Degree of	Proposed Algorithm				ALNS (2018)			
Size	Dynamic	VN	TD	IT	RR	VN	TD	IT	RR
50	90	7.73	**1260.87**	21.47	3.13	**7.47**	1284.42	20.4	**3**
	70	**7.6**	**1178.65**	19.53	2.8	7.67	1243.09	18.93	**2.67**
	50	**8**	**1205.73**	19.13	**0.6**	**8**	1222.15	18.27	0.67
	30	**8.2**	**1097.3**	18.53	**0.6**	8.33	1119.07	18.87	0.67
	10	**8.4**	1092.07	25.67	**0.07**	8.47	**1080.24**	22.33	0.2
	Average	**7.99**	**1166.924**	20.866	**1.44**	7.99	1189.794	19.76	1.44
150	90	**14.13**	**2497.33**	38.07	**5.53**	14.13	2586.38	29.6	5.73
	70	**12.73**	**2356.25**	34.2	4.93	13.07	2405.95	28.73	4.73
	50	12.8	**2207.25**	27.2	3.13	12.73	2261.93	29.6	2.93
	30	**13**	**2062**	40.4	**2.8**	13.07	2029.92	33.73	2.6
	10	**13.47**	**1881.28**	82.4	**0.6**	13.47	1888.82	66	0.67
	Average	**13.23**	**2200.82**	44.45	3.4	13.29	2234.6	37.53	**3.33**
250	90	**20.33**	**3505.72**	58.47	8.47	20.67	3542.19	51	**6.4**
	70	**18.8**	**3246.42**	54.6	7.47	19.07	3408.25	47.93	**6.53**
	50	**17.93**	**3127.12**	54.87	7.73	18	3171.79	49.87	**7**
	30	**18.73**	**2884.07**	59.87	3.67	19.07	3044.41	65.47	**3.27**
	10	**19.8**	**2809.05**	134.4	**0.8**	19.53	2742.8	192.93	1
	Average	**19.12**	**3114.48**	72.44	5.63	19.27	3181.89	81.44	**4.84**
Average (50 customers on 10%, 30%, 50% 70% and 90% DoD)		**7.99**	**1166.92**	20.87	**1.44**	7.99	1189.8	19.76	**1.44**
Average (150 customers on 10%, 30%, 50% 70% and 90% DoD)		**13.23**	**2200.82**	44.45	3.4	13.29	2234.6	37.53	**3.33**
Average (250 customers on 10%, 30%, 50% 70% and 90% DoD)		**19.12**	**3114.48**	72.44	5.63	19.27	3181.9	81.44	**4.84**
Average (10% DoD on 50, 150 & 250 customers)		13.89	1927.47	80.82	**0.49**	**13.82**	**1904**	93.75	0.62
Average (30% DoD on 50, 150 & 250 customers)		**13.31**	**2014.46**	39.6	2.36	13.49	2064.5	39.36	**2.18**
Average (50% DoD on 50, 150 & 250 customers)		**12.91**	**2180.03**	33.73	3.82	**12.91**	2218.6	32.58	**3.53**
Average (70% DoD on 50, 150 & 250 customers)		**13.04**	**2260.44**	36.11	5.07	13.27	2352.4	31.86	**4.64**
Average (90% DoD on 50, 150 & 250 customers)		**14.06**	**2421.31**	39.34	5.71	14.09	2471	33.67	**5.04**
Overall Average		**13.45**	**2160.74**	45.92	3.49	13.52	2202.1	46.24	**3.2**

However, in 70% DoD, NEDPALNS (best NV – 12.73 and best TD – 2356.25) only has the least average of best NV and least average of best TD compared to ALNS (best NV – 13.07 and best TD - 2405.95). In 50% DoD, NEDPALNS has the least average of best TD (2207.25) than ALNS (best TD - 2261.93). Despite the ALNS challenging results, NEDPALNS still has the overall average result better in the best NV (13.23) and best TD (2200.82) than the ALNS (best TD – 13.29 and best TD – 2234.6). This indicates the overall average in NEDPALNS is better than ALNS in the best NV and best TD in the 150 customers comparison.

Finally, in the 250 customers, NEDPALNS has the least average of best NV and the least average of best TD in 10% DoD (best NV - 19.8 and best TD - 2809.05), 30% DoD: (best NV - 18.73 and best TD - 2884.07), 50% DoD (best NV - 17.93 and best TD - 3127.12), 70% DoD (best NV - 18.8 and best TD - 3246.42) and 90% DoD (best NV - 20.33 and best TD - 3505.72) than 10% DoD (best NV - 19.53 and best TD - 2742.8), 30% DoD (best NV - 19.07 and best TD - 3044.41, 50% DoD (best NV - 18 and best TD - 3171.79), 70% DoD (best NV - 19.07 and best TD - 3408.25) and 90% DoD (best NV - 20.67 and best TD - 3542.19) in ALNS.

Overall, NEDPALNS has the best average result with the overall least NV average (19.12) and overall least TD average (3114.48) compared to ALNS (NV - 19.27 and TD -3181.89). This shows that despite the competitive result, in the 250 customers, NEDPALNS has the best result in the least average VN and the least average TD.

If we compare the average result for specific DoD on all customer sizes (50, 150, 250 customers), NEDPALNS has the best average results in 30% DoD (NV - 13.31, TD - 2014.46), 50% DoD (NV - 12.91, TD - 2180.03), 70% DoD (NV - 13.04, TD - 2260.44) and 90% (NV - 14.06, TD - 2421.31) based on all customer sizes than ALNS in 30% DoD (NV - 13.49, TD - 2064.5), 50% DoD (NV - 12.91, TD - 2218.6), 70% DoD (NV - 13.27, TD - 2352.4) and 90% (NV - 14.09, TD - 2471). Except for 10% DoD on all customer sizes, NEDPALNS has better average result in RR (0.49) than ALNS (RR – 0.62). This means that NEDPALNS has best average in the least NV and TD in the 30% DoD, 50% DoD, 70% DoD and 90% DoD than ALNS. In the 10% DoD on all customer size, NEDPALNS only has better average rejection rates than ALNS. Overall, NEDPALNS has the overall average in NV (13.45) and TD (2160.74) outperform the ALNS (NV - 13.52 and TD - 2202.1).

Table 4.10 shows the comparison with ALNS in terms of the best, worst, and average results. NEDPALNS has a better result in the least average vehicle number and the least average total travelled distance than ALNS in 50 customers (best NV - 7.99, worst NV - 8.65, average NV - 8.24, best TD – 1166.92, worst TD - 1535.35 and average TD – 1325.71), 150 customers (best NV - 13.23, worst NV - 15.28, average NV - 14.10, best TD – 2200.82, worst TD - 2783.70 and average TD – 2439.65) and 250 customers (best NV - 19.12, average NV - 20.59, best TD – 3114.48, worst TD - 4026.90 and average TD – 3518.49) than ALNS's 50 customers (best NV - 7.99, worst TD - 8.92, average NV - 8.35, best TD – 1189.79, worst TD - 1577.31 and average TD – 1357.00), 150 customers (best NV - 13.29, worst TD - 15.65, average NV - 14.34, best TD – 2234.60, worst TD - 2937.21 and average TD – 2546.00) and 250 customers (NV - 19.27, average NV - 20.69, best TD – 3181.89, worst TD - 4086.83 and average TD – 3569.25).

If we compare each DoD on all the customer's sizes (50, 150 and 250 customers), NEDPALNS has the least average NV and least average TD in 10% DoD (best NV - 11.77, worst TD - 13.22, average NV - 12.35, worst TD - 2584.56 and average TD – 2205.91), 30% DoD (best NV - 13.31, worst TD - 14.56, average NV - 13.82, best TD – 2014.46, worst TD - 2585.06 and average TD – 2258.56), 50% DoD (best NV - 12.91, worst TD - 14.53, average NV - 13.55, best TD – 2180.03, worst TD - 2777.28 and average TD – 2436.55), 70% DoD (best NV - 13.04, worst TD - 15.62, average NV - 14.18, best TD – 2260.44, worst TD - 2923.07 and average TD – 2549) and 90% DoD (best NV - 14.06, worst TD - 16.98, average NV - 15.49, best TD – 2421.31, worst TD - 3039.94 and average TD – 2689.73) than ALNS in 10% DoD (best NV - 11.82, worst TD - 13.29, average NV - 12.41, worst TD - 2630.47 and average TD – 2216.67), 30% DoD (best NV - 13.49, worst TD - 14.93, average NV - 14.02, best TD – 2064.47, worst TD - 2686.1 and average TD – 2319.64), 50% DoD (worst TD - 15.04, average NV - 13.75, best TD – 2218.62, worst TD - 2866.43 and average TD – 2490.74), 70% DoD (best NV - 13.27, worst TD - 15.78, average NV - 14.47, best TD – 2352.43, worst TD - 3022 and average TD – 2658.67) and 90% DoD (best NV - 14.09, worst TD - 17.36, average NV - 15.68, best TD – 2471, worst TD - 3130.54 and average TD – 2768.01).

In short, NEDPALNS still have the least overall average in NV (best - 13.45, worst - 15.46, average - 14.31) and TD (best - 2160.74, worst - 2781.98, average - 2427.95) than ALNS (best NV - 13.52, worst NV - 15.66, average NV - 14.46, best TD – 2202.09, worst TD - 2867.12 and average TD – 2490.75).

Table 4.10: Comparison with ALNS (best, worst and average)

Customer Size	Degree of Dynamism	Average Ratio of refuse (NEDPALNS)			Average Ratio of refuse (ALNS) 2018			Average vehicle number (NEDPALNS)			Average vehicle number (ALNS)			Average total travelled distance (NEDPALNS)			Average total travelled distance (ALNS)		
		Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg.	Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg
50	90	3.13	**4.6**	4.15	**3**	4.67	**3.91**	**7.73**	**8.13**	**7.86**	7.47	8.67	7.99	**1260.87**	**1595.28**	**1416.59**	1284.42	1595.96	1432.7
	70	2.8	4.33	3.85	2.67	4.07	3.55	**7.6**	**8.27**	**7.86**	7.67	8.67	8.11	**1178.65**	**1536.72**	**1346.98**	1243.09	1629.85	1406.21
	50	**0.6**	**1.87**	1.53	0.67	**1.87**	1.46	**8**	**8.6**	**8.22**	8	8.73	8.29	**1205.73**	**1587.89**	**1359.43**	1222.15	1599.4	1385.73
	30	**0.6**	**1.27**	**1.07**	0.67	1.27	1.07	**8.2**	**9**	**8.63**	8.33	9.27	8.67	**1097.3**	**1430.11**	**1236.55**	1119.07	1512.45	1294.23
	10	**0.07**	**0.33**	**0.24**	0.2	**0.33**	0.28	**8.4**	**9.27**	**8.65**	8.47	9.27	8.71	1092.07	**1526.74**	1268.99	1080.24	1548.87	1266.15
	Average	**1.44**	2.48	2.17	**1.44**	**2.44**	**2.05**	**7.99**	**8.65**	**8.24**	7.99	8.92	8.35	**1166.92**	**1535.35**	**1325.71**	1189.79	1577.31	1357.00
150	90	**5.53**	**10.53**	**8.19**	5.73	10.8	8.99	**14.13**	**16.67**	**15.36**	14.13	17.2	15.64	**2497.33**	**3035.45**	**2713.89**	2586.38	3302.8	2893.01
	70	4.93	**10.2**	8.11	**4.73**	10.27	**8.02**	**12.73**	**15.73**	**14.08**	13.07	16.07	14.54	**2356.25**	**3097.55**	**2637.23**	2405.95	3104.42	2744.83
	50	3.13	5.93	4.84	**2.93**	**5.8**	**4.71**	12.8	**15.07**	**13.59**	**12.73**	15.87	13.99	**2207.25**	**2796.42**	**2460.26**	2261.93	3012.65	2581.9
	30	2.8	**4.13**	**3.67**	**2.6**	4.27	3.68	**13**	**14**	**13.39**	13.07	14.2	13.45	2062	**2588.19**	**2272.71**	**2029.92**	2641.87	2295.56
	10	**0.6**	**1.4**	**1.1**	0.67	1.47	1.18	**13.47**	**14.93**	14.09	**13.47**	**14.93**	**14.07**	**1881.28**	**2400.87**	**2114.16**	1888.82	2624.33	2214.68
	Average	3.40	**6.44**	**5.18**	**3.33**	6.52	5.32	**13.23**	**15.28**	**14.10**	13.29	15.65	14.34	**2200.82**	**2783.70**	**2439.65**	2234.60	2937.21	2546.00
250	90	8.47	17.4	13.09	**6.4**	**16.93**	**12.88**	**20.33**	**26.13**	**23.25**	20.67	26.2	23.4	**3505.72**	**4489.1**	**3938.71**	3542.19	4492.85	3978.33
	70	7.47	13.8	11.19	6.53	**13.4**	**10.58**	**18.8**	22.87	**20.59**	19.07	**22.6**	20.76	**3246.42**	**4134.93**	**3662.78**	3408.25	4331.72	3824.98
	50	7.73	12.07	10.44	**7**	**11.87**	**10.15**	**17.93**	**19.93**	**18.85**	18	20.53	18.97	**3127.12**	**3947.53**	**3489.97**	3171.79	3987.23	3504.6
	30	3.67	6.13	5.37	**3.27**	**6**	**5.08**	**18.73**	**20.67**	**19.43**	19.07	21.33	19.95	**2884.07**	**3736.89**	**3266.42**	3044.41	3904.15	3369.13
	10	**0.8**	**2.27**	**1.68**	1	2.33	1.79	19.8	22.6	20.82	**19.53**	**21.4**	**20.35**	2809.05	3826.06	3234.57	**2742.8**	**3718.21**	**3169.19**
	Average	5.63	10.33	8.35	**4.84**	**10.11**	**8.10**	**19.12**	22.44	**20.59**	19.27	**22.41**	20.69	**3114.48**	**4026.90**	**3518.49**	3181.89	4086.83	3569.25
Average (50 customers on 10%, 30%, 50%, 70% and 90% DoD)		**1.44**	2.48	2.17	**1.44**	**2.44**	**2.05**	**7.99**	**8.65**	**8.24**	7.99	8.92	8.35	**1166.92**	**1535.35**	**1325.71**	1189.79	1577.31	1357.00
Average (150 customers on 10%, 30%, 50%, 70% and 90% DoD)		3.40	**6.44**	**5.18**	**3.33**	6.52	5.32	**13.23**	**15.28**	**14.10**	13.29	15.65	14.34	**2200.82**	**2783.70**	**2439.65**	2234.60	2937.21	2546.00

Average (250 customers on 10%, 30%, 50%, 70% and 90% DoD)	5.63	10.33	8.35	**4.84**	**10.11**	**8.10**	**19.12**	22.44	**20.59**	19.27	**22.41**	20.69	**3114.48**	**4026.90**	**3518.49**	3181.89	4086.83	3569.25
Average (10% DoD on 50, 150 and 250 customers)	**0.49**	**1.33**	**1.01**	0.62	1.38	1.08	**11.77**	**13.22**	**12.35**	11.82	13.29	12.41	1927.47	**2584.56**	**2205.91**	**1903.95**	2630.47	2216.67
Average (30% DoD on 50, 150 and 250 customers)	2.36	**3.84**	3.37	**2.18**	3.85	**3.28**	**13.31**	**14.56**	**13.82**	13.49	14.93	14.02	**2014.46**	**2585.06**	**2258.56**	2064.47	2686.16	2319.64
Average (50% DoD on 50, 150 and 250 customers)	3.82	6.62	5.6	**3.53**	**6.51**	**5.44**	**12.91**	**14.53**	**13.55**	**12.91**	15.04	13.75	**2180.03**	**2777.28**	**2436.55**	2218.62	2866.43	2490.74
Average (70% DoD on 50, 150 and 250 customers)	5.07	9.44	7.72	**4.64**	**9.25**	**7.38**	**13.04**	**15.62**	**14.18**	13.27	15.78	14.47	**2260.44**	**2923.07**	**2549**	2352.43	3022	2658.67
Average (90% DoD on 50, 150 and 250 customers)	5.71	10.84	**8.48**	**5.04**	**10.8**	8.59	**14.06**	**16.98**	**15.49**	14.09	17.36	15.68	**2421.31**	**3039.94**	**2689.73**	2471	3130.54	2768.01
Overall Average	3.49	6.42	5.23	**3.2**	**6.36**	**5.16**	**13.45**	**15.46**	**14.31**	13.52	15.66	14.46	**2160.74**	**2781.98**	**2427.95**	2202.09	2867.12	2490.75

Appendix A37-A41 shows that the hypervolume of NEDPALNS in 10% DoD (0.4325), 30% DoD (0.3858), 50% DoD (0.4231) and 70% DoD (0.3343) outperform ALNS in 10% DoD (0.4279), 30% DoD (0.3756), 50% DoD (0.4102), 70% DoD (0.3258). Although the NEDPALNS hypervolume in 90% DoD (0.3277) is underperformed than ALNS (0.3387). Overall, of the 50 customers, NEDPALNS (0.3807) has an overall average hypervolume better than ALNS (0.3756). These results indicate NEDPALNS has better diversity and convergence than ALNS in the 50 customers.

Appendix A42-A46 show the hypervolume of NEDPALNS in 50% DoD (0.4157), 70% DoD (0.4231) and 90% DoD (0.4256) outperform ALNS in 50% DoD (0.3988), 70% DoD (0.4126) and 90% DoD (0.4194) in the 150 customers. However, NEDPALNS has a better hypervolume average (0.4035) than ALNS (0.4009) in the 150 customers. These results indicate NEDPALNS has better diversity and convergence than ALNS in the 150 customers.

In the 250 customers comparisons, NEDPALNS has better performance in 10% DoD (0.4300), DoD 50% (0.4376) and 70% DoD (0.4694) compared to ALNS in 10% DoD (0.4209), 50% DoD (0.3737) and 70% DoD (0.4584) as shown in Appendix A47-A51. The overall result in the 250 customers, NEDPALNS (0.4383) has a better hypervolume result than ALNS (0.4315). This indicates NEDPALNS has overall better convergence and diversity than ALNS in the 250 customers.

In conclusion, NEDPALNS (0.4075) has an overall hypervolume average better than ALNS (0.4027) for all customers (50, 150, and 250 customers).

4.6 Measurements

We conduct three types of measurements. The first type of measurement uses hypothetical VRPTW and MOVRPTW datasets (i.e., the Solomon dataset) to assess NEDPALNS performance. In the second type of measurement, we evaluate NEDPALNS performance using a hypothetical MODVRPTW dataset (i.e., Solomon dataset and dynamic dataset), and in the final type of measurement, we use a real MODVRPTW dataset (i.e., MODVRPTW) dataset and dynamic dataset) to assess NEDPALNS performance.

4.6.1 First Measurements

In VRPTW, we measure single objective results on speed, optimal solutions, and average optimal solutions.

4.6.1.1 Speed and Optimal Solution (Total Travelled Distance)

In speed, the least response time is 0.21 seconds after 30 runs, and the average least response time after 30 runs is 85.81 seconds. If we compare NEDPALNS's optimal solution (total travelled distance) against the published algorithms as shown in Figure 4.17.

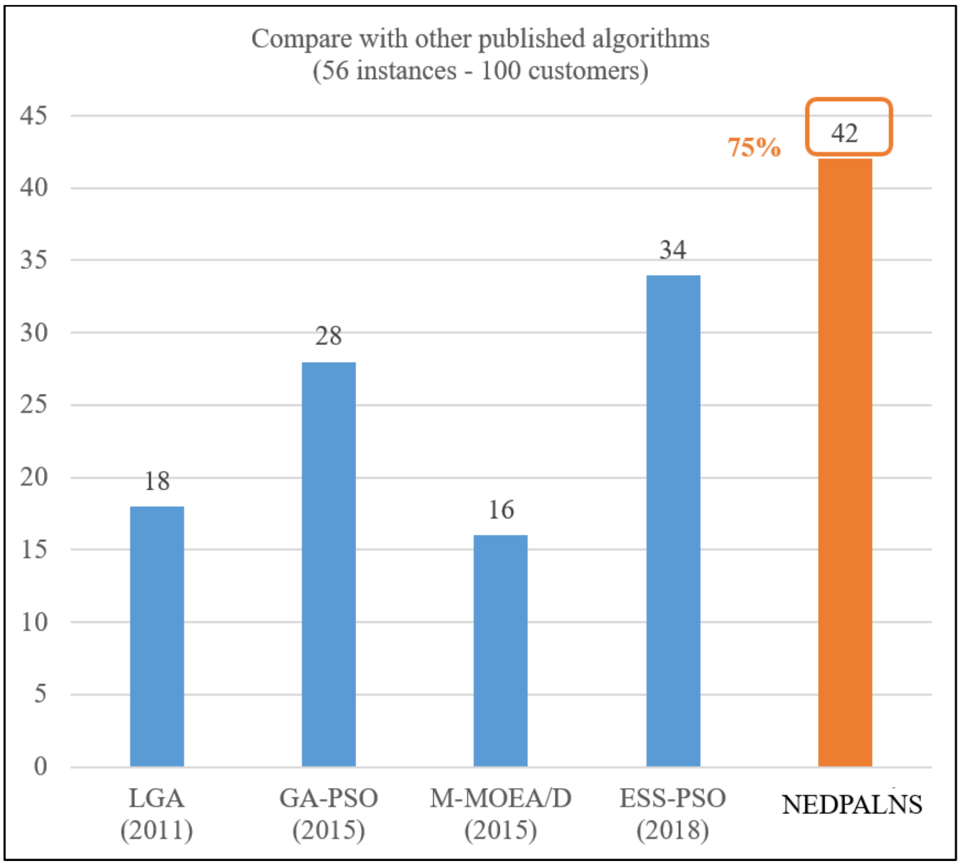


Figure 4.17: Compare with other published algorithms on optimal solutions

NEDPALNS has 42 results that outperform the other published algorithms. This is equivalent to 75% of NEDPALNS results being better than other published algorithms.

If we compare with the best-known solutions, 71% of our results are equal to the best-known solutions. This is shown in Figure 4.18.

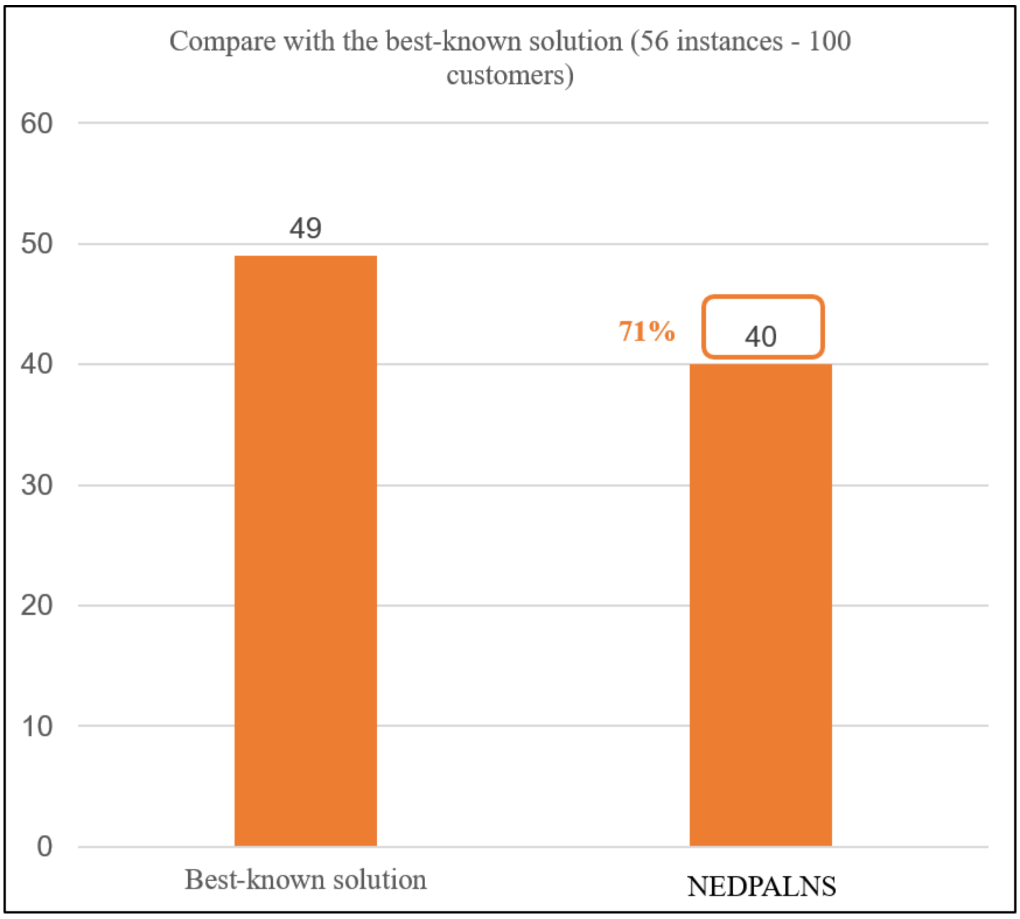


Figure 4.18: Compare the best-known solutions with optimal solutions

This shows that NEDPALNS are quite efficient in the least response time after 30 runs and the least average response time on the 30 runs. When comparing the optimal solution in the least total travelled distance, NEDPALNS outperforms other published algorithms and is competitive with the best-known solution.

4.6.1.2 Average Optimal Solution

In the average optimal solution (30 runs) comparisons, NEDPALNS has the least standard deviation compared to other published algorithms as shown in Figure 4.19. NEDPALNS has a better optimal solution compared to published algorithms and is competitive with the best-known solutions.

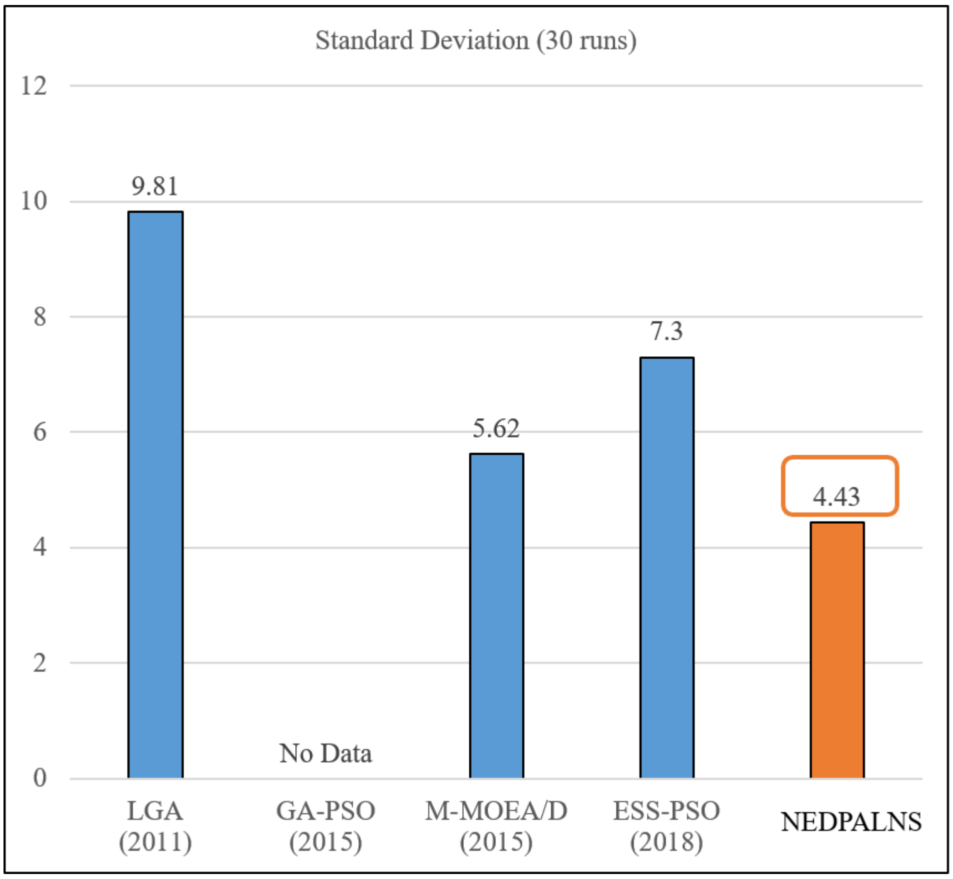


Figure 4.19: Compare with other published algorithms on average optimal solutions standard deviation

In the average optimal solution against the published algorithms, NEDPALNS outperforms other published algorithm results by 86% as shown in Figure 4.20.

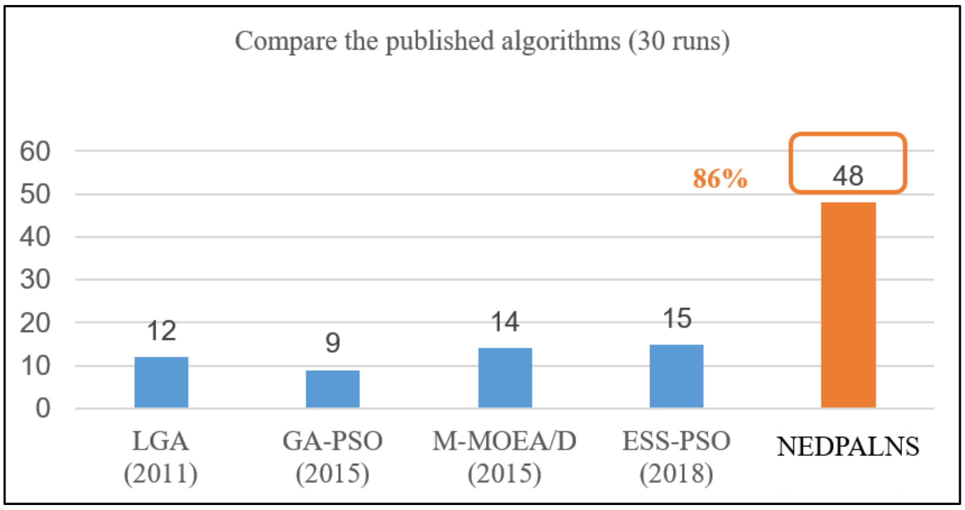


Figure 4.20: Compare with other published algorithms on average optimal solutions

In the average optimal solution against the best-known solutions, NEDPALNS outperforms the best-known solutions by 50% as shown in Figure 4.21.

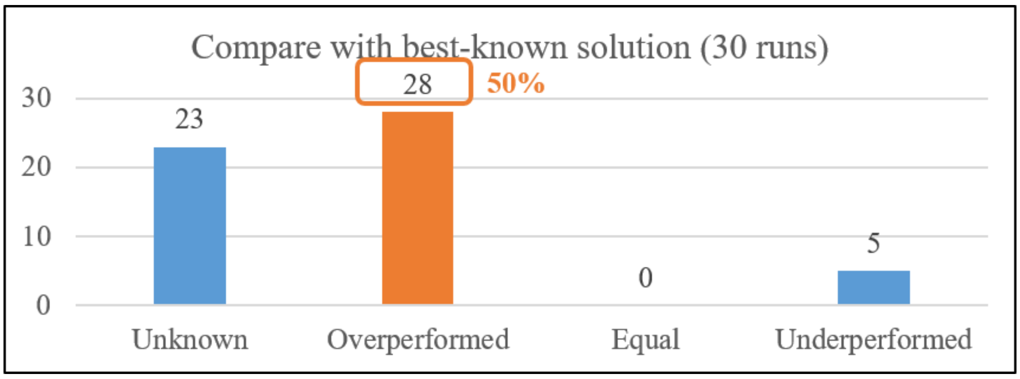


Figure 4.21: Compare the best-known solutions on average optimal solutions

In this average optimal solutions comparison, the optimal solutions generated by NEPALNS are quite consistent. It outperforms other published algorithms and is competitive with the best-known solutions.

4.6.1.3 Pareto Set

In hypervolume comparison with other published algorithms' hypervolume, NEDPALNS outperforms other published algorithms in R1, R2, RC1, and RC2 instance types as shown in Figure 4.21. In the non-dominated solutions size comparison,

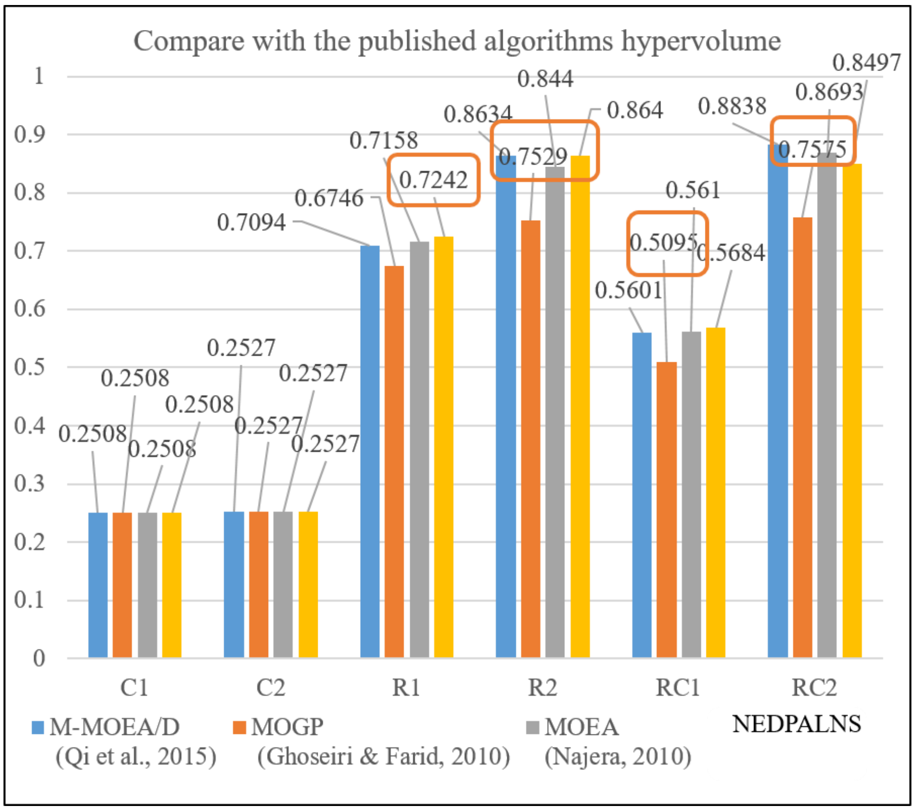


Figure 4.22: Compare with other published algorithms on hypervolume

NEDPALNS has a better non-dominated solutions size than other published algorithms which is also in R1, R2, RC1, and RC2 instance types. These results show that NEDPALNS has better Pareto efficiency than other published algorithms particularly in the R1, R2, RC1, and RC2 instance types.

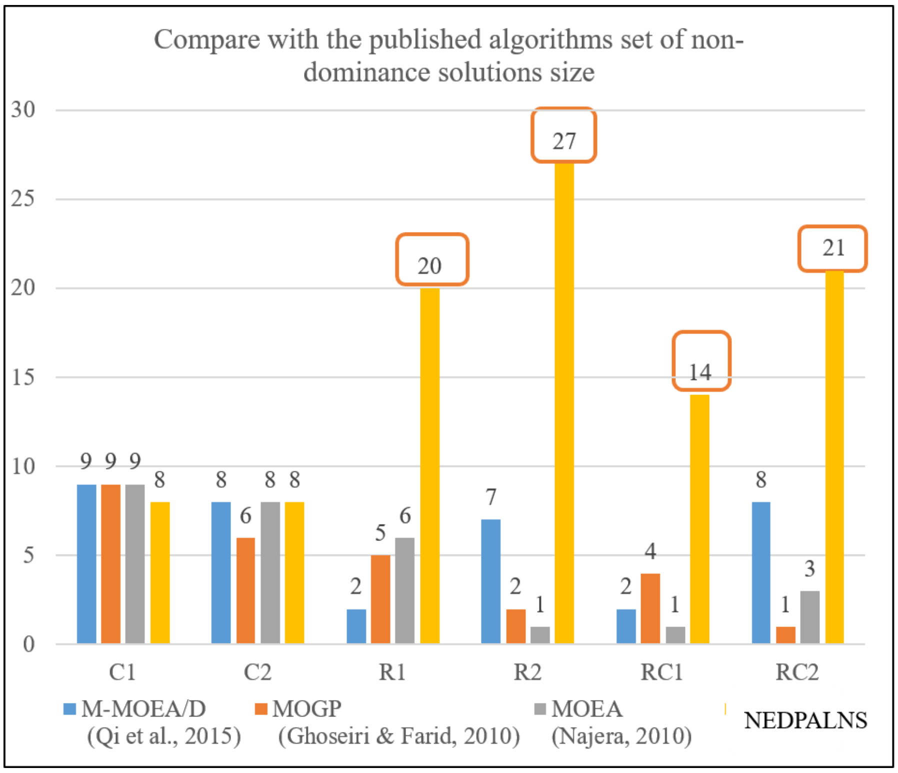


Figure 4.23: Compare with other published algorithms on Pareto Set non-dominance solutions size

4.6.1.4 Optimal Solution (Number of Used Vehicles and Total Travelled Distance)

If we compare optimal solutions on both the number of used vehicles and total travelled distance, 63% of the results in NEDPALNS are better than other published algorithms as shown in Figure 4.24.

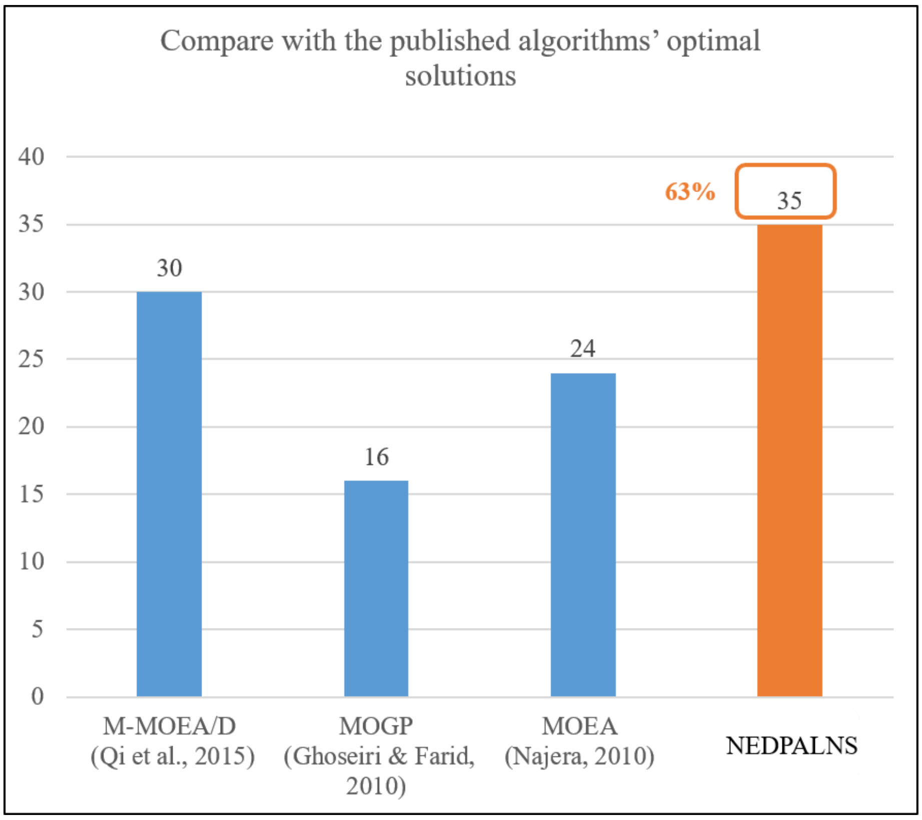


Figure 4.24: Compare with other Published Algorithms Optimal Solutions (Number of Used Vehicles and Total Travelled Distance)

In the optimal solution compared with the best-known solutions on the least number of used vehicles, 33% of the results in NEDPALNS are equalled to the best-known solutions results which are shown in Figure 4.25.

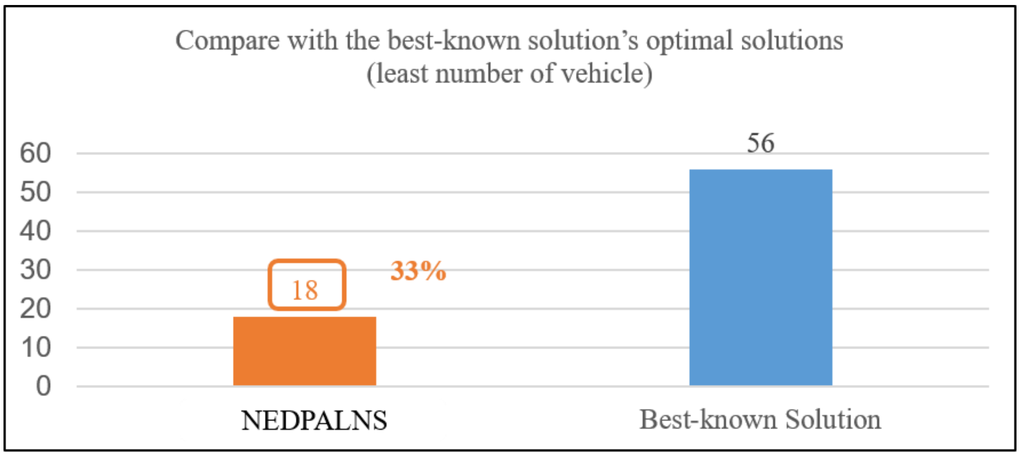


Figure 4.25: Compare the Best-Known Solution's Optimal Solutions on the Least Number of Used Vehicles

In the comparison with the best-known solutions' optimal solutions on the least total travelled distance, 75% of results in NEDPALNS are equalled to the best-known solutions.

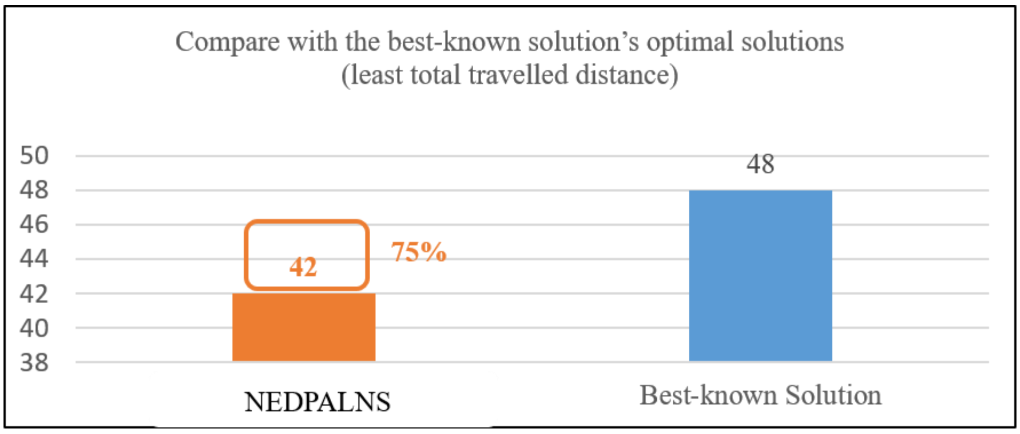


Figure 4.26: Compare the Best-Known Solution's Optimal Solutions on the Least Total Travelled Distance

These results show that NEDPALNS has a better optimal solution (both the number of used vehicles and total travelled distance) than other published algorithms and is competitive with the best-known solutions.

4.6.1.5 Average optimal solutions (Number of Used Vehicles and Total Travelled Distance)

Figure 4.27 shows that NEDPALNS has better average optimal solutions than other published algorithms in C1, C2, R1, and RC1 instance types. If we compare with the average optimal solutions in the best-known solutions, 93% of the results in NEDPALNS outperform the best-known solutions as shown in Figure 4.28.

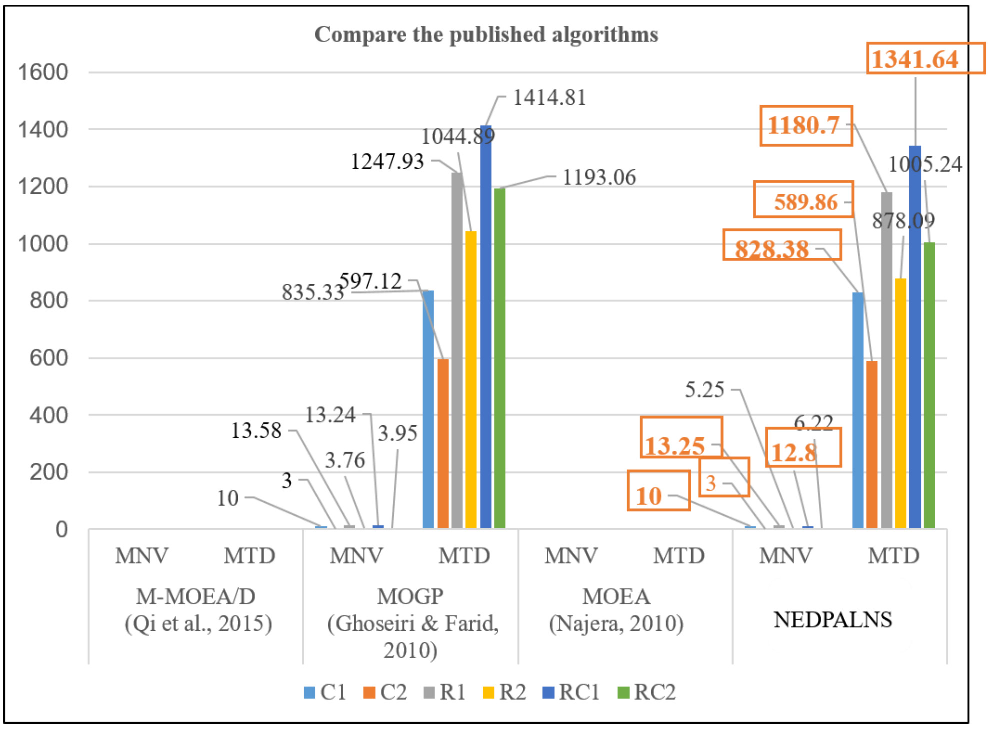


Figure 4.27: Compare Average Optimal Solutions with the other Published Algorithms

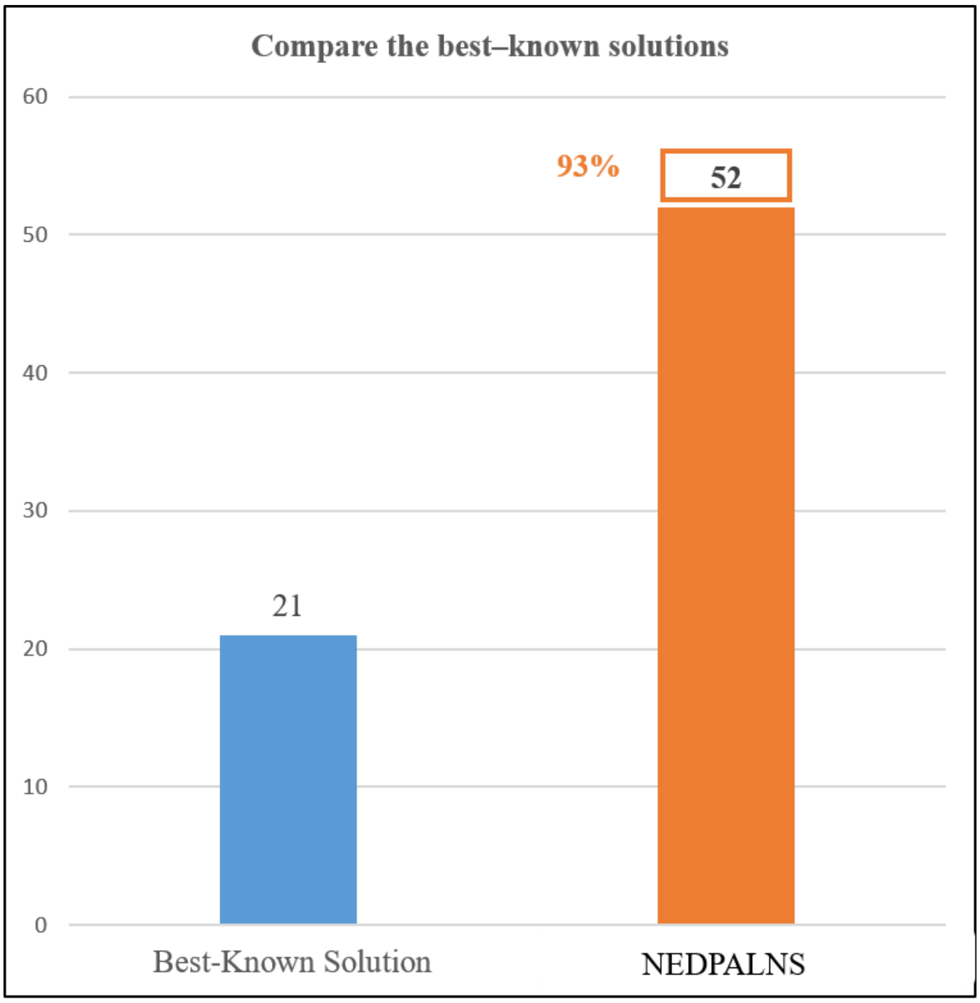


Figure 4.28: Compare Average Optimal Solutions with the Best-Known Solution

These results show that NEDPALNS generally has better optimal solutions on average than other published algorithms and the best-known solutions.

4.6.2 Second Measurements

In this measurement, firstly, we compare optimal solutions against other published algorithms in the least average of the best number of vehicles, least average of best total travelled distance and least average of best rejection rates. Secondly, we compare against ALNS on the least average of best, worst, and an average number of vehicles, total travelled distance, and rejection rates. Lastly, the comparison with ALNS Pareto set.

4.6.2.1 Optimal Solution (the least average of best VN, best TD, and best RR)

NEDPALNS has the least average of best rejection rates if compare with other published algorithms which are shown in Figure 4.28. This result

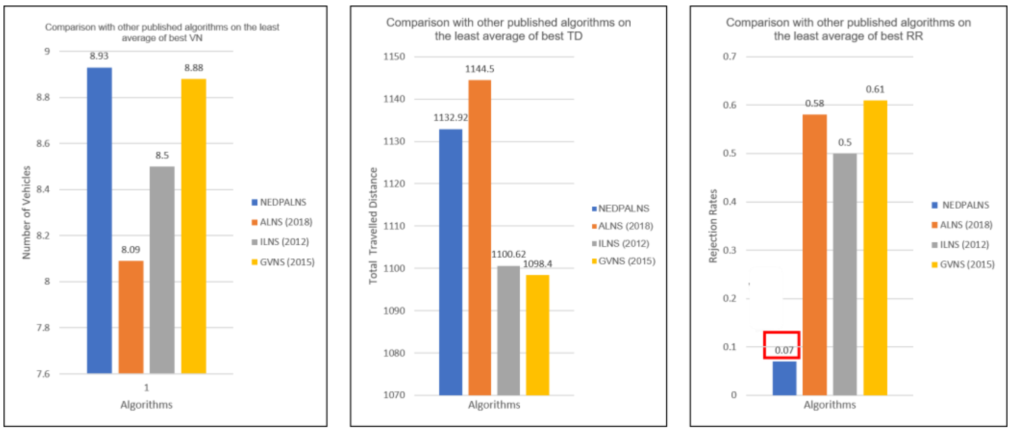


Figure 4.28: Compare Optimal Solution with other Published Algorithms (the least average of the best number of vehicles, best total travelled distance, and best rejection rates)

shows that NEDPALNS can accept more customers than other published algorithms.

4.6.2.2 Optimal Solution (the least average of best, worst, and average on VN, TD, and RR)

Also, if compare with ALNS, NEDPALNS has the least average of best, worst, average on rejection as shown in Figure 4.29. These results imply that NEDPALNS can accept more customers even if the result of the least rejection rates is the best, worst, or average.

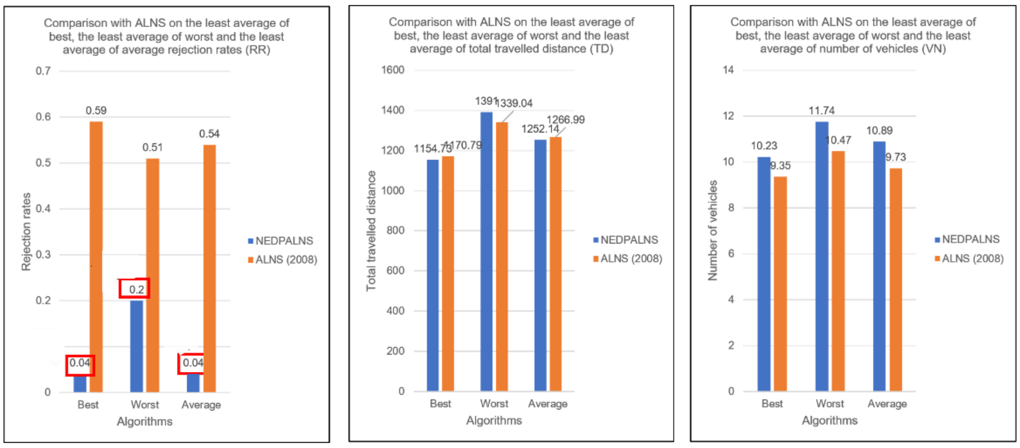


Figure 4.29: Compare optimal solution with ALNS on the least average of best, worst, and average on Number of Vehicles, Total Travelled Distance, and Rejection Rates

4.6.2.3 Pareto set

In this Pareto set comparison, NEDPALNS outperforms ALNS in the 10% DoD, 30% DoD, 50% DoD, 70% DoD and 90% DoD regarding Pareto set as shown in Figure 4.30. These results show that NEDPALNS are Pareto efficient that ALNS in all the participating DoDs.

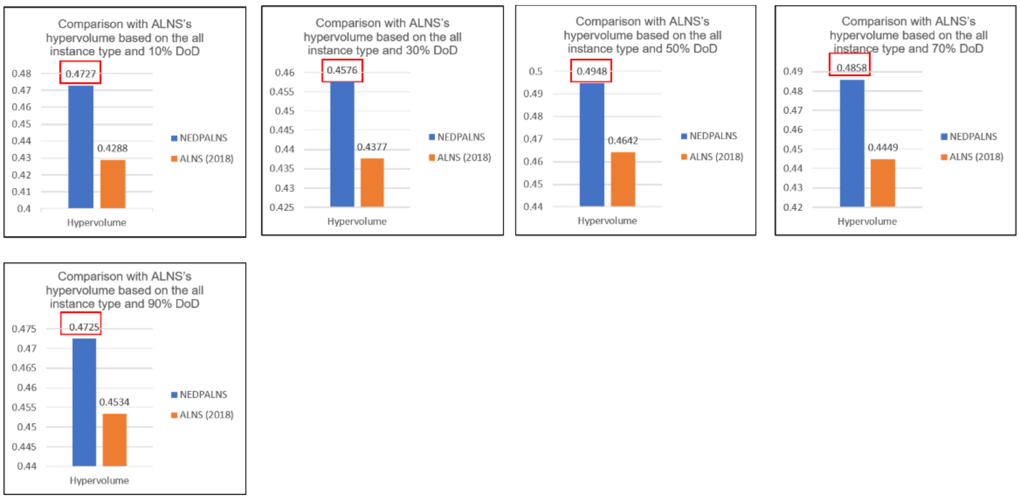


Figure 4.30: Compare Pareto set with ALNS on all Degree of Dynamisms

4.6.3 Third Measurements

In this last measurement, we compare optimal solutions against other published algorithms in the least average of the best number of vehicles, least average of best total travelled distance and least average of best rejection rates. The least average of best, worst, and average number of vehicles, total travelled distance, and rejection rates and with the ALNS Pareto set.

4.6.3.1 Optimal Solution (the least average of best VN, best TD, and best RR)

NEDPALNS has the least average of the best number of used vehicles and best total travelled distance if compare with other published algorithms which are shown in Figure 4.31. This result shows that NEDPALNS has the least average of the best number of used vehicles and best total travelled distance than other published algorithms.

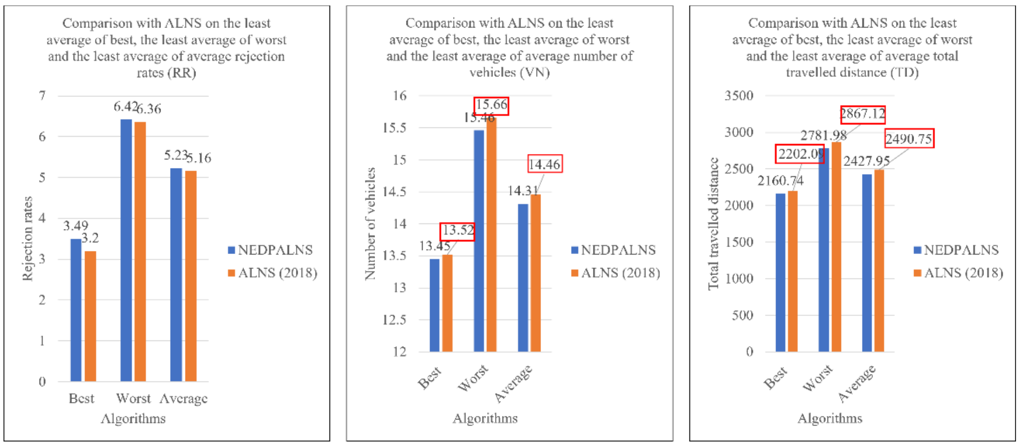


Figure 4.31: Compare Optimal Solution with other Published Algorithms (the least average of the best number of vehicles, best total travelled distance, and best rejection rates)

4.6.3.2 Optimal Solution (the least average of best, worst, and average on VN, TD, and RR)

If compare optimal solution on the least average of best, worst, and average on the number of used vehicles, total travelled distance and rejection ratio with ALNS, NEDPALNS has the least average of best, worst, average on the number of used vehicles and total travelled distance as shown in Figure 4.32. These results imply that NEDPALNS has the least number of used vehicles and total travelled distance even the result of the number of used vehicles and total travelled distance is at its best, worst, or average state.

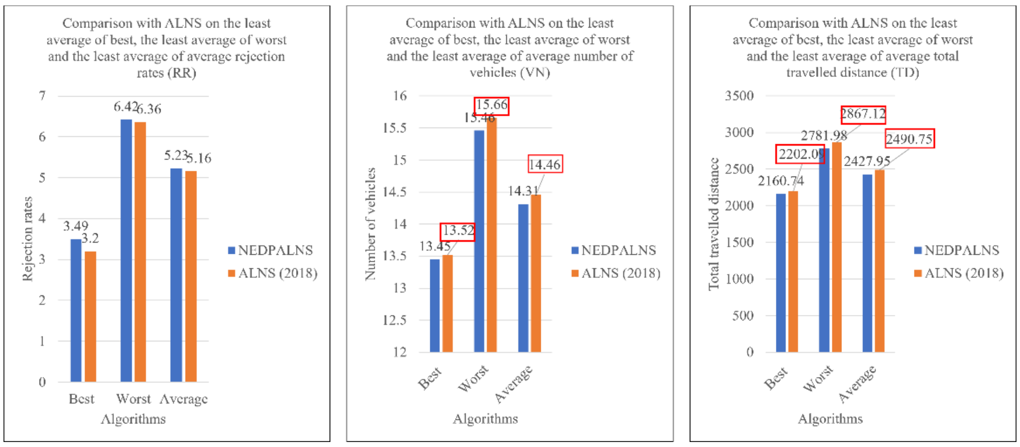


Figure 4.32: Compare optimal solution with ALNS on the least average of best, worst, and average on Number of Vehicles, Total Travelled Distance, and Rejection Rates

4.6.3.3 Pareto set

In this Pareto set comparison, NEDPALNS outperforms ALNS in the 50 customers, 150 customers and 250 customers regarding Pareto set as shown in Figure 4.30. These results show that NEDPALNS are Pareto efficient that ALNS in all the participating customer sizes.

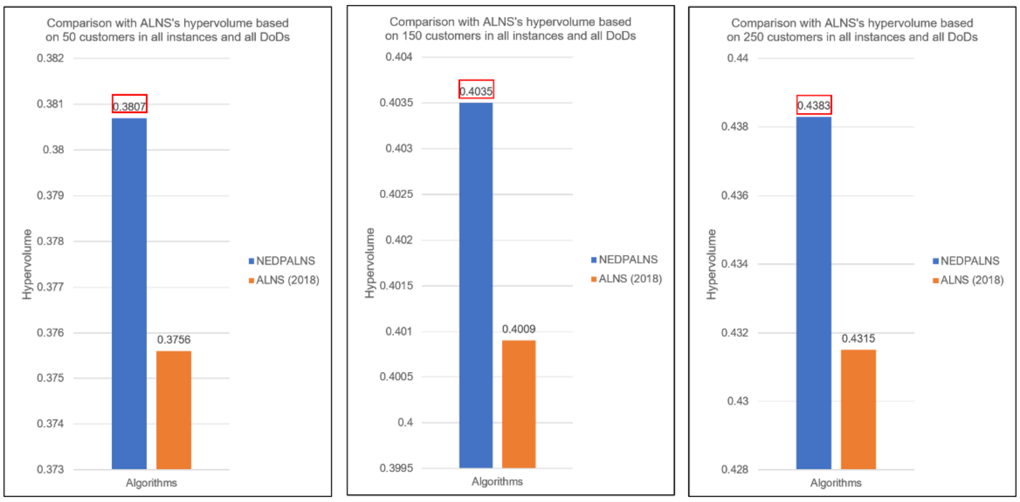


Figure 4.33: Compare Pareto set with ALNS on all Degree of Dynamisms

4.6. Summary

Comprehensive experiments are carried out on the MODVRPTW using three types of datasets. The first dataset uses the Solomon dataset to test the static information of MODVRPTW. The second dataset uses the Solomon and dynamic dataset to solve MODVRPTW and the third dataset uses the MOVRPTW and dynamic dataset to solve the MODVRPTW. The difference between the first, second, and third datasets is that the first and second datasets are hypothetical, and the third is the real dataset. In hypothetical data, we test with 100 customers, whereas in a real dataset, the customer size used for testing is 50, 150, and 250. These experiments are carried out to determine whether NEDPALNS can perform effectively using hypothetical datasets but also real datasets.

In the static testing using hypothetical datasets, the NEDPALNS results are compared with other published algorithms and the BKS. NEDPALNS demonstrated superior results in hypervolume against the other published algorithms, the least minimum of TD against other published algorithms, the least minimum of average TD against other published algorithms as well as the least minimum NV and the minimum TD against the BKS. These results indicated that NEDPALNS has the most optimized NV and TD, better diversity, and convergence than other published algorithms. These results also indicate that NEDPALNS results are highly reliable and optimized compared to BKS.

In the dynamic testing using a hypothetical dataset, NEDPALNS results have the least average of best RR, the least average of worst RR, and the least average of average RR based on all DODs and all instance types against other published algorithms. NEDPALNS also demonstrated better hypervolume versus ALNS. These results show that NEDPALNS can accommodate more customers and overall has better diversity and convergence.

In dynamic testing using a real dataset, NEDPALNS has the least average NV (best, worst, and average), the least average TD (best, worst, and average), and a better hypervolume compared to ALNS. These results show that NEDPALNS has the least average NV, the least average TD, and better diversity and convergence compared to ALNS in dynamic testing using real datasets. Overall, the performance of NEDPALNS outweighs that of other published algorithms and the BKS in static testing that uses hypothetical datasets and in dynamic testing that uses both hypothetical datasets and real datasets.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

Logistics play a vital role in the process of planning and executing the transportation of goods from the original destination to the final destination. Globally, logistics revenue is worth trillions of dollars. During the Covid-19 pandemic, many businesses have closed, but the logistics business remains resilient and versatile.

One of the key trends in shaping the future of logistics is to apply artificial intelligence. These works include developing an efficient and effective dynamic multi-objective algorithm. The development of a dynamic multi-objective algorithm is important because it has a close resemblance to the ubiquitous service rendered such as dial-a-ride, Grab services, food delivery services, courier services, and taxi services among others. The development of an optimized algorithm is a complex, challenging, and time-consuming task because many constraints need to be fulfilled and the result of the algorithm may not be as optimized as other established algorithms. Hence, an algorithm that can optimize better results than other algorithms is highly sought after.

To the best of our knowledge, there have been numerous studies being conducted on the VRPTW but there are limited studies on the DVRPTW let alone MODVRPTW. Therefore, it is important and beneficial economically and commercially to study and research MODVRPTW.

1. *To develop a multi-objective algorithm with a distributed parallelized adaptive rebuilding capability that uses cyclic and non-cyclic optimization strategies.*

We proposed a non-fitness evolutionary distributed parallelized adaptive large neighbourhood search (NEDPALNS) algorithm to solve MODVRPTW. The NEDPALNS is based on two popular algorithms. This first algorithm is the evolutionary algorithm. The classical evolutionary algorithm is based on four core steps, namely fitness calculation, selection, crossover, and mutation. In NEDPALNS, we reorder the classical fitness-oriented solution for the selection process. This means that we will not be based on the fittest candidate for selection, instead of on those least fit candidates. The second algorithm is based on the local neighbourhood search algorithm. The main task of this algorithm is to rebuild the solution. The rebuilding procedure removes part of a solution and repairs the ruined solution. There are many removal and repair procedures. Some of the popular removal procedures namely random procedure, radial procedure, related procedure, and critical procedure among others are used in this development. Other repair procedures are the best insertion and variants of regret insertion are proposed. The purpose of choosing an evolutionary algorithm is because it is a population-based metaheuristic that can

perform exploration of the search, which widens the chances of selecting more interim solutions. On the other hand, the PALNS, single population metaheuristic allows performing exploration and exploitation of search at the same time. Thus, this combination of both algorithms enables the generation of the best solutions through non-cyclic and cyclic optimization. NEDPALNS is designed in part based on microservice architecture.
This means that part of the NEDPALNS module can be executed remotely. In our design, the PALNS is designed in a granular manner. This means that the PALNS is capable of being distributed logically (virtualization, containerization, and on the cloud) or run on bare metal (commodity hardware).

The process flow of the NEDPALNS is divided into two lifecycles. One is the evolutionary lifecycle, and the other is the generation lifecycle.
The evolutionary lifecycle performs a set of procedures in sequence, namely non-fitness selection, population initialization, solutions intercross, and solutions morph. The generation lifecycle comprises the evolutionary lifecycle and a re-optimization cycle. NEDPALNS operates two types of optimizations which are noncyclic optimization and cyclic optimization. Non-cyclic optimization performs one-time optimization. Cyclic optimization performs perpetual optimization until a termination condition is met. This process flow enables the chosen solutions to escape local optima while searching for global optima. The gist of this process flow is to select the top-performing interim solution from the interim solution pool. The interim solution in the population is sorted and the best interim solution is

selected for further refinement in the re-optimization procedure which runs perpetually until the termination criteria are met.

2. *The objectives of this thesis are defined in Chapter 1 and its outcomes are listed as follows: To be able to support hypothetical and real datasets that consistently generated near-optimal solutions and to achieve an optimized Pareto set.*

To verify whether our proposed algorithm works, we perform three types of tests. The first type of test performs a static test that uses a hypothetical dataset that consists of the Solomon dataset. The second type of test performs a dynamic test that uses a hypothetical dataset that consists of Solomon and dynamic dataset, and the third type which also perform the dynamic test but uses the real dataset which consists of MOVRPTW and dynamic dataset. The first type of test is tested on static information and the second and third type of test is tested on dynamic information.

In the static test, we used 100 customers in the Solomon datasets and the customers are distributed in the Euclidean plane. These datasets are divided into six instance types namely R1, R2, C1, C2, RC1, and RC2 instance types. There are 56 instances. Each instance type has a different type of customer distribution, service time, and time windows.

In the dynamic test, we use the real data from a distribution company in Tenerife, Spain, whose core business is to provide food products delivery that serves around 150 customers per day or 1000 customers per week. The

Google Maps database is used to measure the travel distance and travel time between the customers. This type of measurement is a unique and non-symmetrical and realistic representation of travel distance and travel time. Hence, the travel time in urban areas is more time-consuming than the travel time in rural areas.

We performed three types of comparisons to determine NEDPALNS hypervolume performance. First, the first comparison uses the Solomon dataset. The comparison against other algorithms is listed as follows:

- *The overall average in hypervolume comparison against other published algorithms*

  This result is calculated based on the overall average in hypervolume in all the degrees of dynamism and instance types. NEDPALNS's overall average in hypervolume is 0.5821 while MOEA, MOGPGA, and M-MOEA/D achieve 0.5326, 0.5865, and 0.5848 respectively. Although MOGPGA and M-MOEA/D have better overall hypervolume results than NEDPALNS in instance type RC2, NEDPALNS outperforms MOGPGA and M-MOEA/D in most of the instance types such as C1, C2, R1, R2, and RC1 instance types.

The second comparison uses Solomon and dynamic datasets.

- *The hypervolume overall average comparison against the ALNS*

  These results are calculated based on the overall hypervolume average in all degrees of dynamism and all instance types. NEDPALNS achieves 0.4767 in the overall hypervolume average, while ALNS has 0.4458. NEDPALNS outperforms ALNS on the overall hypervolume average in all degrees of dynamism. This shows that NEDPALNS has better diversity and convergence in the hypervolume overall average.

Thirdly, we compare the MOVRPTW and dynamic dataset. The comparisons are listed as follows:

- *The overall average in hypervolume comparison against the ALNS*

  This result is calculated based on the overall average in hypervolume. NEDPALNS achieves 0.4075 in the overall hypervolume average, while ALNS achieves 0.4027. NEDPALNS outperforms ALNS on the overall hypervolume average in all degrees of dynamism. This shows that NEDPALNS has better diversity and convergence in the hypervolume overall average.

3. To evaluate the performance of the proposed algorithm against the recently published results and best-known solutions.

   We perform three types of comparisons to ascertain NEDPALNS performance. First, we test using a Solomon dataset. The comparison against other algorithms is listed as follows:

- *The least number of used vehicles and the least total travelled distance in other published algorithms by instance type.*

NEDPALNS has the least number of used vehicles and the least total travelled distance in instance types R1, R2, RC1, and RC2.

- *The least number of used vehicles with the best-known solutions.*

  NEDPALNS has 18 results similar to the best-known solutions in the least number of used vehicles. This is equivalent to 32% of the total instances used for comparison. This shows that NEDPALNS has good records of the best results in the least number of used vehicles with the best known solutions.

- *The least total travelled distance with the best-known solutions.*

  NEDPALNS has 42 results similar to the best-known solutions which are equivalent to 75% of the total instances used for comparison. This shows that NEDPALNS has great achievement in the least number of used vehicles with the best-known solutions.

- *The least average total travelled distance with the best-known solutions*

  NEDPALNS has 52 instances that have similar or better results than the best-known solutions. This is equivalent to 93% of the 56 instances used for comparison, while the remaining 4 instances are non-dominating solutions. This shows that NEDPALNS achieved significant results in the least total travelled distance with the best-known solutions.

Second, we compare the Solomon and dynamic dataset. The comparison is listed as follows:

- *The overall average of the least rejection rates.*

  NEDPALNS has lower rejection rates in the overall average of the least rejection rates. This shows that NEDPALNS can accommodate more customer requests.

- *The least average on the number of used vehicles and the least average on total travelled distance.*

  Overall, NEDPALNS recorded the least average number of used vehicles and the least average total travelled distance.

Third, we compare the MOVRPTW and dynamic dataset. The comparisons are listed as follows:

- *The overall average on the least number of used vehicles and the least total travelled distance.*

  NEDPALNS outperforms ALNS in the overall average of the least number of used vehicles and the least number of total travelled distance.

- *The overall average of the best, worst, and average on the least average number of used vehicles and the least average total travelled distance.*

  NEDPALNS has better results an overall average of the best, worst, and average on the least average number of used vehicles and the least average total travelled distance.

In summary, NEDPALNS outperform other published algorithms using static test using hypothetical datasets and dynamic test using hypothetical datasets and real datasets although, in some instances, NEDPALNS is underperforming. overall, NEDPALNS shows better least average rejection rates using Solomon datasets, least average of the best, worst and average rejection rates using Solomon and dynamic datasets, and the least average of the best, worst, and the average number of used vehicles and total travelled distance using MOVRPTW and dynamic datasets.

5.2. Limitations and Opportunities for Future Improvement

So far, NEDPALNS is testing on MODVRPTW. There are opportunities and limitations that NEDPALNS can showcase. They are listed as follows:

First, there are many unique and interesting variants of VRP that can be tested. Some can be dynamic vehicle routing problems with pickup and delivery, multi-time windows dynamic vehicle routing problems among others.

Second, the customer size used for testing in these experiments is small to medium. This experiment can be extended to 10,000 customers or even millions of customers. With this experiment, we can measure the threshold limit of how much distributed computing should have and whether it can be supported and maintained in an enterprise organization that supports a huge customer size.

Third, the current trend in artificial intelligence is to apply the algorithm in drones to deliver food or packages to customers. The experiment can be carried out to gauge the effectiveness and efficiency of the NEDPALNS should it be implemented in the drones.

LIST OF REFERENCES

Adulbhan P, M. T. T. (1980) 'Chapter 9, Decision Models for Industrial Systems Engineers and Managers', in *Multicriterion Optimization in Industrial Systems*. Bangkok.

Afsar, H. M., Afsar, S. and Palacios, J. J. (2021) 'Vehicle routing problem with zone-based pricing', *Transportation Research Part E: Logistics and Transportation Review*, 152, p. 102383. doi: https://doi.org/10.1016/j.tre.2021.102383.

Ahmmed, A. *et al.* (2008) 'A Multiple Ant Colony System for Dynamic Vehicle Routing Problem with Time Window', in *2008 Third International Conference on Convergence and Hybrid Information Technology*, pp. 182–187. doi: 10.1109/ICCIT.2008.249.

Altabeeb, A. M. *et al.* (2021) 'Solving capacitated vehicle routing problem using cooperative firefly algorithm', *Applied Soft Computing*. Elsevier B.V., 108, p. 107403. doi: 10.1016/j.asoc.2021.107403.

Andersson, J. and Krus, P. (2001a) 'Metamodel Representations for Robustness Assessment in Multiobjective Optimization', in *The International Conference on Engineering Design ICED 01,2001*. Glasgow, UK.

Andersson, J. and Krus, P. (2001b) 'Multiobjective optimization of mixed variable design problems', in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 624–638.

de Armas, J. and Melián-Batista, B. (2015) 'Variable Neighborhood Search for a Dynamic Rich Vehicle Routing Problem with time windows', *Computers & Industrial Engineering*, 85, pp. 120–131. doi: https://doi.org/10.1016/j.cie.2015.03.006.

Attanasio, A. *et al.* (2004) 'Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem', *Parallel Computing*, 30(3), pp. 377–387. doi: https://doi.org/10.1016/j.parco.2003.12.001.

Baldacci, R. *et al.* (2010) 'An exact solution framework for a broad class of vehicle routing problems', *Computational Management Science*, 7(3), pp. 229–268. doi: 10.1007/s10287-009-0118-3.

Baños, R. *et al.* (2013) 'A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows', *Computers & Industrial Engineering*, 65, pp. 286–296. doi: 10.1016/j.cie.2013.01.007.

Barceló, J., Grzybowska, H. and Pardo, S. (2007) 'Vehicle routing and scheduling models, simulation and city logistics', in *Dynamic Fleet Management*. Springer, pp. 163–195.

Beaudry, A. *et al.* (2010) ‘Dynamic transportation of patients in hospitals’, *OR spectrum*. Springer, 32(1), pp. 77–107.

Bent, R. and Hentenryck, P. Van (2004) ‘A two-stage hybrid local search for the vehicle routing problem with time windows’, *Transportation Science*, 38(4), pp. 515–530. doi: 10.1287/trsc.1030.0049.

Bent, R. W. and Van Hentenryck, P. (2004) ‘Scenario-based planning for partially dynamic vehicle routing with stochastic customers’, *Operations Research*. INFORMS, 52(6), pp. 977–987.

Berger, J., Barkaoui, M. and Bräysy, O. (2003) ‘A route-directed hybrid genetic approach for the vehicle Routing problem with time windows’, *Information Systems and Operational Research*, 41(2), pp. 179–194. doi: 10.1080/03155986.2003.11732675.

Bertsimas, D. (1988) *Probabilistic combinatorial optimization problems*. Massachusetts Institute of Technology.

Bertsimas, D. J. and Simchi-Levi, D. (1996) ‘A new generation of vehicle routing research: robust algorithms, addressing uncertainty’, *Operations research*. INFORMS, 44(2), pp. 286–304.

Bouhmala, N. (2019) ‘Combining simulated annealing with local search heuristic for MAX-SAT’, *Journal of Heuristics*. Springer, 25(1), pp. 47–69.
Bouthilliera, A. Le and Crainic, T. G. (2005) ‘A cooperative parallel meta-heuristic for the vehicle routing problem with time windows’, *Computers & Operations Research*, 32(7), pp. 1685–1708. doi: 10.1016/j.cor.2003.11.023.

Castro-Gutierrez, J. (2012) *Multi-objective tools for the vehicle routing problem with time windows*. University of Nottingham. Available at: http://eprints.nottingham.ac.uk/13713/1/thesis.pdf.

Castro-Gutierrez, J., Landa-Silva, D. and Pérez, J. M. (2011) ‘Nature of real-world multi-objective vehicle routing with evolutionary algorithms’, in *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 257–264.

Chen, S. *et al.* (2018) ‘An adaptive large neighborhood search heuristic for dynamic vehicle routing problems’, *Computers & Electrical Engineering*, 67, pp. 596–607. doi: https://doi.org/10.1016/j.compeleceng.2018.02.049.

Chen, Z.-L. and Xu, H. (2006) ‘Dynamic column generation for dynamic vehicle routing with time windows’, *Transportation Science*. INFORMS, 40(1), pp. 74–88.

Cheng, S., Zhan, H. and Shu, Z. (2016) ‘An innovative hybrid multi-objective particle swarm optimization with or without constraints handling’, *Applied Soft Computing Journal*. Elsevier Ltd, 47, pp. 370–388. doi: 10.1016/j.asoc.2016.06.012.

Cheung, B. K.-S. *et al.* (2008) ‘Dynamic routing model and solution methods for fleet management with mobile technologies’, *International Journal of Production Economics*. Elsevier, 113(2), pp. 694–705.

Christiansen, C. H. and Lysgaard, J. (2007) ‘A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands’, *Operations Research Letters*. Elsevier, 35(6), pp. 773–781.

Coello, C. A. C. C. and Pulido, G. T. (2001) ‘A micro-genetic algorithm for multiobjective optimization’, in *International conference on evolutionary multi-criterion optimization*, pp. 126–140.

Coello, C. A. C., Lamont, G. B. and Veldhuizen, D. A. Van (2006) *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Berlin, Heidelberg: Springer-Verlag.

Coello, C. A. C. and Pulido, G. T. (2005) ‘Multiobjective structural optimization using a microgenetic algorithm’, *Structural and Multidisciplinary Optimization*. Springer, 30(5), pp. 388–403.

Coello, C. A. and Pulido, G. T. (2001) ‘Multiobjective optimization using a micro-genetic algorithm’, in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pp. 274–282.

Cordeau, J.-F., Laporte, G. and Mercier, A. (2001) ‘A unified tabu search algorithm for vehicle routing problems with soft time windows’, *Journal of the Operational Research Society*, 52(8), pp. 928–936. doi: 10.1057/palgrave.jors.2602371.

Corne, D. W., Knowles, J. D. and Oates, M. J. (2000) ‘The Pareto envelope-based selection algorithm for multiobjective optimization’, in *International conference on parallel problem solving from nature*, pp. 839–848.

Cremene, M. *et al.* (2016) ‘Comparative analysis of multi-objective evolutionary algorithms for QoS-aware web service composition’, *Applied Soft Computing Journal*. Elsevier Ltd, 39, pp. 124–139. doi: 10.1016/j.asoc.2015.11.012.

Curtois, T. *et al.* (2018) *Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows*, *EURO Journal on Transportation and Logistics*. doi: 10.1007/s13676-017-0115-6.

Czech, Z. J. and Czarnas, P. (2002) ‘Parallel simulated annealing for the vehicle routing problem with time windows’, in *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pp. 376–383.

Deb, K. (1991) *Binary and Floating-Point Function Optimization Using Messy Genetic Algorithms*. University of Alabama.

Deb, K. (2001) ‘Nonlinear goal programming using multi-objective genetic algorithms’, *Journal of the Operational Research Society*, 52(3), pp. 291–302. doi: 10.1057/palgrave.jors.2601089.

Ding, L. *et al.* (2003) ‘A new multiobjective evolutionary algorithm: OMOEA’, in *The 2003 Congress on Evolutionary Computation, 2003. CEC’03.*, pp. 898–905.

Dong, W. *et al.* (2018) ‘A tissue P system based evolutionary algorithm for multi-objective VRPTW’, *Swarm and Evolutionary Computation*. Elsevier, 39(December 2016), pp. 310–322. doi: 10.1016/j.swevo.2017.11.001.

Dror, M., Laporte, G. and Trudeau, P. (1989) ‘Vehicle routing with stochastic demands: Properties and solution frameworks’, *Transportation science*. INFORMS, 23(3), pp. 166–176.

Ehrgott, M. (2005) *Multicriteria Optimization*. second edi. Springer Berlin Heidelberg.

G.B. Dantzig, J. H. R. (1959) ‘The truck dispatching problem’, *Management Science*, 6(1), pp. 80–91.

Gambardella, L. M., Taillard, É. and Agazzi, G. (1999) ‘New Ideas in Optimization’. McGraw-Hill, London.

Gendreau, M. *et al.* (1999) ‘Parallel tabu search for real-time vehicle routing and dispatching’, *Transportation science*. INFORMS, 33(4), pp. 381–390.

Gendreau, M., Laporte, G. and Potvin, J.-Y. (2002) ‘Metaheuristics for the capacitated VRP’, in *The vehicle routing problem*. SIAM, pp. 129–154.

Gendreau, M., Laporte, G. and Séguin, R. (1996) ‘Stochastic vehicle routing’, *European Journal of Operational Research*, 88(1), pp. 3–12. doi: https://doi.org/10.1016/0377-2217(95)00050-X.

Gendreau, M. and Potvin, J.-Y. (1998) ‘Dynamic vehicle routing and dispatching’, in *Fleet management and logistics*. Springer, pp. 115–126.

Ghannadpour, S. F. *et al.* (2014) ‘A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application’, *Applied Soft Computing Journal*. Elsevier B.V., 14(PART C), pp. 504–527. doi: 10.1016/j.asoc.2013.08.015.

Ghiani, G. *et al.* (2003) ‘Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies’, *European journal of operational research*. Elsevier, 151(1), pp. 1–11.

Ghoseiri, K. and Farid, S. (2010) ‘Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm’, *Applied Soft Computing Journal*, 10(4), pp. 1096–1107. doi: 10.1016/j.asoc.2010.04.001.

Goel, A. and Gruhn, V. (2008) 'A general vehicle routing problem', *European Journal of Operational Research*. Elsevier, 191(3), pp. 650–660.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st edn. USA: Addison-Wesley Longman Publishing Co., Inc.

Goldberg, D. E. and Deb, K. (1991) 'A Comparative Analysis of Selection Schemes Used in Genetic Algorithms', in RAWLINS, G. J. E. (ed.). Elsevier (Foundations of Genetic Algorithms), pp. 69–93. doi: https://doi.org/10.1016/B978-0-08-050684-5.50008-2.

Goldberg, D. E. and Holland, J. H. (1988) 'Genetic algorithms and machine learning'. Kluwer Academic Publishers-Plenum Publishers; Kluwer Academic Publishers~….

Grueninger, T. and Wallace, D. (1996) *Multimodal Optimization Using Genetic Algorithms*, *Handbook of Genetic Algorithms*. Cambridge, Massachusetts, USA. Available at: http://ci.nii.ac.jp/naid/10000000876/.

Güner, A. R., Murat, A. and Chinnam, R. B. (2012) 'Dynamic routing under recurrent and non-recurrent congestion using real-time ITS information', *Computers & Operations Research*. Elsevier, 39(2), pp. 358–373.

Haghani, A. and Jung, S. (2005) 'A dynamic vehicle routing problem with time-dependent travel times', *Computers \& operations research*. Elsevier, 32(11), pp. 2959–2986.

Hansen, P., Mladenović, N. and Pérez, J. A. M. (2010) 'Variable neighbourhood search: methods and applications', *Annals of Operations Research*. Springer, 175(1), pp. 367–407.

Holborn, P. L., Thompson, J. M. and Lewis, R. (2012) 'Combining heuristic and exact methods to solve the vehicle routing problem with pickups, deliveries and time windows', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7245 LNCS(2), pp. 63–74. doi: 10.1007/978-3-642-29124-1_6.

Holland, J. H. (1992) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press.

Homberger, J. (2000) *Verteilt-parallele Metaheuristiken zur Tourenplanung*. doi: 10.1007/978-3-322-97815-8.

Homberger, J. and Hermann, G. (1999) 'Two evolutionary metaheuristics for the vehicle routing problem with time windows', *Information Systems and Operational Research*, 37(3), pp. 297–318. doi: 10.1080/03155986.1999.11732386.

Hong, L. (2012) ‘An improved LNS algorithm for real-time vehicle routing problem with time windows’, *Computers & Operations Research*, 39(2), pp. 151–163. doi: https://doi.org/10.1016/j.cor.2011.03.006.

Horn, J., Nafpliotis, N. and Goldberg, D. (1994) ‘Multiobjective Optimization Using The Niche Pareto Genetic Algorithm’.

HTEC (no date) *What Are The Trends In Transportation In 2021? Title*. Available at: https://htecgroup.com/insights/industry-insights/what-are-the-trends-in-transportation-in-2021/.

Huang, N. *et al.* (2021) ‘The multi-trip vehicle routing problem with time windows and unloading queue at depot’, *Transportation Research Part E: Logistics and Transportation Review*, 152, p. 102370. doi: https://doi.org/10.1016/j.tre.2021.102370.

Hvattum, L. M., Løkketangen, A. and Laporte, G. (2006) ‘Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic’, *Transportation Science*. INFORMS, 40(4), pp. 421–438.

Hwang, C. L. *et al.* (1980) ‘Mathematical programming with multiple objectives: A tutorial’, *Computers & Operations Research*, 7(1), pp. 5–31. doi: https://doi.org/10.1016/0305-0548(80)90011-8.

Ibaraki, T. *et al.* (2005) ‘Effective local search algorithms for routing and scheduling problems with general time-window constraints’, *Transportation Science*, 39(2), pp. 206–232. doi: 10.1287/trsc.1030.0085.

Ichoua, S., Gendreau, M. and Potvin, J.-Y. (2000) ‘Diversion issues in real-time vehicle dispatching’, *Transportation science*. INFORMS, 34(4), pp. 426–438.

Ichoua, S., Gendreau, M. and Potvin, J.-Y. (2007) ‘Planned route optimization for real-time vehicle routing’, *Dynamic Fleet Management*. Springer, pp. 1–18.
Intelligence, T. (2021) *Global logistics market forecast to grow by a CAGR of 4.7% to 2024*. Available at: https://www.hellenicshippingnews.com/global-logistics-market-forecast-to-grow-by-a-cagr-of-4-7-to-2024/.

Ishibuchi, H *et al.* (2015) ‘Difficulties in specifying reference points to calculate the inverted generational distance for many-objective optimization problems’, in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - MCDM 2014: 2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, Proceedings*. Institute of Electrical and Electronics Engineers Inc., pp. 170–177. doi: 10.1109/MCDM.2014.7007204.

Ishibuchi, Hisao *et al.* (2015) ‘Modified distance calculation in generational distance and inverted generational distance’, in *International conference on evolutionary multi-criterion optimization*, pp. 110–125.

Ishibuchi, H. and Murata, T. (1998) ‘A multi-objective genetic local search algorithm and its application to flowshop scheduling', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(3), pp. 392–403. doi: 10.1109/5326.704576.

Jacobsen-Grocott, J. *et al.* (2017) 'Evolving heuristics for Dynamic Vehicle Routing with Time Windows using genetic programming', in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1948–1955. doi: 10.1109/CEC.2017.7969539.

Jaillet, P. and Wagner, M. R. (2008) 'Online vehicle routing problems: A survey', in *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, pp. 221–237.

Kaiwartya, O., Kumar, S., Lobiyal, D. K., *et al.* (2015) 'Multiobjective dynamic vehicle routing problem and time seed based solution using particle swarm optimization', *Journal of Sensors*, 2015. doi: 10.1155/2015/189832.

Kaiwartya, O., Kumar, S., Lobiyal, D K, *et al.* (2015) 'Multiobjective Dynamic Vehicle Routing Problem and Time Seed Based Solution Using Particle Swarm Optimization', *Journal of Sensors*. Edited by T. Zhu. Hindawi Publishing Corporation, 2015, p. 189832. doi: 10.1155/2015/189832.

Kallehauge, B. and Solomon, M. M. (2005) *Vehicle routing problem with time windows*.

Keeney, R. L. and Raiffa, H. (1993) *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Cambridge University Press. doi: 10.1017/CBO9781139174084.

Kenyon, A. S. and Morton, D. P. (2003) 'Stochastic vehicle routing with random travel times', *Transportation Science*. INFORMS, 37(1), pp. 69–82.
Kilby, P., Prosser, P. and Shaw, P. (1998) 'Dynamic VRPs: A study of scenarios', *University of Strathclyde Technical Report*, 1(11).

Knowles, J. D. and Corne, D. W. (2000) 'Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy', *Evolutionary Computation*, 8(2), pp. 149–172. doi: 10.1162/106365600568167.

Kritikos, M. N. and Ioannou, G. (2010) 'The balanced cargo vehicle routing problem with time windows', *International Journal of Production Economics*, pp. 42–51. doi: 10.1016/j.ijpe.2009.07.006.

Kyriakakis, N. A., Marinaki, M. and Marinakis, Y. (2021) 'A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem', *Computers & Operations Research*, 134, p. 105397. doi: https://doi.org/10.1016/j.cor.2021.105397.

Lackner, A. (2004) 'Selection meta-heuristics for dynamic vehicle routing problem (Dynamische Tourenplanung mit ausgewählten Metaheuristiken)', *Göttinger Wirtschaftsinformatik, Herausgeber*, 47.

Laporte, G., Louveaux, F. and Mercure, H. (1992) 'The vehicle routing problem with stochastic travel times', *Transportation science*. INFORMS, 26(3), pp. 161–170.

Larsen, A. (2000) *The dynamic vehicle routing problem*. Institute of Mathematical Modelling, Technical University of Denmark.

Larsen, A., Madsen, O. B. G. and Solomon, M. M. (2004) 'The a priori dynamic traveling salesman problem with time windows', *Transportation Science*. INFORMS, 38(4), pp. 459–472.

Lawrence J. Fogel, Alvin J. Owens, M. J. W. (1966) *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, Inc.

Li, H. and Lim, A. (2003) 'Local search with annealing-like restarts to solve the VRPTW', *European Journal of Operational Research*, 150(1), pp. 115–127. doi: 10.1016/S0377-2217(02)00486-1.

Li, J.-Q., Mirchandani, P. B. and Borenstein, D. (2009) 'A Lagrangian heuristic for the real-time vehicle rescheduling problem', *Transportation Research Part E: Logistics and Transportation Review*. Elsevier, 45(3), pp. 419–433.

Li, M., Yang, S. and Liu, X. (2014) 'Diversity comparison of Pareto front approximations in many-objective optimization', *IEEE Transactions on Cybernetics*. IEEE, 44(12), pp. 2568–2584.

Lorini, S., Potvin, J.-Y. and Zufferey, N. (2011) 'Online vehicle routing and scheduling with dynamic travel times', *Computers & Operations Research*. Elsevier, 38(7), pp. 1086–1090.

Loughlin, D. H. and Ranjithan, S. R. (1997) 'The Neighborhood Constraint Method: A Genetic Algorithm-Based Multiobjective Optimization Technique.', in *ICGA*, pp. 666–673.

Lund, K., Madsen, O. B. G. and Rygaard, J. M. (1996) *Vehicle routing problems with varying degrees of dynamism*. IMM, Institute of Mathematical Modelling, Technical University of Denmark.

Martin, O., Otto, S. W. and Felten, E. W. (1991) *Large-step Markov chains for the traveling salesman problem*. Citeseer.

Mendoza, J. E. *et al.* (2011) 'Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands', *Transportation science*. INFORMS, 45(3), pp. 346–363.

Mendoza, J. E., Medaglia, A. L. and Velasco, N. (2009) 'An evolutionary-based decision support system for vehicle routing: The case of a public utility', *Decision Support Systems*. Elsevier, 46(3), pp. 730–742.

Merz, P. and Huhse, J. (2008) 'An iterated local search approach for finding provably good solutions for very large TSP instances', in *International Conference on Parallel Problem Solving from Nature*, pp. 929–939.

Mester, D. (2002) 'An evolutionary strategies algorithm for large scale vehicle routing problem with capacitate and time windows restrictions', in *Proceedings of the Conference on Mathematical and Population Genetics, University of Haifa, Israel*.

Mihajlović, I., Živković, \DJ and Štrbac, N. (2007) 'Using genetic algorithms to resolve facility layout problem', *Serbian Journal of Management*, 2(1), pp. 35–46.

Montemanni, R. *et al.* (2005) 'Ant colony system for a dynamic vehicle routing problem', *Journal of combinatorial optimization*. Springer, 10(4), pp. 327–343.
Mu, Q. *et al.* (2011) 'Disruption management of the vehicle routing problem with vehicle breakdown', *Journal of the Operational Research Society*. Springer, 62(4), pp. 742–749.

Mühlenbein, H. and Paaß, G. (1996) 'From recombination of genes to the estimation of distributions I. Binary parameters', in. Springer Berlin Heidelberg, pp. 178–187. doi: 10.1007/3-540-61723-X_982.

Najera, A. G. (2010) *Multi-objective evolutionary algorithms for vehicle routing problems*. The University of Birmingham.

Necula, R., Breaban, M. and Raschip, M. (2017) 'Tackling Dynamic Vehicle Routing Problem with Time Windows by means of ant colony system', *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, pp. 2480–2487. doi: 10.1109/CEC.2017.7969606.

Novoa, C. and Storer, R. (2009) 'An approximate dynamic programming approach for the vehicle routing problem with stochastic demands', *European journal of operational research*. Elsevier, 196(2), pp. 509–515.

Pillac, V. *et al.* (2013) 'A review of dynamic vehicle routing problems', *European Journal of Operational Research*, 225(1), pp. 1–11. doi: https://doi.org/10.1016/j.ejor.2012.08.015.

Pillac, V., Gueret, C. and Medaglia, A. L. (2013) 'A parallel matheuristic for the technician routing and scheduling problem', *Optimization Letters*. Springer, 7(7), pp. 1525–1535.

Pisinger, D. and Ropke, S. (2007) 'A general heuristic for vehicle routing problems', *Computers & Operations Research*, 34(8), pp. 2403–2435. doi: https://doi.org/10.1016/j.cor.2005.09.012.

Potvin, J.-Y. and Rousseau, J.-M. (1993) 'A parallel route building algorithm for the vehicle routing and scheduling problem with time windows', *European Journal of Operational Research*, 66(3), pp. 331–340. doi: https://doi.org/10.1016/0377-2217(93)90221-8.

Powell, W. B. (2007) *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley \& Sons.

Psaraftis, H. N. (1980a) 'A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem', *Transportation Science*, 14(2), pp. 130–154. doi: 10.1287/trsc.14.2.130.

Psaraftis, H. N. (1980b) 'A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem', *Transportation Science*, 14(2), pp. 130–154. doi: 10.1109/SMC.2016.7844483.

Qi, Y. *et al.* (2015a) 'A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows', *Computers and Operations Research*, 62, pp. 61–77. doi: 10.1016/j.cor.2015.04.009.

Qi, Y. *et al.* (2015b) 'A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows', *Computers & Operations Research*, pp. 61–77. doi: 10.1016/j.cor.2015.04.009.

Ravber, M., Mernik, M. and Črepinšek, M. (2017) 'The impact of Quality Indicators on the rating of Multi-objective Evolutionary Algorithms', *Applied Soft Computing*, 55, pp. 265–275. doi: https://doi.org/10.1016/j.asoc.2017.01.038.

Rechenberg, I. (1965) *Cybernetic solution path of an experimental problem*. Royal Aircraft Establishment, Library Translation 1122, Farnborough, UK.

Repoussis, P. P., Tarantilis, C. D. and Ioannou, G. (2009) 'Arc-guided evolutionary algorithm for the vehicle routing problem with time windows', *IEEE Transactions on Evolutionary Computation*, 13(3), pp. 624–647. doi: 10.1109/TEVC.2008.2011740.

Rizzoli, A. E. *et al.* (2007) 'Ant colony optimization for real-world vehicle routing problems', *Swarm Intelligence*. Springer, 1(2), pp. 135–151.

Rochat, Y. and Taillard, É. D. (1995) 'Probabilistic Diversification and Intensification in Local Search for Vehicle Routing', *Journal of Heuristics*, 1, pp. 147–167. doi: doi.org/10.1007/BF02430370.

Rodrigue, J.-P. (2020) *The Geography of Transport Systems FIFTH EDITION*. Routledge. Available at: https://transportgeography.org/contents/chapter3/transport-costs/logistic_costs_breakdown/.

Ropke, S. and Pisinger, D. (2006) 'An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows', *Transportation Science*, 40(4), pp. 455–472. doi: 10.1287/trsc.1050.0135.

Rosenberg, R. S. (1970) 'Simulation of genetic populations with biochemical properties: II. Selection of crossover probabilities', *Mathematical Biosciences*,

8(1), pp. 1–37. doi: https://doi.org/10.1016/0025-5564(70)90140-9.
Rousseau, L. M., Gendreau, M. and Pesant, G. (2002) 'Using constraint-based operators to solve the vehicle routing problem with time windows', *Journal of Heuristics*, 8(1), pp. 43–58.

Saint-Guillain, M., Deville, Y. and Solnon, C. (2015) 'A multistage stochastic programming approach to the dynamic and stochastic VRPTW', in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 357–374.

Sartori, C. S. (2016) *Optimizing Solutions for the Pickup and Delivery Problem*. UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL.

Schaffer, J. D. (1985) 'Multiple Objective Optimization with Vector Evaluated Genetic Algorithms', in *Proceedings of the 1st International Conference on Genetic Algorithms*. USA: L. Erlbaum Associates Inc., pp. 93–100.

Schneider, M. (2016) 'The vehicle-routing problem with time windows and driver-specific times', *European Journal of Operational Research*, 250(1), pp. 101–119. doi: https://doi.org/10.1016/j.ejor.2015.09.015.

Schott, J. R. (1995) *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Massachusetts Institute of Technology.

Schrimpf, G. *et al.* (2000a) 'Record breaking optimization results using the ruin and recreate principle', *Journal of Computational Physics*. Elsevier, 159(2), pp. 139–171.

Schrimpf, G. *et al.* (2000b) 'Record Breaking Optimization Results Using the Ruin and Recreate Principle', *Journal of Computational Physics*, 159(2), pp. 139–171. doi: 10.1006/jcph.1999.6413.

Secomandi, N. (2000) 'Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands', *Computers & Operations Research*. Elsevier, 27(11–12), pp. 1201–1225.

Secomandi, N. and Margot, F. (2009) 'Reoptimization approaches for the vehicle-routing problem with stochastic demands', *Operations research*. INFORMS, 57(1), pp. 214–230.

Shaw, P. (1997) 'A new local search algorithm providing high quality solutions to vehicle routing problems', *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, pp. 1–12.

Shaw, P. (1998) 'Using constraint programming and local search methods to solve vehicle routing problems', in *International conference on principles and practice of constraint programming*, pp. 417–431.

Sheikh, V. *et al.* (2021) 'Land use optimization through bridging multiobjective optimization and multicriteria decision-making models (case study: Tilabad

Watershed, Golestan Province, Iran)', *Natural Resource Modeling*, 34(2), p. e12301. doi: https://doi.org/10.1111/nrm.12301.

Solomon, M. M. (1987) 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations research*. Informs, 35(2), pp. 254–265.

Srinivas, N. and Deb, K. (1994) 'Muiltiobjective optimization using nondominated sorting in genetic algorithms', *Evolutionary computation*. MIT Press, 2(3), pp. 221–248.

Storn, R. (1996) 'On the usage of differential evolution for function optimization', in *Proceedings of North American Fuzzy Information Processing*. IEEE, pp. 519–523. doi: 10.1109/NAFIPS.1996.534789.

Storn, R. and Price, K. (1997) 'Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces', *Journal of Global Optimization*, 11(4), pp. 341–359. doi: 10.1023/A:1008202821328.

Stützle, T. (2006) 'Iterated local search for the quadratic assignment problem', *European Journal of Operational Research*. Elsevier, 174(3), pp. 1519–1539.

Tagmouti, M., Gendreau, M. and Potvin, J.-Y. (2011) 'A dynamic capacitated arc routing problem with time-dependent service costs', *Transportation Research Part C: Emerging Technologies*. Elsevier, 19(1), pp. 20–28.

Taillard, E. *et al.* (1997) 'A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows', *Tranportation Science*. Transportation Science, 31(2), pp. 170–186.

Talbi, E.-G. (2007) 'Metaheuristics: From Design to Implementation', in *Metaheuristics: From Design to Implementation*. 1st edn. John Wiley & Sons, Inc, p. 18.

Tan, K. C., Chew, Y. H. and Lee, L. H. (2006) 'A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows', *Computational Optimization and Applications*, 34(1), pp. 115–151. doi: 10.1007/s10589-005-3070-3.

Tang, H. and Hu, M. (2005) 'Dynamic Vehicle Routing Problem with Multiple Objectives', *Transportation Research Record: Journal of the Transportation Research Board*, 1923(1), pp. 199–207. doi: 10.1177/0361198105192300121.

Thomas, B. W. (2007) 'Waiting strategies for anticipating service requests from known customer locations', *Transportation Science*. Informs, 41(3), pp. 319–331.

Ursani, Z. *et al.* (2011) 'Localized genetic algorithm for vehicle routing problem with time windows', *Applied Soft Computing Journal*. Elsevier B.V., 11(8), pp. 5375–5390. doi: 10.1016/j.asoc.2011.05.021.

Utama, D. M. *et al.* (2020) 'The vehicle routing problem for perishable goods: A systematic review', *Cogent Engineering*. Edited by D. Pham. Cogent OA, 7(1), p. 1816148. doi: 10.1080/23311916.2020.1816148.

Van Veldhuizen, D. A. and Lamont, G. B. (2000) 'Multiobjective optimization with messy genetic algorithms', in *Proceedings of the 2000 ACM symposium on Applied computing-Volume 1*, pp. 470–476.

Veldhuizen, D. A. Van and Veldhuizen, D. A. Van (1999) *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*.

Vent, W. (1975) 'Rechenberg, Ingo, Evolutionsstrategie — Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 170 S. mit 36 Abb. Frommann-Holzboog-Verlag. Stuttgart 1973. Broschiert', *Feddes Repertorium*, 86(5), p. 337. doi: https://doi.org/10.1002/fedr.19750860506.

Verweij, B. *et al.* (2003) 'The sample average approximation method applied to stochastic routing problems: a computational study', *Computational optimization and applications*. Springer, 24(2), pp. 289–333.

Waters, C. D. J. (1989) 'Vehicle-scheduling problems with uncertainty and omitted customers', *Journal of the Operational Research Society*. Taylor & Francis, 40(12), pp. 1099–1108.

Wilson, N. H. M. and Colvin, N. J. (1977) *Computer control of the Rochester dial-a-ride system*. Massachusetts Institute of Technology, Center for Transportation Studies.

Xu, S.-H. *et al.* (2015) 'A Combination of Genetic Algorithm and Particle Swarm Optimization for Vehicle Routing Problem with Time Windows', *Sensors*, 15(9), pp. 21033–21053. doi: 10.3390/s150921033.

Yang, J., Jaillet, P. and Mahmassani, H. (2004) 'Real-time multivehicle truckload pickup and delivery problems', *Transportation Science*. INFORMS, 38(2), pp. 135–148.

Yu, V. F. *et al.* (2017) 'Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem', *Applied Soft Computing Journal*. Elsevier B.V., 52, pp. 657–672. doi: 10.1016/j.asoc.2016.10.006.

Yu, Xiaobing, Lu, Y. and Yu, Xianrui (2018) 'Evaluating Multiobjective Evolutionary Algorithms Using MCDM Methods', *Mathematical Problems in Engineering*. Edited by D. Bigaud. Hindawi, 2018, p. 9751783. doi: 10.1155/2018/9751783.

Zeimpekis, V. S. *et al.* (2007) *Dynamic fleet management: concepts, systems, algorithms \& case studies*. Springer Science \& Business Media.

Zeng, S. *et al.* (2005) 'An efficient multi-objective evolutionary algorithm:

OMOEA-II', in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 108–119.

Zhang, J., Yang, F. and Weng, X. U. N. (2018) 'An Evolutionary Scatter Search Particle Swarm Optimization Algorithm for the Vehicle Routing Problem With Time Windows', *IEEE Access*. IEEE, 6, pp. 63468–63485. doi: 10.1109/ACCESS.2018.2877767.

Zhang, W. *et al.* (2020) 'Hybrid multiobjective evolutionary algorithm with fast sampling strategy-based global search and route sequence difference-based local search for VRPTW', *Expert Systems with Applications*. Elsevier Ltd, 145, p. 113151. doi: 10.1016/j.eswa.2019.113151.

Zitzler, E. *et al.* (2003) 'Performance assessment of multiobjective optimizers: An analysis and review', *IEEE Transactions on Evolutionary Computation*, 7(2), pp. 117–132. doi: 10.1109/TEVC.2003.810758.

Zitzler, E. and Thiele, L. (1998) 'Multiobjective optimization using evolutionary algorithms—a comparative case study', in *International conference on parallel problem solving from nature*, pp. 292–301.

APPENDICES

Appendix A: List of Tables for Comparisons with other Algorithms

Appendix A1: Comparison with other published algorithms based on instance type C1

Instance	M-MOEA/D (Qi *et al.*, 2015a)		MOGPGA (Ghoseiri and Farid, 2010)		MOEA (Najera, 2010)		NEDPALNS	
	NV	**TD**	**NV**	**TD**	**NV**	**NV**	**TD**	**NV**
C101	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**
C102	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**
C103	10	**828.06**	10	**828.06**	10	**828.06**	10	**828.06**
C104	10	**824.78**	10	**824.78**	10	**824.78**	10	**824.78**
C105	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**
C106	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**
C107	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**
C108	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**
C109	10	**828.94**	10	**828.94**	10	**828.94**	10	829.71
Num Ns	9		9		9		8	

Appendix A2: Comparison with other published algorithms based on instance type C2

Instance	M-MOEA/D (Qi *et al.*, 2015a)		MOGPGA (Ghoseiri and Farid, 2010)		MOEA (Najera, 2010)		NEDPALNS	
	NV	**TD**	**NV**	**TD**	**NV**	**NV**	**TD**	**NV**
C201	3	**591.56**	3	**591.56**	3	**591.56**	3	**591.56**
C202	3	**591.56**	3	**591.56**	3	**591.56**	3	**591.56**
C203	3	**591.17**	3	**591.17**	3	**591.17**	3	**591.17**
C204	3	**590.6**	3	**599.96**	3	**590.6**	3	**590.6**
C205	3	**588.88**	3	**588.88**	3	**588.88**	3	**588.88**
C206	3	**588.49**	3	588.88	3	**588.49**	3	**588.49**
C207	3	**588.29**	3	591.56	3	**588.29**	3	**588.29**
C208	3	**588.32**	3	**588.32**	3	**588.32**	3	**588.32**
Num Ns	8		6		8		8	

Appendix A3: Comparison with other published algorithms based on instance type R1

Instance	M-MOEA/D (Qi *et al.*, 2015a)		MOGPGA (Ghoseiri and Farid, 2010)		MOEA (Najera, 2010)		NEDPALNS	
	NV	TD	NV	TD	NV	TD	NV	TD
R101	19	1652.17	19	1677	**19**	**1650.8**	**19**	**1650.8**
	20	1644.7	20	1651.1	**20**	**1642.88**	**20**	**1642.88**
	17	**1486.12**	N/A	N/A	**17**	**1486.12**	17	1494.15
R102	18	1473.73	18	1511.8	18	1474.19	**18**	**1472.81**
	N/A	N/A	**19**	**1494.7**	N/A	N/A	N/A	N/A
R103	13	1354.22	N/A	N/A	**13**	**1308.28**	13	1351.98

	14	**1213.62**	14	1287	14	1219.37	**14**	**1213.62**
	N/A	N/A	**15**	**1264.2**	N/A	N/A	N/A	N/A
R104	10	999.31	**10**	**974.24**	10	990.79	10	981.23
	11	991.91	N/A	N/A	11	984.56	**11**	**976.61**
R105	14	1410.64	N/A	N/A	**14**	**1377.11**	14	1377.33
	15	1366.58	15	1424.6	15	1364.91	**15**	**1360.78**
	N/A	N/A	**16**	**1382.5**	N/A	N/A	N/A	N/A
R106	12	1265.99	N/A	N/A	**12**	**1261.52**	12	1263.98
	13	1249.22	13	1270.3	13	1241.65	**13**	**1239.37**
R107	10	1139.47	N/A	N/A	10	1154.38	**10**	**1131.69**
	11	1086.22	11	1108.8	11	1083.3	**11**	**1072.12**
R108	N/A	N/A	N/A	N/A	9	984.75	**9**	**978.33**
	10	965.52	10	971.91	10	960.03	**10**	**938.2**
R109	12	1157.44	12	1212.3	12	1157.76	**12**	**1153.02**
	13	1155.38	N/A	N/A	13	1154.61	**13**	**1151.84**
	N/A	N/A	**14**	**1206.7**	N/A	N/A	N/A	N/A
R110	11	1110.68	N/A	N/A	11	1094.75	**11**	**1078.8**
	12	1106.03	12	1156.5	12	1088.61	**12**	**1072.41**
R111	N/A	N/A	N/A	N/A	N/A	N/A	**10**	**1123.36**
	11	1073.82	11	1111.9	11	1061.37	**11**	**1054.23**
	N/A	N/A	N/A	N/A	N/A	N/A	**12**	**1053.50**
R112	10	981.43	N/A	N/A	10	980.83	**10**	**958.03**
	N/A	N/A	11	1011.5	N/A	N/A	**11**	**967.32**
Num Ns	2		5		6		**20**	

Appendix A4: Comparison with other published algorithms based on instance type R2

Instance	**M-MOEA/D (Qi, Hou, Li, Huang, & Li, 2015b)**		**MOGPGA** (Ghoseiri and Farid, 2010)		**MOEA (Najera, 2010)**		**NEDPALNS**	
	NV	**TD**	**NV**	**TD**	**NV**	**TD**	**NV**	**TD**
R201	**4**	**1253.23**	4	1351.4	4	1254.77	4	1331.25
	5	1196.5	N/A	N/A	**5**	**1194.07**	**5**	**1194.07**
	6	1185.79	N/A	N/A	N/A	N/A	**6**	**1170.25**
	N/A	N/A	N/A	N/A	N/A	N/A	**7**	**1152.96**
	N/A	N/A	N/A	N/A	N/A	N/A	**8**	**1147.80**
R202	4	1081.82	4	1091.22	4	1087.29	**4**	**1079.39**
	5	1049.72	N/A	N/A	5	1050.41	**5**	**1041.1**
	N/A	N/A	N/A	N/A	N/A	N/A	**6**	**1034.97**
R203	**3**	**955.7**	3	1041	3	950.9	3	972.58
	4	904.46	N/A	N/A	4	912.24	**4**	**897.02**
	5	889.36	5	995.8	5	905.34	**5**	**880.82**
	N/A	N/A	6	978.5	N/A	N/A	**6**	**874.87**
R204	3	753.32	3	1130.1	3	752.83	**3**	**751.06**
	4	745.96	4	927.7	N/A	N/A	**4**	**737.06**
	5	743.29	5	831.8	N/A	N/A	**5**	**735.8**
	N/A	N/A	**6**	**826.2**	N/A	N/A	N/A	N/A
R205	**3**	**1017.96**	3	1422.3	3	1040.29	3	1064.71
	4	960.33	4	1087.8	4	968.09	**4**	**959.74**
	5	954.48	N/A	N/A	N/A	N/A	**5**	**954.16**
R206	**3**	**915.49**	3	940.12	3	930.58	3	942.08
	4	887.9	N/A	N/A	4	899.83	**4**	**887.7**
	N/A	N/A	N/A	N/A	N/A	N/A	**5**	**884.85**
R207	3	813.47	3	904.9	3	818.97	**3**	**811.51**
	4	809.51	N/A	N/A	N/A	N/A	**4**	**797.99**

R208	**2**	**728.63**	N/A	N/A	2	736.9	2	864.3
	3	711.59	3	774.18	3	712.98	**3**	**706.74**
	N/A	N/A	N/A	N/A	N/A	N/A	**4**	**705.33**
R209	**3**	**918.82**	N/A	N/A	N/A	N/A	3	1002.82
	4	867.47	4	1008	4	878.05	**4**	**862.67**
	N/A	N/A	N/A	N/A	N/A	N/A	**5**	**860.11**
R210	3	952.91	**3**	**938.58**	3	961.36	3	1038.78
	4	928.35	N/A	N/A	4	936.68	**4**	**920.3**
	5	920.06	N/A	N/A	N/A	N/A	**5**	**909.66**
	N/A	N/A	N/A	N/A	N/A	N/A	**6**	**905.21**
R211	**3**	**774.68**	3	1310.4	3	785.97	3	777.08
	4	767.1	4	1101.5	N/A	N/A	**4**	**753.15**
NumNs	7		2		1		**27**	

Appendix A5: Comparison with other published algorithms based on instance type RC1

Instance	M-MOEA/D (Qi *et al.*, 2015b)		MOGPGA (Ghoseiri and Farid, 2010)		MOEA (Najera, 2010)		NEDPALNS	
	NV	TD	NV	TD	NV	TD	TD	NV
RC101	14	1758.17	N/A	N/A	N/A	N/A	**14**	**1705.40**
	15	1646.81	15	1690.6	15	1625.26	**15**	**1623.58**
	16	**1646.65**	N/A	N/A	N/A	N/A	N/A	N/A
RC102	13	1509.18	N/A	N/A	13	1501.11	**13**	**1477.54**
	14	1484.89	14	1509.4	14	1480.26	**14**	**1461.23**
	15	**1484.48**	15	1493.2	N/A	N/A	N/A	N/A
RC103	11	1274.85	N/A	N/A	11	1278.19	**11**	**1261.67**
	N/A	N/A	**12**	**1331.8**	N/A	N/A	N/A	N/A
RC104	10	1145.79	N/A	N/A	10	1144.39	**10**	**1135.52**
	N/A	N/A	**11**	**1177.2**	N/A	N/A	N/A	N/A
RC105	14	1548.43	N/A	N/A	14	**1540.18**	**14**	**1540.18**
	15	1528.61	15	1611.5	15	1519.44	**15**	**1519.27**
	N/A	N/A	16	1589.4	N/A	N/A	**16**	**1518.58**
RC106	12	1447.84	N/A	N/A	12	1395.7	**12**	**1379.08**
	13	1399.17	13	1437.6	13	1379.68	**13**	**1376.99**
	N/A	N/A	**14**	**1425.3**	N/A	N/A	N/A	N/A
RC107	11	1254.67	11	**1222.1**	11	1234.49	11	1232.2
	12	1235.54	N/A	N/A	12	1215.06	**12**	**1212.83**
RC108	10	1183.85	N/A	N/A	10	1158.22	**10**	**1147.2**
	11	1138.95	11	1156.5	11	1122.98	**11**	**1118.07**
NumNs	2		4		1		**14**	

Appendix A6: Comparison with other published algorithms based on instance type RC2

Instance	M-MOEA/D (Qi *et al.*, 2015b)		MOGPGA (Ghoseiri and Farid, 2010)		MOEA (Najera, 2010)		NEDPALNS	
	NV	TD	NV	TD	NV	TD	NV	TD
RC201	**4**	**1421.88**	4	1423.7	4	1438.43	N/A	N/A
	5	**1316.61**	N/A	N/A	5	1329.26	5	1321.93
	6	1297.47	N/A	N/A	6	1316.25	**6**	**1284.12**
	7	1289.94	N/A	N/A	7	1299.58	**7**	**1269.94**
	N/A	N/A	N/A	N/A	N/A	N/A	**8**	**1266.38**

RC202	N/A	N/A	N/A	N/A	N/A	N/A	**9**	**1265.56**
	4	**1161.29**	4	1369.8	4	1165.57	4	1214.17
	5	**1118.66**	N/A	N/A	5	1120.15	**5**	**1118.66**
	N/A	N/A	N/A	N/A	N/A	N/A	**6**	**1110.4**
	N/A	N/A	N/A	N/A	N/A	N/A	**7**	**1098.86**
	N/A	N/A	N/A	N/A	N/A	N/A	**8**	**1095.64**
RC203	**3**	**1097.4**	N/A	N/A	N/A	N/A	N/A	N/A
	4	**944.5**	4	1060	4	954.51	4	947.95
	5	940.55	N/A	N/A	N/A	N/A	**5**	**926.82**
	N/A	N/A	**6**	**1020.1**	N/A	N/A	N/A	N/A
RC204	3	801.9	3	901.46	3	802.71	**3**	**798.46**
	4	792.98	N/A	N/A	4	792.84	**4**	**786.38**
RC205	4	1327.09	4	1410.3	**4**	**1318.71**	N/A	N/A
	5	**1245.94**	N/A	N/A	5	1259	5	1247.85
	6	1187.48	N/A	N/A	6	1214.49	**6**	**1177.58**
	N/A	N/A	N/A	N/A	7	1205.06	**7**	**1157.55**
RC206	3	1200.92	N/A	N/A	**3**	**1191.62**	N/A	N/A
	4	1092.7	4	1194.8	**4**	**1085.82**	4	1087.93
	5	1089.14	N/A	N/A	5	1077.48	**5**	**1063.53**
	N/A	N/A	N/A	N/A	N/A	N/A	**6**	**1056.21**
	N/A	N/A	N/A	N/A	N/A	N/A	**7**	**1054.61**
RC207	**3**	**1107.71**	N/A	N/A	3	1133.27	N/A	N/A
	4	1000.98	4	1040.6	4	1001.73	**4**	**996.94**
	5	987.88	N/A	N/A	5	1001.51	**5**	**970.78**
	N/A	N/A	N/A	N/A	N/A	N/A	**6**	**969.8**
RC208	3	841.37	3	898.5	3	844.96	**3**	**829**
	4	807.83	N/A	N/A	4	780.07	**4**	**778.93**
NumNs	8	1	3	**21**	NumNs	8	1	3

Appendix A7: Comparison with ALNS hypervolume based on C1 instance type and 10% DoD

Instance	**Proposed Algorithm**				**ALNS**			
	NV	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
C101	11	904.9	0	**0.4486**	11	906.2	0	0.4375
	12	893	0					
C102	12	969.2	0	**0.5253**	12	1085.3	0	0.4879
					13	1005.4	0	
C103	10	853.9	0	**0.3788**	10	858.3	0	0.3750
C104	10	1057.3	0	**0.5811**	11	1047	0	0.4504
	11	959.3	0		12	1040.9	0	
C105	10	827.3	0	**0.3756**	10	828	0	0.3750
C106	11	872.5	0	**0.4459**	11	963.6	0	0.3750
C107	10	827.3	0	**0.3753**	10	827.6	0	0.3750
C108	11	1005.5	0	**0.4988**	12	1081.3	0	0.3750
C109	11	937.1	0	**0.4472**	11	1036.9	0	0.3750
Count		**11**				**11**		
Average				**0.4530**				0.4029

Appendix A8: Comparison with ALNS hypervolume based on C2 instance type and 10% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
C201	3	589	1	**0.4928**	4	679.9	0	0.3750
	4	672.9	0					
C202	3	632.1	0	**0.3763**	3	632.1	0	0.3750
		628.9	1					
C203	4	645.5	0	0.3750	3	610.1	0	**0.6242**
C204	3	624.6	0	**0.7693**	4	765.3	0	0.3750
C205	3	586.4	0	**0.3764**	3	587.5	0	0.3750
C206	3	586	0	**0.3750**	3	586	0	**0.3750**
C207	3	694.7	0	**0.5629**	4	680.9	0	0.3899
	4	694.3	0					
C208	3	589.4	0	**0.3802**	3	593.5	0	0.3750
Count		**11**				8		
Average				**0.4635**				0.4080

Appendix A9: Comparison with ALNS hypervolume based on the R1 instance type and 10% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R101	23	1988.6	0	**0.4415**	24	1944	0	0.4368
	25	1962.3	2		25	1926.4	0	
		1986.5	0			1902.5	1	
		1957.1	3					
R102	19	1733.5	0	**0.4544**	20	1715.7	0	0.4191
	21	1715	0					
R103	16	1396.2	0	**0.4445**	17	1439.8	0	0.3750
R104	12	1141	0	0.3750	12	1095.1	0	**0.4052**
R105	17	1631.7	0	**0.4345**	17	1614.4	0	0.4440
	18	1592.9	0		18	1574.1	0	
R106	14	1420.7	0	**0.4263**	15	1408.2	0	0.3816
	15	1418.2	0					
R107	14	1310.4	0	**0.4250**	15	1289.1	0	0.3872
R108	12	1102.5	0	0.3750	12	1080.1	0	**0.3902**
R109	15	1357.6	0	**0.4455**	16	1396.7	0	0.3750
R110	14	1315.1	0	**0.4273**	14	1317.1	0	0.4250
	15	1313.4	0					
R111	14	1279.4	0	0.3750	14	1279.3	0	**0.3751**
R112	12	1145.2	0	**0.3804**	12	1153.5	0	0.3750
Count		**19**				15		
Average				**0.4170**				0.3991

Appendix A10: Comparison with ALNS's hypervolume based on the R2 instance type and 10% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R201	5	1447.8	0	**0.5913**	6	1390.9	0	0.5200
	7	1443.9	0					
R202	5	1253.2	0	**0.6469**	5	1330.2	0	0.5502
	6	1095.3	0		6	1241.2	0	
R203	5	950.7	0	**0.3905**	5	970.7	0	0.3750
R204	4	859.5	0	**0.6887**	4	960.4	0	0.5940
	5	791.1	0		5	872	0	
R205	5	1129.9	0	0.3750	5	1093.2	0	**0.3994**
R206	4	1006.2	0	**0.5986**	4	1077.6	0	0.5592
	5	1000.4	0		5	1028.5	0	
R207	4	855.3	0	**0.4271**	4	919.1	0	0.3750
R208	3	774.5	0	**0.7055**	3	885.8	0	0.5807
	4	772.5	0		4	864.3	0	
R209	4	1188.7	0	**0.6345**	5	1253.2	0	0.3750
	5	1096	0					
R210	4	1116.2	0	0.5692	4	1049.9	0	**0.5874**
	5	1050.4	0					
R211	4	858.7	0	**0.4916**	4	1016.8	0	0.3750
Count		**18**				15		
Average				**0.5563**				0.4810

Appendix A11: Comparison with ALNS's hypervolume based on the RC1 instance type and 10% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC101	18	1869.9	0	**0.4402**	19	1929.7	0	0.3750
RC102	17	1748.2	0	0.3750	16	1609.1	0	**0.4858**
RC103	14	1538.9	0	**0.3819**	14	1553.1	0	0.3750
RC104	12	1476.3	0	**0.5303**	14	1522.6	0	0.3760
	13	1457.5	0			1516.7	1	
		1413.6	1					
RC105	18	1833.9	0	0.3750	18	1766.2	0	**0.4027**
RC106	16	1619.4	0	0.3750	16	1610.2	0	**0.3793**
RC107	14	1499.5	0	**0.4250**	15	1468.7	0	0.3904
RC108	14	1415.5	0	0.3750	13	1348.3	0	**0.4693**
Count		**10**				9		
Average				**0.4097**				0.4067

Appendix A12: Comparison with ALNS's hypervolume based on the RC2 instance type and 10% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC201	7	1463	0	**0.6454**	7 8	1802.5 1541.8	0 0	0.5772
RC202	6 7	1363.9 1204.4	0 0	**0.5699**	6	1354.9	0	0.4885
RC203	5	1059.5	0	**0.4802**	5	1232.3	0	0.3750
RC204	4 5	988.8 886	0 0	**0.6100**	4	996	0	0.5250
RC205	7 8	1338.8 1328.3	0 0	0.5604	6 7	1474.5 1442.4	0 0	**0.5829**
RC206	5 6	1294.9 1292.9	0 0	**0.6065**	5	1447.5	0	0.5000
RC207	5	1185.2	0	**0.4360**	5	1290.1	0	0.3750
RC208	4	903.3	0	**0.3868**	4	917.8	0	0.3750
Count		12				10		
Average				**0.5369**				0.4748
Overall Count		81				68		
Overall Average				**0.4727**				0.4288

Appendix A13: Comparison with ALNS hypervolume based on the C1 instance type and 30% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
C101	11	903.4	0	0.3756	11	904.1 865.5	0 1	**0.3857**
C102	11	949	0	**0.5801**	11 12	1133.7 1028.9	0 0	0.5068
C103	11	1007.6	0	0.3750	11	952.4	0	**0.4161**
C104	10	981.4	0	**0.4585**	10	1104.3	0	0.3750
C105	11	947.6	0	**0.3833**	11	958.2	0	0.3750
C106	11	946	0	**0.3762**	11	947.5	0	0.3750
C107	11	858.6	0	**0.3750**	10	827.6	0	**0.4752**
C108	11	941.7	0	**0.4017**	11	976.4	0	0.3750
C109	11	1002.4	0	**0.4330**	11	1086.4	0	0.3750
Count		9				**11**		
Average				**0.4176**				0.4065

Appendix A14: Comparison with ALNS's hypervolume based on the C2 instance type and 30% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
C201	3	586.5	3	**0.4898**	3	586.5	3	0.3987
	4	681.5	0			589	2	
C202	3	587	5	**0.2263**	3	590.1	3	0.2256
		590.1	3			589.1	4	
C203	3	627.1	0	**0.8267**	4	706.5	0	0.5543
					5	678.9	0	
C204	4	693.1	0	**0.3833**	4	700.9	0	0.3750
C205	3	586.4	0	**0.4801**	3	682	0	0.3750
C206	3	586	0	**0.3750**	3	586	0	**0.3750**
C207	3	592	0	**0.3808**	3	596.6	0	0.3750
C208	3	587.1	0	**0.3793**	3	590.5	0	0.3750
Count		10				**11**		
Average				**0.4427**				0.3817

Appendix A15: Comparison with ALNS's hypervolume based on the R1 instance type and 30% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R101	22	1971.3	0	**0.4793**	22	1923.9	0	0.4953
	23	1939	0		24	1910.5	0	
					25	1904	0	
R102	19	1708	0	**0.4791**	20	1772.6	0	0.4184
					21	1754.5	0	
R103	16	1347	0	**0.5549**	16	1398.4	1	0.4246
					18	1505.7	0	
R104	13	1261	0	0.4477	13	1203.7	0	**0.4675**
R105	14	1228.8	0	**0.3902**	17	1593.9	0	0.3750
	17	1561.5	0					
R106	15	1423.3	0	0.3750	15	1387.5	0	**0.3939**
R107	13	1311.7	0	**0.4677**	13	1345.7	0	0.4628
	14	1280.4	0		14	1284.3	0	
R108	12	1087.6	0	**0.4453**	12	1200.1	0	0.3750
R109	14	1391.6	0	**0.4417**	15	1390.9	0	0.3754
	15	1360.6	0					
R110	14	1286	0	0.4334	14	1299.2	0	**0.4450**
					15	1264.5	0	
R111	14	1267.5	0	0.3750	14	1251.5	0	**0.3845**
R112	12	1131.3	0	**0.4671**	13	1178.1	0	0.3750
Count		16				**18**		
Average				**0.4464**		0.4160		

Appendix A16: Comparison with ALNS's hypervolume based on the R2 instance type and 30% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R201	6	1398.2	0	0.3881	5	1423.1	0	**0.5000**
R202	5	1423.8	0	0.6165	5	1287.7	0	**0.6372**
	6	1202.6	0		6	1208.7	0	
R203	5	956.8	0	**0.3814**	5	965	0	0.3750
R204	4	832.7	0	**0.7989**	4	1118	0	0.6219
	5	823.8	0		5	973.5	0	
R205	5	1101.7	0	0.3800	4	1109.1	0	**0.5250**
R206	4	1041.9	0	**0.4031**	4	1082.4	0	0.3750
R207	4	907.5	0	**0.4398**	4	993.3	0	0.3750
R208	3	790.4	0	**0.7516**	3	950.1	0	0.6652
					4	820	0	
R209	5	1047	0	0.4700	4	1198.7	0	**0.5923**
					5	1091.1	0	
R210	5	1036	0	0.4731	5	1139.9	0	**0.5578**
					4	1192.1	0	
R211	4	889.1	0	**0.4537**	4	993.3	0	0.3750
Count		13				**16**		
Average				0.5051				**0.5090**

Appendix A17: Comparison with ALNS's hypervolume based on the RC1 instance type and 30% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC101	19	2003.9	0	0.4203	19	1925.6	0	**0.4447**
	20	1983.1	0					
RC102	16	1686.2	0	**0.4132**	16	1776.6	0	0.3750
RC103	14	1592.5	0	**0.4462**	14	1571.3	0	0.4363
	15	1557.6	0					
	14	1535.2	1					
RC104	13	1495.3	0	0.3750	13	1422.5	0	**0.4115**
RC105	17	1985.5	0	**0.4672**	18	1814.6	0	0.4396
	18	1851.8	0					
RC106	15	1598.6	0	0.3750	15	1574.3	0	**0.3864**
RC107	14	1448	0	**0.5344**	15	1557.3	0	0.4318
					16	1536.6	0	
RC108	13	1385	0	**0.4698**	14	1451.4	0	0.4303
					13	1454.8	0	
Count		**12**				10		
Average				**0.4376**				0.4195

Appendix A18: Comparison with ALNS's hypervolume based on the RC2 instance type and 30% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC201	7	1913.1	0	0.6492	7	1519.9	0	**0.6922**
	8	1452.8	0		8	1441.3	0	
RC202	6	1623	0	0.6288	6	1351.4	0	**0.6435**
	7	1305.7	0					
RC203	5	1044.1	0	**0.3858**	5	1059.4	0	0.3750
RC204	4	881.7	0	**0.4691**	4	1008.2	0	0.3750
RC205	7	1239	0	0.4585	6	1394.2	0	**0.4948**
					7	1370.6	0	
RC206	5	1344.5	0	0.3750	5	1252.9	0	**0.4261**
RC207	5	1194	0	0.5476	5	1253.6	0	**0.5682**
					6	1139.6	0	
RC208	4	914.5	0	**0.4533**	4	1021.1	0	0.3750
Count		10				**11**		
Average				**0.4959**				0.4937
Overall Count		70				**77**		
Overall Average				**0.4576**				0.4377

Appendix A19: Comparison with ALNS's hypervolume based on the C1 instance type and 50% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
C101	11	968	0	**0.3959**	11	961.7	0	0.3853
		887.1	1			940.8	1	
C102	12	1043.5	0	0.3976	11	1075.9	0	**0.4375**
C103	12	1077.8	0	0.3750	10	1035.8	0	**0.5390**
C104	10	1094.2	0	**0.5249**	10	1159.9	0	0.4569
	11	1045.5	0		11	1138.7	0	
C105	10	873.3	0	**0.5383**	11	978.3	0	0.3750
C106	11	945	0	**0.3767**	11	947.2	0	0.3750
C107	11	1044.8	0	0.4524	11	937.8	0	**0.5271**
	12	1024	0					
C108	11	930.3	0	**0.4506**	11	1034.6	0	0.3750
C109	11	1002	0	**0.5968**	11	1214.9	0	0.4510
	12	992.4	0		12	1193.1	0	
Count		**12**				11		
Average				**0.4565**				0.4357

Appendix A20: Comparison with ALNS hypervolume based on C2 instance type and 50% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
C201	3	586.5	3	0.3739	3	586.6	2	**0.7897**
	5	876.4	0		3	586.5	3	
C202	4	604	5	0.5110	4	613.7	1	**0.5507**
	4	613.7	1		4	610.5	2	
	4	600.4	6		4	824	0	
	4	605.3	2		4	604	4	
C203	3	639.7	0	**0.5686**	4	637.1	0	0.3780
	4	634.5	0					
C204	4	645.9	0	**0.3787**	4	649.1	0	0.3750
C205	4	623.4	0	**0.3858**	4	632.5	0	0.3750
C206	3	586	0	**0.4267**	3	629.4	0	0.3750
C207	3	592	0	**0.3808**	3	596.6	0	0.3750
C208	3	587.6	0	**0.3848**	3	595.4	0	0.3750
Count		13				12		
Average				0.4263				**0.4492**

Appendix A21: Comparison with ALNS's hypervolume based on the R1 instance type and 50% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R101	21	1868.2	2	0.4405	22	1966.8	0	**0.4442**
	22	1959.5	0		22	1851.8	1	
	23	1952	0		23	1929	0	
R102	16	1431.5	0	**0.4791**	15	1365.2	2	0.4184
	20	1766.7	1		19	1762.7	0	
R103	15	1383.8	2	0.3954	21	1720.5	0	**0.5013**
	15	1391.2	1		18	1546.1	0	
	16	1377.2	1		16	1455.7	1	
	23	1925.2	1					
R104	13	1256.2	0	0.4508	13	1159.2	0	**0.4948**
	14	1219	0					
R105	17	1561.6	0	0.3750	17	1542.3	0	**0.3843**
R106	14	1405	0	**0.4498**	15	1428.7	0	0.4345
					14	1447	0	
R107	13	1274.8	0	**0.4349**	14	1284.3	0	0.3750
R108	12	1185.2	0	0.4411	12	1169.1	0	**0.4444**
	13	1171.9	0					
R109	15	1365.8	0	0.3750	14	1334	0	**0.4448**
R110	14	1282.1	0	**0.4226**	14	1368.9	0	0.3750
R111	13	1450.4	0	**0.5888**	14	1358.5	0	0.4861
	14	1256.3	0		15	1344.5	0	
R112	12	1136	0	**0.4837**	12	1207.3	0	0.4554

					13	1170.7	0	
Count		21				20		
Average				**0.4451**				0.4376

Appendix A22: Comparison with ALNS's hypervolume based on the R2 instance type and 50% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R201	5	1470.6	0	**0.5630**	5	1496.4	0	0.5000
	6	1379.4	0					
R202	5	1255.4	0	**0.6083**	5	1383.7	0	0.5000
	6	1226.7	0					
R203	5	1023.4	0	**0.6492**	5	1115.9	0	0.5586
R203	4	1046.5	0		4	1168.2	0	
R204	4	831.5	0	**0.7368**	4	1041.6	0	0.6105
					5	922.8	0	
R205	4	1253.6	0	**0.5883**	4	1266.3	0	0.5250
	5	1164.5	0					
R206	4	992.9	0	**0.6877**	4	1174.9	0	0.6044
					5	1050.5	0	
R207	4	921.8	0	**0.4432**	4	1014	0	0.3750
R208	3	884.2	0	**0.6795**	3	910.8	0	0.6457
	4	782	0		4	809.8	0	
R209	5	1066.5	0	**0.6890**	4	1312.8	0	0.5597
	4	1210.9	0		5	1252	0	
R210	5	1124.5	0	**0.4027**	5	1167.7	0	0.3750
R211	3	881.4	0	**0.7890**	4	933.9	0	0.6778
					3	1103.6	0	
Count		**17**				**17**		
Average				**0.6215**				0.5393

Appendix A23: Comparison with ALNS's hypervolume based on the RC1 instance type and 50% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC101		1890.6	1	0.4683	17	1918.8	0	**0.5076**
	18	2022.2	0			1900.9	1	
		1858	3		19	1896.3	4	
	19	1986	0					
RC102	16	1642.4	1	**0.4501**				0.4124
	17	1777.7	0		17	1711.1	0	
	16	1800.8	0					
RC103	14	1519.2	1	0.4366	14	1565.6	0	**0.4428**
	14	1461.1	2			1520.5	1	
	15	1568.4	0					

RC104	13	1443	0	**0.4491**	13 14	1478.7 1471.2	0 0	0.4324
RC105	17	1772.3	0	**0.4324**	17	1752.6	0	0.3833
RC106	15	1580	0	**0.4231**	15	1688.3	0	0.3750
RC107	14	1443.9	0	**0.4662**	15	1517.4	0	0.3750
RC108	13	1405.3	0	**0.4352**	14	1416.3	0	0.3750
Count		13				12		
Average				**0.4451**				0.4129

Appendix A24: Comparison with ALNS's hypervolume based on the RC2 instance type and 50% DoD

Instance	Proposed Algorithm				ALNS			
	NV	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
RC201	6 8	1794.4 1443.2	0 0	**0.7902**	6 7	1956.3 1509	0 0	0.7769
RC202	6 7	1412.9 1321.1	0 0	0.5309	6	1296.3	0	**0.5617**
RC203	5	1038	0	**0.4223**	5	1107.8	0	0.3750
RC204	4	858.5	0	**0.3943**	4	881.2	0	0.3750
RC205	6 7	1543 1435.3	0 0	**0.5604**	6 7	1588.1 1558.7	0 0	0.4960
RC206	5 6	1436.4 1146.9	0 0	**0.7032**	5 6	1530.3 1427.3	0 0	0.5505
RC207	5 6	1218.1 1142.3	0 0	**0.6818**	5 6	1419.2 1281.4	0 0	0.5728
RC208	4	940.9	0	**0.5126**	4	1152.3	0	0.3750
Count		**13**				12		
Average				**0.5745**				0.5104
Overall Count		**90**				85		
Overall Average				**0.4948**				0.4642

Appendix A25: Comparison with ALNS Hypervolume based on the C1 instance type and 70% DoD

Instance	Proposed Algorithm				ALNS			
	NV	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
C101	11	906.1	1	0.1250	11	905.9	1	**0.1251**
C102	12 13	1200.7 1160.6	0 0	0.4577	11	1088.2	0	**0.5823**
C103	11 12 13	1093.2 1084.4 1077.6	0 0 0	0.5020	11	1007.2	0	**0.5675**

C104	10	994.8	0	**0.5048**	11	1042.3	0	0.3750
	11	965.3	0					
C105	11	902.4	0	**0.4346**	11	980.3	0	0.3750
C106	11	1005.9	0	**0.3882**	11	1023.9	0	0.3750
C107	11	929.2	0	**0.4211**	11	990	0	0.3750
C108	11	964	0	0.3750	11	928.1	0	**0.4029**
C109	11	969.1	0	**0.4468**	11	1071.7	0	0.3750
Count		13				9		
Average				**0.4061**				0.3948

Appendix A26: Comparison with ALNS's hypervolume based on the C2 instance type and 70% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
C201	3	591.7	5	**0.3493**	3	591.7	5	0.6466
		591.6	6			598.3	3	
					4	738.2	0	
C202	3	580.3	7	**0.7156**	4	604	4	0.4838
		586.5	5			613.7	1	
		587.1	3					
		771.3	2					
	4	613.7	1					
C203	3	643.8	0	**0.6422**	4	692.9	0	0.3750
C204	4	751.2	0	**0.4162**	4	794.9	0	0.3750
C205	3	593.2	5	**0.4835**	4	672.7	0	0.3750
	4	670.4	0					
C206	3	586	0	**0.3776**	3	587.1	0	0.3750
		583.1	1					
C207	3	602.7	0	**0.3762**	3	601.7	0	**0.3762**
		599.9	1					
C208	3	589.4	0	0.3750	3	588.2	0	**0.3765**
Count		**16**				11		
Average				**0.4670**				0.4229

Appendix A27: Comparison with ALNS Hypervolume based on the R1 instance type and 70% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R101	21	1826.3	2	**0.5052**	22	1875.5	0	0.4742
		1867.4	1					
		1957.7	0					
	24	1949.1	0					
R102	19	1757.5	0	**0.4637**	19	1636.1	2	0.4622
	20	1742.5	0			1683.4	1	
	21	1778	0					
R103	16	1394	1	0.1622	15	1516.1	1	**0.4255**
					16	1435.9	1	

					17	1497.1	0	
R104	12	1160.7	0	**0.4622**	13	1201.6	0	0.3750
R105	16	1510.5	0	**0.3851**	16	1531.1	0	0.3750
R106	14	1455.5	0	**0.5190**	15	1518.7	0	0.4400
	15	1395	5		16	1482	0	
R107	13	1274.7	0	**0.4353**	14	1268.8	0	0.3785
		1244.6	2					
R108	11	1154.6	0	**0.4666**	11	1191.7	0	0.4547
	12	1151.6	0		12	1164.4	0	
R109	14	1349	0	**0.6304**	16	1534.8	0	0.4302
					17	1512.1	0	
R110	14	1407.6	0	**0.4882**	15	1370.7	0	0.3947
	15	1289	0					
R111	13	1328.2	0	**0.4600**	14	1276.3	0	0.4043
	14	1272.5	0					
R112	12	1138.1	0	**0.4629**	13	1179.3	0	0.3750
Count		**22**				18		
Average				**0.4500**				0.4200

Appendix A28: Comparison with ALNS's hypervolume based on the R2 instance type and 70% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R201	5	1416.7	0	**0.4023**	5	1470.2	0	0.375
R202	5	1173	0	**0.574**	5	1266.4	0	0.5457
R202	5	1173	0	**0.574**	5	1266.4	0	0.5457
					6	1189.3	0	
R203	5	1038	0	0.669	4	1248.7	0	**0.7627**
					5	1089.7	0	
					6	1072.4	0	
R204	4	852.4	0	**0.6079**	4	925.1	0	0.525
	5	851.9	0					
R205	4	1268.3	0	**0.626**	4	1357.7	0	0.525
	5	1210.6	0					
R206	4	1093.1	0	**0.6088**	4	1179.5	0	0.525
	5	1082.3	0					
R207	4	940	0	**0.4342**	4	1020.5	0	0.375
R208	4	806.3	0	**0.4154**	3	852.2	0	**0.4154**
					4	826	0	
R209	4	1049.1	0	**0.4285**	4	1129.7	0	0.375
R210	5	1062.7	0	**0.4744**	5	1225	0	0.375
R211	3	1076.5	0	**0.6972**	3	1094.5	0	0.6608
	4	906.9	0		4	951.1	0	
Count		15				16		
Average				**0.5398**				0.4963

Appendix A29: Comparison with ALNS's hypervolume based on the RC1 instance type and 70% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC101	17	1761	3	**0.5524**	17	1823.9	4	0.4938
		1803	0			1727.8	4	
					18	1778.2	3	
						1871.1	1	
					19	1999	0	
RC102	15	1627.3	1	**0.4089**	16	1717.2	0	0.3750
	16	1705	0					
RC103	13	1489.7	0	**0.4778**	14	1580.4	0	0.3750
RC104	13	1478.1	0	**0.4429**	14	1503.2	0	0.3750
RC105	17	1805	0	**0.5205**	18	1948.2	0	0.4547
					19	1843.7	0	
RC106	15	1637.2	0	0.4252	15	1585.5	0	**0.4485**

	16	1630	0					
RC107	14	1501.6	0	**0.4720**	15	1589.5	0	0.3750
RC108	14	1436	0	**0.4722**	14	1520.9	0	0.4252
					15	1520.4	0	
Count		11				14		
Average				**0.4715**				0.4153

Appendix A30: Comparison with ALNS's hypervolume based on the RC2 instance type and 70% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC201	7	1522.4	0	**0.5752**	7	1619.4	0	0.4704
	8	1413.7	0		8	1615.9	0	
RC202	5	1552.3	0	**0.7404**	5	1526.4	0	0.6134
	6	1500.6	0		6	1519.3	0	
	7	1254.3	0					
RC203	5	1117	0	**0.6117**	5	1257.4	0	0.5134
					6	1235	0	
RC204	4	909.8	0	0.4346	3	988.3	0	**0.5625**
RC205	7	1352	0	0.5176	6	1669.3	0	**0.5463**
					7	1526.4	0	
RC206	5	1361.5	0	**0.5481**	5	1298.8	0	0.5461
	6	1274.1	0					
RC207	5	1231	0	**0.4412**	5	1350.1	0	0.3750
RC208	4	917.3	0	**0.7724**	4	1200.1	0	0.5312
					5	1190.2	0	
Count		12				**13**		
Average				**0.5802**				0.5198
Overall Count		**89**				81		
Overall Average				**0.4858**				0.4449

Appendix A31: Comparison with ALNS's hypervolume based on the C1 instance type and 90% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
C101	11	891.7	4	**0.5724**	12	1043.9	0	0.4526
		907.7	3					
		983.9	2					
		986.2	1					
	12	1164.3	0					
C102	12	1022.9	0	0.3750	11	975.2	0	**0.4783**

C103	11	1045	0	**0.5340**	11	1174.9	0	0.5122
					12	1057.9	0	
C104	10	1029.9	0	**0.5917**	11	1109.4	0	0.5182
	11	1004.5	0		12	990.1	0	
C105	11	967.3	0	**0.4032**	11	994.1	0	0.3781
		935.4	4			987.9	2	
C106	11	1104.2	1	**0.5617**	13	1248.1	0	0.3750
	12	1096.1	0					
C107	11	928.4	0	**0.4198**	11	987.4	0	0.3750
C108	11	983.3	0	**0.5453**	11	1121.5	0	0.4764
					12	1063.3	0	
C109	10	1054.4	0	**0.6168**	12	1193.8	0	0.3750
Count		16				13		
Average				**0.5060**				0.4379

Appendix A32: Comparison with ALNS's hypervolume based on the C2 instance type and 90% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
C201	3	591.1	8	**0.4783**	4	605.5	1	0.3990
		588.5	9			608.4	0	
	4	625.7	0					
C202	3	572.8	8	**0.9751**	4	614.1	1	0.8342
		583.8	4			897.6	0	
		583.8	5		5	823	0	
	4	645.7	1					
C203	4	685.5	0	**0.4149**	4	724	0	0.3750
C204	4	686.3	0	**0.4036**	4	713.5	0	0.3750
C205	3	654.3	7	0.5003	3	638.3	0	**0.7739**
	4	786	0					
C206	3	607.5	0	0.3750	3	600.9	0	**0.3831**
C207	3	620.8	0	0.3750	3	608.6	0	**0.3897**
C208	3	599.6	0	**0.4398**	3	656.3	0	0.3750
Count		14				11		
Average				**0.4952**				0.4881

Appendix A33: Comparison with ALNS's hypervolume based on the R1 instance type and 90% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
R101	23	1937.2	2	0.3277	22	1850.2	4	**0.3512**
		1967	1			1908.2	3	
					23	1991.5	1	
						1939	2	
R102	20	1767	0	0.3777	19	1773.8	0	**0.4173**
					20	1739.7	1	
R103	15	1444.5	1	**0.3098**	15	1386.3	2	0.3087
	17	1438.8	2		16	1433.2	1	
R104	12	1198.9	0	**0.4904**	13	1202.7	0	0.4286

	14	1190.5	0					
	15	1508	0					
R105	16	1492	6	0.4273	15	1478	0	**0.4387**
	16	1499.7	0					
R106	14	1478.7	0	**0.4606**	15	1400.5	0	0.4147
	15	1408.5	0					
	12	1228.4	2					
R107	13	1211.9	1	**0.4866**	13	1275.7	0	0.4425
		1296.8	0					
	14	1292.4	0					
R108	12	1152.4	0	0.4341	12	1114.9	0	**0.4609**
	13	1150.3	0					
	14	1358.4	2					
R109	14	1419.4	0	**0.4556**	15	1427.2	0	0.3750
	15	1380	0					
R110	15	1334.7	0	**0.3850**	15	1352.8	0	0.3750
R111	13	1286.2	0	**0.5160**	14	1323.1	0	0.4752
	14	1264.3	0		13	1410.9	0	
R112	13	1184.9	0	**0.3846**	13	1200.2	0	0.3750
Count		**25**				18		
Average				**0.4213**				0.4052

Appendix A34: Comparison with ALNS's hypervolume based on the R2 instance type and 90% DoD

Instance	**Proposed Algorithm**				**ALNS**			
	NV	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
R201	5	1388.4	0	**0.4060**	5	1448.3	0	0.3750
R202	5	1293.4	0	0.5259	5	1292.6	0	**0.5309**
	6	1248.8	0		6	1240.3	0	
R203	5	1073	0	**0.5272**	5	1103	0	0.5186
					6	1075.6	0	
R204	4	835.2	3	0.3920	4	872.7	0	**0.3947**
	4	896.2	0					
R205	4	1185.7	0	**0.4331**	4	1285.3	0	0.3750
R206	4	1101.1	0	0.5292	4	1074.1	0	**0.5507**
	5	1094.9	0					
R207	4	925.2	0	**0.4491**	4	1026.6	0	0.3750
R208	3	918.5	0	**0.6391**	4	899.2	0	0.3908
	4	824.7	0					
R209	4	1163.4	0	0.3750	4	1004.1	0	**0.4777**
R210	5	1163	0	0.3750	5	1120.4	0	**0.4025**
R211	4	924.6	0	0.3750	4	912.8	0	**0.3846**
Count		**15**				**15**		
Average				**0.4570**				0.4341

Appendix A35: Comparison with ALNS's hypervolume based on the RC1 instance type and 90% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC101	17	1835.6	2	**0.4243**	18	1860.1	0	0.3818
		1846.9	0			1809.7	2	
RC102	16	1704.8	0	**0.5344**	17	1709.4	2	0.4403
						1776.7	1	
		1694	2		18	1747.4	1	
						1855.2	0	
RC103	14	1508.6	1	0.4570	14	1599.6	0	**0.4809**
		1672.5	0		15	1557.5	0	
	15	1658.3	0					
RC104	13	1343	0	**0.6220**	14	1588.9	0	0.4582
					15	1518.6	0	
RC105	18	1876.1	2	**0.4385**	17	1957.8	3	0.4195
		1849.8	3		18	1733.6	4	
						1878.7	2	
	19	1891	0		19	1810.3	1	
RC106	14	1655.9	0	0.3750	14	1595.6	0	**0.4023**
RC107	14	1474.4	0	**0.3766**	14	1477.5	0	0.3750
RC108	14	1473	0	**0.4348**	14	1489.7	0	0.4294
					15	1480.9	0	
Count		14				18		
Average				**0.4578**				0.4234

Appendix A36: Comparison with ALNS's hypervolume based on the RC2 instance type and 90% DoD

Instance	Proposed Algorithm				ALNS			
	NV	TD	RR	HV	NV	TD	RR	HV
RC201	7	1533.5	0	0.6433	7	1520.5	0	**0.7678**
					5	1884.2	0	
	6	1694	0		6	1587.1	0	
RC202	5	1637.3	0	0.6024	5	1544.5	0	**0.6063**
	6	1413.8	0		6	1436.2	0	
RC203	5	1162.6	0	0.375	5	1106.7	0	**0.4111**
RC204	4	995.5	0	0.375	4	951.2	0	**0.4084**
RC205	6	1489.8	0	0.5783	6	1627.6	0	**0.6068**
	7	1458.2	0		7	1357	0	
RC206	5	1318	0	0.5186	5	1247.4	0	**0.5536**
	6	1285.4	0					
RC207	5	1326.2	0	0.5138	5	1302.8	0	**0.5176**
	6	1301.8	0					
RC208	4	1010.6	0	0.375	4	1000.6	0	**0.3824**
Count		**13**				12		
Average				0.4977				**0.5317**
Overall Count		**97**				87		

Overall Average	**0.4725**	0.4534

Appendix A37: Comparison with ALNS's hypervolume based on the 50 customers and 10% DoD

Customer Size	DoD	Instance	NV	TD	RR	HV	NV	TD	RR	HV
50	10	test50-0-0-0-0.d0.tw0	3	585.4	0	0.3750	3	563.6	0	0.4029
		test50-0-0-0-0.d0.tw1	5	830	0	**0.3784**	5	833.8	0	0.3750
		test50-0-0-0-0.d0.tw2	6	847.9	0	**0.4500**	6	903.7	0	0.3750
		test50-0-0-0-0.d0.tw3	7	1001	0	0.3750	7	939.5	0	0.4211
		test50-0-0-0-0.d0.tw4	4	841.6	0	0.5998	4	793.6	0	0.6375
			5	757.7	0		5	734.6	0	
		test50-0-0-0-0.d1.tw0	5	740.2	0	0.3750	5	693.7	0	0.4221
		test50-0-0-0-0.d1.tw1	6	830.9	0	**0.7100**	5	1083.7	0	0.5000
			5	931.6	0					
		test50-0-0-0-0.d1.tw2	6	957.7	0	0.3750	6	941.6	0	0.3876
		test50-0-0-0-0.d1.tw3	7	1033	0	**0.3806**	7	1040.8	1	0.3750
		test50-0-0-0-0.d1.tw4	5	843	0	0.5415				0.5512
			6	796.4	0		5	799.8	0	
		test50-0-0-0-0.d2.tw0	15	1567.9	0	0.3750	15	1457.9	0	0.4276
		test50-0-0-0-0.d2.tw1	15	1673.8	0	0.4038	15	1468.8	0	0.4669
			15	1598.5	1					
			14	1631.3	1					
		test50-0-0-0-0.d2.tw2	15	1732.7	0	0.3750	15	1559.5	0	0.4500
		test50-0-0-0-0.d2.tw3	15	1641.4	0	**0.4472**	15	1579.9	0	0.4535
			16	1585.9	0					
		test50-0-0-0-0.d2.tw4	15	1515.9	2	**0.3268**	15	1602.7	2	0.1736
			16	1615.3	1					
			14	1692.7	2		14	1708.2	2	
		Count		**23**				17		
		Average				**0.4325**				0.4279

Appendix A38: Comparison with ALNS's hypervolume based on the 50 customers and 30% DoD

Customer Size	DoD	Instance	NV	TD	RR	HV	NV	TD	RR	HV
50	30	test50-0-0-0-0.d0.tw0	3	562	0	**0.5439**	3	709.4	0	0.4287
			3	525	1		3	557.1	1	
		test50-0-0-0-0.d0.tw1	5	730	4	**0.3482**	5	697.5	4	0.2499
			5	929.8	3					
			6	631.1	4					
		test50-0-0-0-0.d0.tw2	6	1216.1	0	0.5110	6	1216.1	0	**0.5684**
			5	872	1		5	872	1	
		test50-0-0-0-0.d0.tw3	7	1060.2	1	**0.5480**	8	987.3	1	0.5374
			8	1230.7	0		8	1246.7	0	
			4	729.7	1		9	1208.2	0	
		test50-0-0-0-0.d0.tw4	4	708.9	2	**0.3822**	4	717.6	1	0.3616

			5	686.8	1					
		test50-0-0-0-0.d1.tw0	5	827.2	0	**0.4424**	5	908.9	0	0.3750
		test50-0-0-0-0.d1.tw1	6 5	907.2 918.2	1 1	0.1808	5 5	855.4 950.2	1 0	**0.5333**
		test50-0-0-0-0.d1.tw2	5 6	936.2 1165	1 0	**0.4821**	5	1005.4	1	0.2123
		test50-0-0-0-0.d1.tw3	7 7	1300.2 1173.1	0 2	**0.4212**	7 7	1340.4 1163.9	1 2	0.2829
		test50-0-0-0-0.d1.tw4	6 5	862.5 988.9	0 0	**0.5959**	5	959.6	0	0.5296
		test50-0-0-0-0.d2.tw0	15	1540.4	2	**0.1361**	15	1612.1	2	0.1250
		test50-0-0-0-0.d2.tw1	14	1607.4	1	**0.1410**	14	1717.2	1	0.1250
		test50-0-0-0-0.d2.tw2	14 15	1648.7 1618.9	0 0	**0.4386**	15	1557	0	0.4167
		test50-0-0-0-0.d2.tw3	15	1565.8	1	0.1566	15 15	1568.7 1792.7	1 0	**0.4062**
		test50-0-0-0-0.d2.tw4	15 14	1645.7 1722.1	0 0	0.4583	15 14	1602.2 1631.9	0 0	**0.4825**
		Count		**28**				23		
		Average				**0.3858**				0.3756

Appendix A39: Comparison with ALNS's hypervolume based on the 50 customers and 50% DoD

Customer Size	**DoD**	**Instance**	**NV**	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
50	50	test50-0-0-0-0.d0.tw0	3	620.9	0	0.3750	3	546.4	0	**0.4650**
		test50-0-0-0-0.d0.tw1	5 4 5	1172.6 1000.3 1007.7	0 2 1	**0.6499**	4 5	1000.9 1432.1	2 0	0.5304
		test50-0-0-0-0.d0.tw2	6 5 5	1138.3 1154.5 1045.2	0 1 2	0.5630	5 5 6	953.6 1024 1248.5	2 1 0	**0.5970**
		test50-0-0-0-0.d0.tw3	7 7	1152 1356.7	2 1	**0.2877**	7	1227.1	2	0.1489
		test50-0-0-0-0.d0.tw4	4 5	857.3 912	4 1	**0.4076**	4 4	857 944.3	4 2	0.3824
		test50-0-0-0-0.d1.tw0	5 5	929.3 1166.6	1 0	0.4259	5 5	983.6 926.6	0 1	**0.5049**
		test50-0-0-0-0.d1.tw1	5 5	1131.8 1090.7	0 1	0.4025	5 5	1031.1 1160.6	1 0	**0.4029**
		test50-0-0-0-0.d1.tw2	5 5 6	903.6 904 957	4 2 1	**0.5855**	6 5 5 6	939.5 1199.5 968.8 1060.1	2 3 4 1	0.5139
		test50-0-0-0-0.d1.tw3	6 6	1124.1 1176.3	5 4	0.2192	6 6	1151 1303.8	5 3	**0.2543**
		test50-0-0-0-0.d1.tw4	5	973.8	0	**0.3755**	5	974.4	0	0.3750
		test50-0-0-0-0.d2.tw0	15	1897.1	0	**0.4493**	15	2105.8	0	0.3750
		test50-0-0-0-0.d2.tw1	14	1661.9	0	0.3750	14	1661.1	0	**0.3754**
		test50-0-0-0-0.d2.tw2	15 16	1836.3 1786.1	0 0	0.4424	15	1655.1	0	**0.5051**
		test50-0-0-0-0.d2.tw3	13 13	1396.3 1307.4	2 5	**0.3573**	13 13	1413.9 1594.4	3 2	0.3259
		test50-0-0-0-0.d2.tw4	14 15	1829.6 1736.3	1 1	**0.4314**	15	1928.1	0	0.3974

	15	1877.5	0		14	1888.8	1	
Count	**30**				27			
Average				**0.4231**				0.4102

Appendix A40: Comparison with ALNS's hypervolume based on the 50 customers and 70% DoD

Customer Size	**DoD**	**Instance**	**NV**	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
			3	763.3	2		4	971.7	0	
		test50-0-0-0-0.d0.tw0	3	934.2	1	**0.6439**	3	924.6	2	0.5197
			4	1050.2	0					
			5	929.5	3		4	1231.4	1	
		test50-0-0-0-0.d0.tw1	4	956.7	3	0.3974	4	1008.2	2	0.5817
			4	904	4		5	923	4	
		test50-0-0-0-0.d0.tw2	4	930.2	5	**0.1992**	4	999.4	5	0.1848
							5	960.1	5	
			5	1153.1	6		5	1010.5	7	
		test50-0-0-0-0.d0.tw3	5	1070.2	7	**0.2364**	5	918	8	0.2150
			5	948.2	8					
			3	1172.6	4		4	892	4	
		test50-0-0-0-0.d0.tw4	4	1022.6	1	**0.6193**	4	966	2	0.5453
			4	722.9	6		3	1196.8	6	
			4	764.6	5		4	1036.1	1	
		test50-0-0-0-0.d1.tw0	4	783.1	5	**0.5782**	5	1007.7	2	0.4193
			4	1017.6	2		5	1216.1	1	
50	70	test50-0-0-0-0.d1.tw1	6	1157.3	0	**0.5261**	5	1189	0	0.5000
			5	1160.1	0					
			6	1141.2	3		6	1279.3	1	
		test50-0-0-0-0.d1.tw2	6	1388.9	0	0.4398	6	1130.6	2	0.4798
							6	1429.9	0	
		test50-0-0-0-0.d1.tw3	5	949.5	6	0.2314	5	944.1	6	0.2746
							6	1178.2	5	
		test50-0-0-0-0.d1.tw4	5	971.4	5	0.2495	5	1330.7	3	0.2868
			5	1238.2	4		5	1001.7	5	
		test50-0-0-0-0.d2.tw0	14	1893.3	2	0.1250	14	1884.3	2	0.1262
			15	1842.7	3		14	2045.8	3	
		test50-0-0-0-0.d2.tw1	14	1975.8	3	0.1676	15	1820.6	3	0.3513
							15	1951.1	1	
		test50-0-0-0-0.d2.tw2	15	1751.7	3	**0.1592**	15	1708	3	0.1514
			16	1621.4	3		16	1681.6	3	
		test50-0-0-0-0.d2.tw3	13	1665.2	3	**0.2875**	13	1651.9	7	0.1270
			13	1534.4	7					
		test50-0-0-0-0.d2.tw4	14	1516.8	4	**0.1542**	14	1717.2	4	0.1250
		Count		**31**				**31**		
		Average				**0.3343**				0.3258

Appendix A41: Comparison with ALNS's hypervolume based on the 50 customers and 90% DoD

Customer Size	DoD	Instance	NV	TD	RR	HV	NV	TD	RR	HV
50	90	test50-0-0-0-0.d0.tw0	3	721.7	4	**0.8268**	3	793.4	5	0.7179
		test50-0-0-0-0.d0.tw0	3	986.5	0		3	1086.4	2	
							4	953.9	0	
		test50-0-0-0-0.d0.tw0	3	748.1	3		3	841.4	3	
		test50-0-0-0-0.d0.tw1	4	1018.2	3	0.6014	4	1226.8	2	**0.7021**
		test50-0-0-0-0.d0.tw1	4	1216.1	2		7	1518.3	0	
		test50-0-0-0-0.d0.tw2	3	870.9	9	**0.1572**	3	870.9	9	0.1534
		test50-0-0-0-0.d0.tw2	3	845.7	10		3	859.1	10	
		test50-0-0-0-0.d0.tw3	5	1107.5	6	0.2664	3	870.9	9	**0.3211**
		test50-0-0-0-0.d0.tw3	5	897.1	8					
		test50-0-0-0-0.d0.tw4	4	826.3	4	0.2592	4	811.2	2	**0.2782**
		test50-0-0-0-0.d0.tw4	4	857.7	2		4	807.3	4	
		test50-0-0-0-0.d1.tw0	5	1078.8	0	**0.4948**	5	1283.8	0	0.3863
							5	1225.8	3	
		test50-0-0-0-0.d1.tw1	5	1005.4	5	**0.7869**	5	1211.1	4	0.6516
							6	918.7	7	
		test50-0-0-0-0.d1.tw1	4	1073.4	1		6	961.6	1	
							4	1295.6	7	
							6	961.6	2	
		test50-0-0-0-0.d1.tw2	6	1206.3	3	0.2566	6	1429.7	1	**0.3259**
		test50-0-0-0-0.d1.tw2	6	1351	2		6	1234	3	
		test50-0-0-0-0.d1.tw3	5	1043.6	6	**0.1299**	5	1064.3	6	0.1250
		test50-0-0-0-0.d1.tw4	5	1314.7	3	0.1250	5	1110.5	1	**0.4066**
							5	982.5	3	
		test50-0-0-0-0.d2.tw0	13	1844.4	0	0.3750	13	1843.3	0	**0.3754**
		test50-0-0-0-0.d2.tw1	15	1983.6	4	**0.1281**	15	2008.8	4	0.1250
		test50-0-0-0-0.d2.tw2	16	1825.7	2	**0.1760**	16	2111.1	2	0.1606
							15	2294	2	
		test50-0-0-0-0.d2.tw3	15	1701.2	5	0.1578	15	1726.8	5	0.1712
							14	1957.7	5	
		test50-0-0-0-0.d2.tw4	13	1828.3	5	0.1740	13	1593.5	5	0.1796
		test50-0-0-0-0.d2.tw4	14	1600.4	5					
Count					24			**30**		
Average						0.3277				**0.3387**
Overall Count					136			128		
Overall Average						**0.3807**				0.3756

Appendix A42: Comparison with ALNS's hypervolume based on the 150 customers and 10% DoD

Customer Size	DoD	Inst	NV	TD	RR	HV	NV	TD	RR	HV
150	10	test150-0-0-0-0.d0.tw0	7	920.9	2	**0.2825**	7	961.6	2	0.2608
							8	960.2	3	
		test150-0-0-0-0.d0.tw1	10	1535.9	1	**0.4825**	11	2014.1	0	0.3938
			11	1955.3	0		11	1862.6	1	
		test150-0-0-0-0.d0.tw2	12	2039.1	1	**0.5109**	13	2319.7	0	0.3967
			13	2026.6	0		12	2300	1	
			13	1866.8	1					
		test150-0-0-0-0.d0.tw3	17	2128.3	0	0.3763	17	2070	0	0.4409
							16	2132	0	
		test150-0-0-0-0.d0.tw4	12	1665.2	1	0.5983	12	1850.3	0	0.6390
			11	1724.5	2		12	1662.2	2	
			12	2083.7	0		13	1605.6	1	
			11	1831.8	1		11	1886.9	1	
							11	1806.6	2	
		test150-0-0-0-0.d1.tw0	7	965	1	**0.4276**	7	1062.8	2	0.3450
							8	1059.3	2	
							7	1182.9	1	
		test150-0-0-0-0.d1.tw1	10	1621.1	1	0.2213	10	1702.9	1	0.5162
							11	1611	1	
							10	2159	0	
		test150-0-0-0-0.d1.tw2	13	1833.3	1	0.1692	13	1745	1	0.1829
			12	2037	1		12	1884.8	1	
		test150-0-0-0-0.d1.tw3	16	2010.1	2	**0.4880**	16	2226.2	1	0.3239
			16	2418.9	0					
			17	2209.4	1					
		test150-0-0-0-0.d1.tw4	12	2004.6	1	**0.3263**	11	1941.8	2	0.2050
			11	1623.4	2		12	1540.3	2	
		test150-0-0-0-0.d2.tw0	18	2403.1	0	**0.4607**	18	2459.3	0	0.4145
			19	2313.5	0					
		test150-0-0-0-0.d2.tw1	18	2503.5	0	0.5245	19	2161.4	0	0.6029
							18	2886.7	0	
		test150-0-0-0-0.d2.tw2	17	2500.9	3	**0.3449**	17	2232.3	3	0.3215
			17	2451.7	4		18	2029.8	4	
			18	2897.8	2					
			18	2373.5	3					
		test150-0-0-0-0.d2.tw3	20	2511.9	0	0.5404	18	2296.5	0	0.6360
			18	2856.2	0		19	2263.2	0	
		test150-0-0-0-0.d2.tw4	18	2346.8	1	0.2126	19	2965.3	0	0.5256
							18	2438.4	1	
							20	2685	0	
		Count		30				**34**		
		Average				0.3977				**0.4137**

Appendix A43: Comparison with ALNS's hypervolume based on the 150 customers and 30% DoD

Customer Size	DoD	Instance	NV	TD	RR	HV	NV	TD	RR	HV
150	30	test150-0-0-0-0.d0.tw0	8	1408.8	0	**0.4464**	7	1117.6	1	0.2208

	7	1281.2	1					
	8	1214.6	1					
test150-0-0-0-0.d0.tw1	10	2245.2	0	0.5481	10	1923.5	1	**0.5557**
	11	1946.9	0		10	2078.8	0	
	10	2235.9	1		10	1877.9	2	
	10	1992.8	2					
test150-0-0-0-0.d0.tw2	11	1875.6	6	0.2306	11	1721.3	6	**0.2949**
	11	1896.4	5		11	2327.3	4	
					11	1728.7	5	
test150-0-0-0-0.d0.tw3	15	2138.9	4	**0.1313**	15	2194.6	4	0.1250
test150-0-0-0-0.d0.tw4	12	1809.4	2	**0.2685**	12	2115	2	0.2339
					12	1899.1	3	
test150-0-0-0-0.d1.tw0	7	1180.3	1	**0.6122**	7	1146.5	1	0.5797
	8	1221.7	0		8	1464.3	0	
	8	1144.9	1		7	1115.9	2	
test150-0-0-0-0.d1.tw1	9	1798.3	6	**0.3647**	9	1715.2	4	0.3599
	10	1787.8	6					
	10	1933.1	5					
	10	1993.7	3					
	9	2198.6	5					
test150-0-0-0-0.d1.tw2	13	1648.1	2	0.5804	13	2105.9	0	**0.6177**
	12	2094.8	1		12	1917.6	2	
	13	2675.6	0		13	1983.9	1	
					13	1913	2	
					12	2382.2	1	
test150-0-0-0-0.d1.tw3	13	1980.7	10	0.2113	14	1994.3	8	**0.2542**
	13	2302.4	9		13	1915.8	10	
test150-0-0-0-0.d1.tw4	11	2108.3	4	0.2779	11	2056.7	6	**0.3615**
	12	2078.6	5		12	1918.6	3	
					13	1913.8	4	
test150-0-0-0-0.d2.tw0	18	2569	1	**0.2961**	17	2823.5	2	0.2758
	18	2557.3	2		18	2756.4	1	
test150-0-0-0-0.d2.tw1	18	2667.5	0	**0.5165**	18	2544.5	2	0.4587
	18	2622.5	3		18	3184.2	0	
	17	2915.2	3					
test150-0-0-0-0.d2.tw2	18	2900.8	1	**0.5050**	19	2846.6	1	0.4503
	19	2697.4	0		19	2984.1	0	
					20	2838.2	0	
test150-0-0-0-0.d2.tw3	18	2917.1	7	0.1382	18	2493.8	7	**0.2644**
					18	2679.7	6	
					19	2731.2	5	
test150-0-0-0-0.d2.tw4	18	2657.1	7	0.2040	17	2568.1	8	**0.2068**
	17	2510.2	8		18	2431.7	8	
					18	2817.5	7	
Count		36				**37**		
Average				0.3554				**0.3599**

Appendix A44: Comparison with ALNS's hypervolume based on the 150 customers and 50% DoD

Customer Size	DoD	Instance	NV	TD	RR	HV	NV	TD	RR	HV
			7	1324.2	2		8	1297.1	0	
		test150-0-0-0-0.d0.tw0	7	1354.5	0	**0.5787**	7	1507	1	0.5601
			7	1302	3		7	1320.1	3	
		test150-0-0-0-0.d0.tw1	10	2403.6	1		10	2438.7	2	
						0.5265	9	2394.2	3	0.4545
		test150-0-0-0-0.d0.tw1	9	2127.6	3		10	3399	1	
							10	2365.6	3	
							13	1922.2	7	
							10	2238.2	8	
		test150-0-0-0-0.d0.tw2	11	1917.3	6	0.3155	13	2060.4	5	0.3381
							11	2015.7	8	
							15	2155.7	9	
		test150-0-0-0-0.d0.tw3	13	2118.5	11	0.2627	14	2387.4	12	0.2846
							14	2684.9	11	
			12	2033.1	4		12	2340.8	1	
			12	2127	2		13	2034.2	3	
		test150-0-0-0-0.d0.tw4	11	2052.9	4	0.5508	11	1917.4	4	0.5954
			12	1818.9	5		13	2278	2	
			15	1908.2	3					
			8	1392.3	1		7	1296.4	2	
		test150-0-0-0-0.d1.tw0	7	1320.5	2	0.3822	8	1329.5	0	0.4780
			7	1268.9	3					
			9	2098.7	6		9	2074.4	6	
		test150-0-0-0-0.d1.tw1	10	2214.9	1	**0.3994**	9	2136.2	5	0.2302
			9	2119.4	4		10	2006.7	6	
			10	2112.2	8		12	2321.2	4	
			11	2197.8	7		14	2124.8	7	
		test150-0-0-0-0.d1.tw2				0.3303	11	2276.9	6	0.4064
			11	2205.9	6		10	2131.1	8	
150	50		15	2377.2	4		16	2587.7	4	
							15	3275.6	3	
		test150-0-0-0-0.d1.tw3	15	2755.1	3	**0.3790**	15	2330	5	0.3567
							15	2920.5	4	
			11	1984.1	5		12	1840.1	7	
			10	1966	7		11	1949.5	6	
		test150-0-0-0-0.d1.tw4				**0.3739**	11	1890.1	7	0.3218
			11	1901	6		10	2295.8	8	
			17	2518.1	3		18	2816.8	0	
			18	3111.4	0					
		test150-0-0-0-0.d2.tw0	17	2658.9	2	**0.5043**	17	2879.2	3	0.4700
			17	2497.3	4					
			17	3253.8	5		17	3187.4	5	
			18	3169.6	3		18	3055.4	3	
		test150-0-0-0-0.d2.tw1	19	2862.8	2	0.4256	19	3503.6	1	0.4398
			18	3223.4	2		19	3085.7	2	
			18	3086.8	5					
			19	2862.8	2		18	3174.4	6	
		test150-0-0-0-0.d2.tw2	18	2895.4	6	**0.4516**	20	3366.8	4	0.2763
			19	3481.6	4		19	3275.3	5	
			19	3030.4	5		19	3203.5	5	
			21	2950	6		23	3382.7	3	
		test150-0-0-0-0.d2.tw3	21	2981.4	5	0.3393	20	2932.8	5	0.3719
			20	3158.5	4		20	2863.9	6	
							19	3198.7	6	
			20	2919.5	3		18	2762.4	6	
		test150-0-0-0-0.d2.tw4	19	3194.1	5					0.3983
			19	2893.6	6	0.3146	19	3099.8	5	
			20	2897.3	5					

Count	46		**51**	
Average		**0.4157**		0.3988

Appendix A45: Comparison with ALNS's hypervolume based on the 150 customers and 70% DoD

Customer Size	DoD	Inst	NV	TD	RR	HV	NV	TD	RR	HV
150	70	test150-0-0-0-0.d0.tw0	7	1660.2	3	0.5716	7	1660.2	3	0.5716
			7	1731.6	1		7	1731.6	1	
			7	1597.4	6		7	1597.4	6	
			8	1554.7	1		8	1554.7	1	
			8	1544.9	3		8	1544.9	3	
			7	1789.7	0		7	1789.7	0	
		test150-0-0-0-0.d0.tw1	9	2372.9	5	**0.4195**	10	2484.6	3	0.3916
			10	2619.1	2		9	2252.4	7	
			10	2259.4	6		9	2687.6	3	
			9	2256.4	7					
		test150-0-0-0-0.d0.tw2	13	2323.2	8	0.3124	14	2389.8	8	**0.3734**
			16	2553.6	7		15	2161.1	7	
			15	2712.3	7		13	2401.4	8	
							18	2822.2	6	
		test150-0-0-0-0.d0.tw3	14	2371.2	10	**0.3210**	15	2948.3	10	0.2607
			13	2979.4	12		16	2620.3	11	
			14	2256.6	12		14	2570.9	12	
							15	2757	11	
							16	2465.3	12	
		test150-0-0-0-0.d0.tw4	12	2198.3	8	0.3431	11	2048.1	11	**0.3785**
			10	2639.1	11		12	2346.5	6	
			11	2760.3	8		12	2199.3	7	
			12	2080.4	9					
		test150-0-0-0-0.d1.tw0	7	1488.3	0	**0.5888**	7	1448.1	7	0.5016
			7	1233.2	8		7	1385.1	8	
			7	1321.9	7		7	1591.4	2	
							8	1593.5	0	
		test150-0-0-0-0.d1.tw1	8	2180.2	8	**0.6247**	10	2641.6	4	0.5846
			11	2894.4	3		11	2589	7	
			9	2249.7	7		12	2745.7	2	
			11	2538.2	5		14	2361	8	
							13	2563.9	4	
		test150-0-0-0-0.d1.tw2	13	2182.2	12	0.3271	14	2182.3	6	0.3896
			14	2285.4	9		12	2177.7	12	
			11	2620.2	10		13	2214.3	10	
			12	2294.6	12					
		test150-0-0-0-0.d1.tw3	20	2648.3	13	0.2083	19	2932.6	12	0.2261
			21	2559.2	13		20	2766.4	13	
			19	2825.1	14		18	2744.1	14	
		test150-0-0-0-0.d1.tw4	12	2647.7	2	**0.5883**	12	2653	11	0.4708
			11	2677.6	4		11	2417	12	
			12	2642.9	14		11	2686.8	9	
							13	3197.1	7	
							10	2409.3	14	
		test150-0-0-0-0.d2.tw0	18	3097.4	3	**0.6015**	17	2900.6	9	0.5507
			17	2901.9	4		18	3506	5	
			17	2680.1	6		20	3116.2	1	
			20	3213	1		17	2998.3	6	

test150-0-0-0-0.d2.tw1	19	3336	3	0.4833	18	2916.8	8	0.5858
	17	3005.6	8		18	2719.4	9	
					18	3688.1	6	
					19	3417.1	5	
					17	2946.9	8	
					21	3728.5	0	
					19	3845.6	1	
test150-0-0-0-0.d2.tw2	18	2912.2	5	**0.2477**	19	3348.7	5	0.1972
	18	2724.6	6		19	3116.8	6	
					18	3340.3	6	
test150-0-0-0-0.d2.tw3	18	3119.6	12	0.1552	18	2846.9	12	0.2110
	18	3171.3	11		18	3230.8	10	
					17	3150	12	
test150-0-0-0-0.d2.tw4	15	2554.6	3	**0.5541**	20	2910.2	9	0.4961
test150-0-0-0-0.d2.tw4	13	2684.7	5		19	3070	8	
test150-0-0-0-0.d2.tw4	19	2936.3	0		16	3117.7	11	
					19	3140.3	7	
					21	3623.2	3	
					20	3159.1	6	
Count		50				**64**		
Average				**0.4231**				0.4126

Appendix A46: Comparison with ALNS Hypervolume based on the 150 customers and 90% DoD

Customer Size	DoD	Instance	NV	TD	RR	HV	NV	TD	RR	HV
			7	1710.1	4		8	2212.5	0	
		test150-0-0-0-0.d0.tw0	7	1737.4	0	**0.7059**	7	1512.8	4	0.6253
			8	1698.1	0		7	2269.5	3	
			7	1676.6	5		8	1923.8	1	
			17	2863.5	2		14	2680.4	4	
		test150-0-0-0-0.d0.tw1	16	3005	6	0.4381	13	2702.6	5	0.5754
			14	3349.3	5		16	3180.7	1	
			22	3260.8	3		19	2647.5	10	
			22	2732.5	9		22	2747.4	7	
		test150-0-0-0-0.d0.tw2	22	2693.9	11	**0.4048**	21	3486.8	8	0.3798
			22	2979.6	8		22	2747.4	9	
			22	2726.3	10					
			18	2788.7	12		19	2723.9	13	
		test150-0-0-0-0.d0.tw3	19	3340.1	11	0.2313	17	3039.9	13	0.2503
							18	2952.5	12	
							19	3165	11	
			13	2428.7	10		12	2435.9	15	
			13	2056.4	17		9	2808.5	15	
		test150-0-0-0-0.d0.tw4	10	2168.9	15	**0.4481**	11	2764.8	11	0.393
			11	2533	12		10	2620.1	13	
							11	2657.3	12	
			8	1479.8	0		7	1672.5	2	
							8	1389.5	2	
		test150-0-0-0-0.d1.tw0	7	1698.5	3	0.8154	10	1753	0	0.8317
							7	1963.4	0	
							8	1770	0	
150	90						11	2197.2	2	
		test150-0-0-0-0.d1.tw1	11	2236.3	0	**0.5694**	11	2097.5	5	0.4963
							14	2308.9	1	
			16	2706.7	4		16	2706.7	4	
		test150-0-0-0-0.d1.tw2	18	2621.3	5	**0.4062**	18	2621.3	5	**0.4062**
			19	3180.1	2		19	3180.1	2	
			16	2653.6	10		15	3121.5	7	
		test150-0-0-0-0.d1.tw3	16	2902.5	8	0.2631	14	3139.7	9	0.3054
			15	2972.1	11		13	2917	11	
			9	2467	17		9	2674.9	14	
		test150-0-0-0-0.d1.tw4	10	2658.3	13	0.2407	10	2267.4	13	0.2799
			9	2586.2	15		9	2375.8	16	
			16	2653.6	10		16	2747.1	11	
			16	2902.5	8		16	2997.5	10	
		test150-0-0-0-0.d2.tw0				0.3738	16	2643.5	13	0.4225
			15	2972.1	11		17	2725.7	8	
							17	3402.7	4	
			15	3889.2	5		17	4001.4	6	
			16	3815.1	7		19	3850.1	4	
			18	3760.1	6		22	4110.6	2	
		test150-0-0-0-0.d2.tw1	19	3850.1	4	**0.4993**	18	3698.3	7	0.4496
			22	4110.6	2					
			18	3698.3	7		17	3814.8	7	
			17	3814.8	7					
							20	3389.7	10	
		test150-0-0-0-0.d2.tw2	20	3066	8	0.2585	21	3262.5	10	0.2693
							21	3713	6	

		test150-0-0-0-0.d2.tw3	22	2849.3	15	**0.2722**	20	2848.5	15	0.2511
			21	3176.1	14		23	3369.3	13	
			22	3180.9	11		19	3284.9	15	
			16	3117.7	11		19	3993.1	12	
		test150-0-0-0-0.d2.tw4	18	3698.3	7	**0.458**	19	3629.7	13	0.3552
							19	3187.4	14	
							20	3049.7	12	
							20	3274.5	8	
Count				46				**58**		
Average						**0.4256**				0.4194
Overall Count				208				244		
Overall Average						**0.4035**				0.4009

Appendix A47: Comparison with ALNS's hypervolume based on the 250 customers and 10% DoD

Customer Size	**DoD**	**Proposed Algorithm**					**ALNS (2018)**			
		Instance	**NV**	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
250	10	test250-0-0-0-0.d0.tw0	11	1521.3	0	**0.5170**	12	1433.6	1	0.3425
							11	1673.4	2	
		test250-0-0-0-0.d0.tw1	18	2876.8	1	**0.4452**	17	2589.3	1	0.3758
			19	2735.8	0		19	2447.9	2	
			19	2653.6	2		18	2585.5	1	
		test250-0-0-0-0.d0.tw2	20	2598.3	4	**0.3492**	20	2586.5	5	0.3305
			20	3175	3		22	3233	3	
							20	2832.3	4	
		test250-0-0-0-0.d0.tw3	27	3911.8	3	0.5162	28	3652.1	1	**0.5368**
			31	3454.9	1		29	3261	2	
			30	3788.6	1		30	3267.7	1	
			27	4513	1					
		test250-0-0-0-0.d0.tw4	20	2826.9	2	0.4249	17	2746.1	2	**0.6032**
			19	2932.1	3		19	3989	1	
			19	2836.9	4		18	2684.5	3	
		test250-0-0-0-0.d1.tw0	11	1613.3	0	0.4375	12	1555	0	**0.4740**
							11	1519.7	1	
							11	1583.9	0	
		test250-0-0-0-0.d1.tw1	17	2570.8	1	0.4946	18	2453.1	0	**0.5163**
			17	2936.4	0		17	2520	1	
			18	2682	0					
		test250-0-0-0-0.d1.tw2	21	3315.7	3	**0.3792**	22	2877.5	3	0.2174
			22	3650.1	1		20	3570.4	3	
			23	2959.8	3		21	3351.8	3	
		test250-0-0-0-0.d1.tw3	27	3809.2	4	0.5178	27	3829.7	4	**0.5244**
			28	4215.1	1		27	4626.8	3	
			31	3629.9	2		28	3780.2	1	
		test250-0-0-0-0.d1.tw4	18	3139	1	0.4650	17	2855.8	1	0.6111
			19	2815.8	1		18	2590	1	
			20	2731.9	1		19	2961.4	0	
			21	3330.6	0					
		test250-0-0-0-0.d2.tw0	19	2949.5	1	0.5661	19	3979.7	0	0.3950
			19	2973.9	0		19	3662	1	
		test250-0-0-0-0.d2.tw1	20	3080.7	1	0.1508	21	2829.1	1	0.1689
							20	3244.8	1	
		test250-0-0-0-0.d2.tw2	22	3287	3	0.3181	20	2845.3	3	0.5701
			23	3264.1	3		21	3591.2	1	
			20	3113.1	4		21	3027.5	2	

			25	3042.3	4		21	2761.5	4	
		test250-0-0-0-0.d2.tw3	28	3524.7	0	**0.4009**	29	3363.9	0	0.4092
		test250-0-0-0-0.d2.tw4	21	2977.9	2	**0.4670**	20	3221.1	2	0.2382
			23	3914.7	0		21	2891.5	2	
		Count		37				**39**		
		Average				**0.4300**		0.4209		

Appendix A48: Comparison with ALNS's hypervolume based on the 250 customers and 30% DoD

Customer Size	DoD	Instance	Proposed Algorithm				ALNS			
			NV	TD	RR	HV	NV	TD	RR	HV
							11	1957.3	0	
		test250-0-0-0-0.d0.tw0	11	1739.7	0	0.5994	12	1818.5	0	0.6282
							13	1619	0	
			17	2792.1	4		17	2874.7	0	
		test250-0-0-0-0.d0.tw1	16	2909.9	4	0.2249	16	2752.4	4	0.4463
			17	3052.7	3					
			20	3160.8	8		20	2940.9	9	
		test250-0-0-0-0.d0.tw2	19	2854.9	9	0.3653	22	2925.8	6	0.3653
			21	2896.5	6		21	3698.6	5	
		test250-0-0-0-0.d0.tw3	25	3599.2	9	**0.1792**	25	3958.2	9	0.1500
			25	3567.7	10					
			17	3131.5	5		19	2848.1	3	
			18	3059.4	4		19	2824.1	6	
		test250-0-0-0-0.d0.tw4	18	2687.3	5	**0.4056**				0.3200
			18	3060.3	2		18	3069.7	4	
			19	2791.7	4					
			11	1762.9	1		12	1836.9	1	
		test250-0-0-0-0.d1.tw0	11	1689	2	0.4217	13	1806	0	0.4927
			12	1566.6	2		14	1754.4	1	
			16	2808.8	5		16	2882.8	5	
		test250-0-0-0-0.d1.tw1	16	3145.9	3	0.2832	17	2800.8	5	0.3205
250	30						17	3157.8	2	
			20	3330.2	7		23	3154.3	5	
			21	3394.2	6		20	3723.1	5	
			21	4006.7	4		24	3208.9	1	
			23	4286.3	2		23	3901.6	2	
		test250-0-0-0-0.d1.tw2	22	3614.2	3	0.5315	23	3349.9	4	0.6223
							23	3508.9	3	
			22	3228.7	5		21	3398	5	
							22	3847.7	4	
		test250-0-0-0-0.d1.tw3	26	3987.9	6	**0.1984**	26	3618.4	6	0.1543
			26	3605.2	7		27	3610	6	
			17	3227.4	6		18	3025.2	2	
			18	3358.7	3		18	2966.7	6	
		test250-0-0-0-0.d1.tw4	19	2809.6	6	0.4183	17	3098	6	0.5182
			18	3099.9	6		17	3025.1	7	
							19	3674.1	1	
			19	2886.7	10		18	4017.1	7	
			18	3148.5	5		18	3716.8	8	
		test250-0-0-0-0.d2.tw0	18	3132.7	9	**0.5635**				0.2936
			18	3663.9	4		18	4179.3	6	
			19	3517.7	1					
		test250-0-0-0-0.d2.tw1	19	3595.6	5	**0.4433**	19	4812.6	4	0.4080
			20	3262.7	4		22	3997.1	3	

							20	3496	4	
			21	3,415.80	3		20	3494.1	5	
							19	3751.8	5	
			21	3287.8	4		23	3506.9	4	
			22	3373.5	2		25	3151.8	4	
			23	3143.5	4		24	3417	2	
			21	3582.9	3					
			25	3334.4	9		25	3559.1	9	
		test250-0-0-0-0.d2.tw3	25	3925.1	8	0.2050	26	3542.2	8	0.2575
							26	3642.7	6	
		test250-0-0-0-0.d2.tw4	20	2973.2	7	0.2272	21	3395.5	6	0.2407
			21	3613.1	6		20	3688.7	5	
		Count	47				**49**			
		Average	0.3619				**0.3895**			

Appendix A49: Comparison with ALNS's hypervolume based on the 250 customers and 50% DoD

Customer Size	DoD	Instance	Proposed Algorithm				ALNS(2018)			
			NV	TD	RR	HV	NV	TD	RR	HV
		test250-0-0-0-0.d0.tw0	11	1929.3	7		11	1894.1	10	
		test250-0-0-0-0.d0.tw0	12	2101	3		12	1919.9	2	
							12	1876.6	9	
						0.5053	11	1935.2	9	0.5703
		test250-0-0-0-0.d0.tw0	12	2119.1	1		11	1951.4	2	
							13	1691.3	13	
							13	1768.8	6	
		test250-0-0-0-0.d0.tw1	14	2799.5	13		14	3057	13	
							15	2757.3	13	
							20	3847.2	8	
						0.364	15	2882.3	12	0.5483
		test250-0-0-0-0.d0.tw1	15	3085.1	12		16	2634.5	13	
							17	2935.7	9	
							21	3402.6	7	
			21	3292.3	5		22	4003.3	1	
			22	4173.6	3		19	3309.7	8	
		test250-0-0-0-0.d0.tw2	19	3302.7	9	0.5222	21	3072.6	7	0.5944
250	50		24	3622.1	4		20	3518.5	7	
			22	3847	4		20	3262.3	9	
			23	3915.5	14		24	4475.8	12	
		test250-0-0-0-0.d0.tw3	24	4046	13	0.238	24	3789.4	14	0.2414
			24	3849.1	14		23	3905.1	14	
			24	4958	12					
			18	3282	9		19	3361.4	8	
			18	2911.5	11	0.284	18	3212.9	9	0.2937
			17	3792.2	9		17	3273	10	
		test250-0-0-0-0.d0.tw4	11	1929.5	2		11	2095	2	
						0.4748	12	2033	0	0.5061
			12	2118	0		12	1975.9	2	
							11	2275.4	1	
		test250-0-0-0-0.d1.tw1	15	2918.6	9		15	2801	10	
							16	3503.1	5	
						0.2676	16	3167.3	7	0.3658
		test250-0-0-0-0.d1.tw1	15	2793.9	10		15	2742.6	11	
							16	3044.3	9	
		test250-0-0-0-0.d1.tw2	18	3438.8	12	0.2819	18	3375.7	12	0.289

		test250-0-0-0-0.d1.tw3	20	3624	7	0.1354	20	3626.8	7	0.1964
			20	3384.4	9		19	3340.5	10	
			23	3937.6	14		24	3645.2	12	
							23	3678.2	14	
		test250-0-0-0-0.d1.tw4	19	3235.2	9	**0.3475**	17	3328.2	10	0.2438
			17	3084.4	11		17	3306.6	13	
			17	2907.2	13		18	3242.9	12	
			18	3441.7	7					
			19	2985.8	12					
			19	3245.7	8					
		test250-0-0-0-0.d2.tw0	18	3176.2	11	**0.5291**	18	3605.6	8	0.3961
			17	2741.5	17		18	3577	11	
			18	3207.5	7		19	3620.8	4	
			19	3788.7	2		19	3752.8	3	
			18	3103.2	12		18	3773.6	7	
			18	2736.9	16					
		test250-0-0-0-0.d2.tw1	19	3788.2	13	0.3285	19	3326.5	13	0.3822
			20	3496.2	11		20	3588.1	11	
			22	4191.4	8		20	4036.4	8	
		test250-0-0-0-0.d2.tw2	21	3507.4	15	0.2405	21	3694	15	0.3565
			22	3640	14		20	4242.2	15	
			20	3672.2	15					
			22	3780.3	13					
			22	3493.5	15					
		test250-0-0-0-0.d2.tw3	23	3928.2	15	0.1887	23	3858.4	15	0.2154
			24	4310.5	14		24	4053.7	14	
							24	4265.1	13	
							25	4071.3	13	
		test250-0-0-0-0.d2.tw4	24	3907.7	5	0.3572	22	3773.2	10	0.4151
			22	3517.7	9		24	4024.5	4	
			25	3748.1	5		23	3960.4	5	
			23	3889.1	8		25	3842.9	2	
							23	3946.2	8	
Count				60				**61**		
Average				**0.4376**				0.3737		

Appendix A50: Comparison with ALNS's hypervolume based on the 250 customers and 70% DoD

Customer Size	**DoD**	**Instance**	**Proposed Algorithm**				**ALNS (2018)**			
			NV	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
250	70	test250-0-0-0-0.d0.tw0	12	2153.7	0	**0.8499**	14	1844	2	0.7852
			13	1890.1	5		12	1993.5	4	
			11	2188.9	0		11	2100.3	4	
			11	2157.5	3		12	2915.2	0	
			11	2140	6		11	1902.5	6	
		test250-0-0-0-0.d0.tw1	16	3351.2	12	0.4312	16	3697.8	12	0.4598
			16	3389	11		16	3416.4	13	
			18	3386.8	9		17	4111.5	5	
			14	4175.9	13		17	3458	9	
			19	3508.7	8					
			15	3633.4	12					
		test250-0-0-0-0.d0.tw2	15	3999.8	9	0.5389	18	3723.3	6	0.6184
			27	4185.9	7		18	3527.5	16	
			17	3695.8	16		21	3766.4	9	
			22	3484.6	12		23	4295.7	5	

	22	3428.5	13		23	3632.8	8	
	20	3911.5	12		22	3431.2	10	
	24	3312.8	13		22	3914	6	
					25	3763.6	7	
					21	3346	14	
	27	3665.4	8		21	3655.9	13	
	27	3755.5	17		28	4340	15	
	25	4095	18		28	3998.1	17	
	27	4195.5	13		26	4226.6	18	
test250-0-0-0-0.d0.tw3				**0.2953**	29	4414.6	12	0.2718
					29	4118.8	16	
	28	4071	16		28	4343.1	13	
	18	3415.9	11		19	3700.1	9	
	17	3459.2	10		18	3740.5	11	
	18	4005.6	7		20	3940.5	7	
test250-0-0-0-0.d0.tw4	18	3690.8	9	0.4214	20	4086.7	5	0.4378
					19	3760.8	8	
	19	3218.8	11		18	3936.8	10	
					20	4747.4	4	
	11	2072.1	2		11	2040.3	10	
	11	2044.6	13		11	2376.2	3	
					13	2348	0	
test250-0-0-0-0.d1.tw0				**0.6011**	12	2031.6	9	0.5882
	12	2370.3	0		11	2262.5	9	
					12	2152.9	1	
	18	3721.8	1		19	3276.2	3	
	18	3312.8	4		18	3579.5	5	
	21	3156.9	2	**0.7803**	20	3588.8	2	0.6608
	24	3867.5	0		17	4137.2	4	
test250-0-0-0-0.d1.tw1	17	3492.4	7		16	3619.8	6	
test250-0-0-0-0.d1.tw2	23	3855.2	6		21	3940.7	7	
test250-0-0-0-0.d1.tw2	21	3324.8	9	**0.4452**	21	3761.2	10	0.4039
					27	3901.1	9	
test250-0-0-0-0.d1.tw2	19	3805.9	8		28	4018.8	4	
	27	3723	16		24	3844.7	16	
	29	4585.1	12	0.2703	28	4034.7	13	0.2872
test250-0-0-0-0.d1.tw3	26	4569.6	15		27	4056.3	14	
	17	3714.7	14		18	3528.9	9	
	18	3069.1	9		16	3220.7	12	
	18	3602.1	8	0.4218	19	3471.2	10	0.4316
	19	4030.1	7					
test250-0-0-0-0.d1.tw4	18	2930.3	10		19	3779.9	6	
	18	3266.3	5		18	2832.9	12	
	18	3045.7	7		18	3930.1	10	
test250-0-0-0-0.d2.tw0	18	2792.2	10	0.6603	19	3028	3	0.6488
					18	4147.6	7	
	20	3364.5	0		19	3950.8	0	
	24	4329	5		21	5531.2	9	
	21	3781.3	11		21	4377.7	10	
test250-0-0-0-0.d2.tw1	21	3977.8	10	0.4788	22	4471.8	9	0.3402
					20	5499.1	11	
	21	4339.10	8		22	5231.3	8	
	24	4014.3	12		24	4332.1	10	
	25	4288	11		25	3560.8	14	
test250-0-0-0-0.d2.tw2	23	4539.8	13	0.3307	26	4304.4	11	0.3672
	25	3898.5	13		23	3942.2	15	
	26	3749.4	15		25	3983	13	
	28	4200.5	11					
test250-0-0-0-0.d2.tw3	24	3882	21	0.2700	26	3823.2	19	0.2752
test250-0-0-0-0.d2.tw3	23	4086.4	24		25	4012.8	20	

		test250-0-0-0-0.d2.tw3	23	4096.9	22		27	4531.2	18	
		test250-0-0-0-0.d2.tw3	24	3736.6	22					
		test250-0-0-0-0.d2.tw4	23	3928.8	16	0.2465	24	4612.5	14	0.2993
			24	4029.8	15		24	4637.5	11	
							23	4442.6	16	
							25	4090.1	17	
							25	4203	14	
			26	3944.7	15		24	4536	15	
Count			68				**78**			
Average			**0.4694**				0.4584			

Appendix A51: Comparison with ALNS's hypervolume based on the 250 customers and 90% DoD

Customer Size	DoD	Instance	Proposed Algorithm				ALNS(2018)			
			NV	**TD**	**RR**	**HV**	**NV**	**TD**	**RR**	**HV**
		test250-0-0-0-0.d0.tw0	11	2124.3	9	0.7115	11	2164.4	4	0.7804
			12	2867.3	0		12	2842.8	0	
			12	2491	5		11	1896.9	12	
			13	2139.7	6		14	2521.9	1	
			13	2648.9	0		13	2068.7	10	
		test250-0-0-0-0.d0.tw1	19	4030.5	10	0.6072	23	3593	12	0.6196
			16	3336	12		21	3774.7	13	
			20	3927.8	8		22	3923	9	
			14	3763	14		23	4985.6	1	
			18	4104.8	8		21	4020.2	8	
							16	4097.2	13	
		test250-0-0-0-0.d0.tw2	27	3889.2	15	**0.4781**	24	4077.7	11	0.4421
			25	4023.8	14		26	4073.6	16	
			25	4196.1	9		23	4536.9	9	
			25	4695	5		27	4512	9	
			22	4098.5	11		27	4530.8	8	
							26	4674.4	5	
250	90	test250-0-0-0-0.d0.tw3	33	4867.6	19	0.3038	34	4425.5	20	0.3528
			34	4071.5	22		31	4374.3	23	
							33	4786.6	22	
			34	4314.4	17		34	4707.4	18	
							35	4697.4	16	
							36	5138.3	12	
		test250-0-0-0-0.d0.tw4	18	3713	14	0.5864	18	4224.5	14	0.6097
			17	3787	18		18	4310.2	13	
			21	3738.8	12		17	3730.6	18	
			19	3940.3	12		18	3804.5	17	
			24	4688.2	11		24	4681.4	4	
			17	3983.5	16		19	3916.3	15	
			25	4943	6		19	4019.9	14	
							17	4504.8	14	
		test250-0-0-0-0.d1.tw0	11	1937.2	2	**0.6903**	12	2067.9	4	0.6362
			10	2235.5	18		11	2130	14	
			12	2254.4	0		13	2253.9	1	
							11	1887.2	16	
			12	1933.4	14		11	2164	5	
							14	2054	0	
		test250-0-0-0-0.d1.tw1	17	3806.6	6	0.5282	22	3715.9	2	0.5623
			24	3562.3	5		22	3867.9	2	
			20	4308.8	4		20	3973.2	3	

	19	4147.9	5		21	4221.3	1	
					19	3467.7	6	
test250-0-0-0-0.d1.tw2	27	4239.1	14		25	4666.4	8	
					28	4913.7	5	
					26	4229.7	18	
test250-0-0-0-0.d1.tw2	29	4678.2	10	0.3344	26	4145.7	19	0.4679
					27	4061	18	
					27	4330	15	
	32	4489	17		30	4556.5	17	
	30	4479.9	19		33	4643.4	15	
	31	4103.1	20		36	4593.4	16	
test250-0-0-0-0.d1.tw3				0.2555				0.2600
	30	4447.8	20					
	33	4425.4	19		36	4466.4	18	
	34	4367.6	19					
	16	3600.7	16		15	3440.3	18	
	17	3384.4	18		15	3436.8	20	
test250-0-0-0-0.d1.tw4	15	3392.4	23	0.3907	21	3893.6	12	0.4493
	15	3864.2	21		21	3845.6	17	
	16	3359.5	20		16	3113.6	21	
	18	3014.9	7		18	3034	9	
	17	2714.5	20		18	3146.3	7	
test250-0-0-0-0.d2.tw0	18	2854.1	8	**0.3908**	18	3232.1	5	0.3900
					18	2888.2	12	
	18	2710.1	13					
					18	2696.8	14	
	25	4168.8	2		36	5671.9	1	
	23	3921.1	6		35	6006	0	
test250-0-0-0-0.d2.tw1				**0.9336**	26	4425.4	2	0.8243
	33	4804.7	0		24	4436.6	5	
					27	4319.5	4	
	26	3745	17		22	4130.7	16	
	26	4340.1	12		26	4389.6	12	
	27	3831.7	16		25	3879.9	19	
	28	4091.8	15		25	4100	18	
test250-0-0-0-0.d2.tw2				**0.4314**				0.3745
	21	4112.7	16					
	28	3706.1	16					
					25	4204.3	14	
	25	4502.9	13					
	19	4056	20					
	29	4086.4	27		25	4100	18	
	32	4005.4	22		30	3743.2	27	
test250-0-0-0-0.d2.tw3				0.2681				0.3775
	26	4226.4	27					
					30	4068.9	25	
	30	4519	24					
test250-0-0-0-0.d2.tw4	31	4423	9		23	3829.1	15	
test250-0-0-0-0.d2.tw4	24	3911.7	16		30	4409.4	6	
test250-0-0-0-0.d2.tw4	31	4260.1	10					
				0.4778				0.5025
test250-0-0-0-0.d2.tw4	26	4550.1	11					
					23	4080.1	13	
test250-0-0-0-0.d2.tw4	27	4433.5	12					
test250-0-0-0-0.d2.tw4	31	4846.1	5					
Count		71				**78**		
Average		0.4925				**0.5148**		

Average 10 DoD (50,150,250 customers)	0.4201	**0.4208**
Average 30 DoD (50,150,250 customers)	0.3677	**0.3750**
Average 50 DoD (50,150,250 customers)	**0.3809**	0.3756
Average 70 DoD (50,150,250 customers)	**0.4089**	0.3989
Average 90 DoD (50,150,250 customers)	0.4153	**0.4243**
Average 50 customers (10%,30%, 50%,70% & 90% DoD)	**0.3807**	0.3756
Average 150 customers (10%,30%, 50%,70% & 90% DoD)	**0.4035**	0.4009

Average 250 customers (10%,30%, 50%,70% & 90% DoD)	**0.4383**	0.4315
Overall Average (50, 150 & 250 customers)	**0.4075**	0.4027