UNIVERSITY OF SOUTHAMPTON

FACULTY OF SOCIAL AND HUMAN SCIENCES

School of Mathematics

Vehicle Routing Problem with Availability Constraints

by

Farhana Johar

Thesis for the degree of Doctor of Philosophy

December 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF SOCIAL AND HUMAN SCIENCES

SCHOOL OF MATHEMATICS

Doctor of Philosophy

VEHICLE ROUTING PROBLEM WITH AVAILABILITY CONSTRAINTS

by Farhana Johar

This work is concerned with solving the vehicle routing problem (VRP) which takes into account the customer's release and due date. The problem studied can also be categorized as a non-classical VRP as the departure times of vehicles depend on the dates of orders released from the production line and become available for the distribution process. Hence, the problem is known as VRP with availability constraints (VRPAC).

The VRPAC is investigated through two stages. In the first stage, vehicle routing problem with release and due date (VRPRDD) is treated.At the beginning of the planning, it is assumed that the dates where the customer orders become available are known. A mathematical formulation is developed to represent the problem studied which has been solved by several heuristics, i.e. Variable Neighborhood Search (VNS), Large Neighborhood Search (LNS) and Tabu Search (TS). The algorithms are written in C++ and run on a PC computer with an Intel PentiumCore by using 56's Solomon instances with some modification. Different kinds of vehicle routing problem has been tackled in order to see the performance of proposed heuristics. The results are then compared in order to find the best method which yields the least routing cost solution. From the outcome obtained, VNS is proved to be the best algorithm which generates the least cost solution to our problem.

Further investigation has been carried out in stage two which considers the extension of VRPRDD. The coordination of production sequence and vehicle routing (PS-VRPRDD) is the main subject to our problem studied in which the best production sequence will leads to the least routing. Two proposed algorithms have been used to run the test instances. The first is classical decomposition approach; Alternate which decompose the problems into two sub-problems, i.e. production sequence and vehicle routing. This will be used as benchmark to the second approach; InOneMove which take these two decisions of the sub-problems as a whole. Decision on both sub-problems is considered simultaneously as one move. The results proved that effective coordination shows the large potential savings that attract the interest of industrial distributors in optimizing their distribution process in practice.

Contents

List of Figures

List of Tables

DECLARATION OF AUTHORSHIP

I, **Farhana Johar**, declare that the thesis entitled **Vehicle Routing Problem with Availability Constraints** and the work presented in the thesis are both my own, and have been generated by me as the result of my own research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- any part if this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations this thesis entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published.

Signed:
Date: ...

Acknowledgements

Thanks to Almighty Allah S.W.T. for graciously bestowing me the perseverance to undertake this research. Special thanks are due to University of Southampton for the opportunity to carry out research and to Universiti Teknologi Malaysia (UTM) for the financial support.

I would like to express my deep gratitude to my supervisors, Professor Chris Potts and Professor Julia Bennell for their invaluable advice and patient guidance, for the encouragement and useful critiques of this research work. This thesis would not be possible without generous support in all aspects of my graduate study in the few past years.

I would like to express my appreciation to the staff of Department of Mathematical Sciences, UTM, for all their help and support in completing my thesis writing.

I wish to express my heartfelt gratitude, appreciation and thanks to my beloved parents, my dear husband and my family, for a lifetime of love, concern, support and strength all these years; none of this would have been possible without them. I thank them for their unconditional and endless support to encourage and urge me to finish this thesis.

Last but not least, thanks to my friends, for all their support and for sharing me so often with my problems.

To my beloved parents, my dear husband and my family. . .

Chapter 1

Introduction

1.1 Background

Supply Chain Management (SCM) is the management of materials and information flows both in and between facilities, such as vendors, manufacturing and assembly, plants and distribution centers [129]. Over the past decades, SCM has received ever-growing interests both in the literature as well as from industrial practice [122],[23]. Generally, the chain consists of a network of organizations, people, activities, information and resources involved in the physical flow of products from suppliers to end customers [29], [46].

According to Fahimnia *et al.* [46] and Thomas and Griffin [129], there are three stages in the supply chain:

- Procurement: the acquisition of raw materials and parts from suppliers and their transportation to the manufacturing plants;
- Production: the transformation and/or assembly of the acquired materials into finished products;
- Distribution: a network for channelling materials to and between plants, and delivering products to end users through distribution centers.

Previously, organizations have focused their efforts on making effective decisions within each facility. However, under today's competitive environments, decisions become complex as numerous activities spread over multiple functions and organizations need to be taken into considerations. Due to the complexity within multiple activities, all supply chain stages must work towards a well-coordinated system and integrate with each other. Integration among raw-material suppliers, manufacturers and distributors leads to increased information flows, reduced uncertainties and a more profitable supply chain. Thus, it becomes crucial to develop an overall logistics process in order to provide goods and services to the customer at low cost and high service level. The effectiveness of integration in supply chain stages pose interesting challenges to the companies in order to maximize the potentials for converting competitive advantages into profitability.

1.2 Production-Distribution Chain

Kanda *et.al.* [69] reviewed various perspectives on supply chain integration issues, such as those related to integration of production and distribution, procurement and production, production and inventory, and distribution and inventory. Lee and Kim [79] asserted that production and the distribution planning are the main processes in supply chain.

Generally, a supply chain starts at the production level where the manufacturer transforms the raw materials and parts into finished product. Then, the distributor will deliver the products to the end customers. Thus, the cost of a product does not only include the material handling costs and inventory, but also the cost of delivering goods and servicing the end customers.

As distribution planning connects manufacturers to end users, distribution/transportation is of course the most important stage in supply chain. However, prior to distribution, production planning is a state just as important as the former in the supply chain. Production planning covers the design and management of the entire manufacturing process [79]. Each of the stages involves operation costs which may not be dependent on the other. Ertogal [45] and Wang [135] reported that production cost is the largest, followed by transportation and inventory cost. This study will only consider the costs for production and distribution. A recent review of the integrated production and distribution scheduling models in the literature can be found in Chen [27].

Traditionally, production scheduling and distribution planning have been dealt with independently in a sequential manner with little or no integration [27]. Production scheduling which focuses on determination of schedule for production will be developed first. Minimizing the production cost is done without considering the routing plan and vice versa.

However, due to market globalization nowadays, treating these components separately tends to result in substantial inefficiencies and, consequently, poor total system performance. Thus, in order to provide goods and services to customers at low costs but with high service levels, it seems that there is a strong need to overlay a coordinated system to have control of all functions in the supply chain [129],[69],[38]. Blumenfeld and Burns

[14], Park [100], Chen and Variraktarakis [28], Pundoor and Chen [109] and Coccola *et al.* [34] showed that there are significant benefits by using the optimal coordinated production-distribution schedule as compared to the schedule generated by a sequential approach in the context of the models they consider. Effective coordination within and beyond each boundary is needed to maximize the potential for converting competitive advantage into profitability or to minimize the total production and distribution costs.

Thus, the efficient coordination of production scheduling and distribution planning becomes a challenging problem as companies move towards higher collaborative and competitive environments. There have been many studies on this. Farahani *et.al.*[47] in catering food company, Buer *et.al.*[134], Russell *et.al.*[114], Chiang *et.al.* [31] in newspaper industry, Garcia and Lozano [52] in perishable product manufacturing, Li *et.al.*[82] in consumer electronics manufacturing and Marchetti *et al.* [89] in industrial gas supply-chains are some examples showing that coordinating production and distribution operations becomes crucial for satisfying on-time delivery requirements.

1.3 Make-To-Order (MTO)

Thomas and Griffin [129] estimated that over 30% of total cost of product can be reduced through MTO strategy. The concept has been widely used in order to lessen the overall logistics expenditure. For example, in cross docking logistics where the products from a supplier or manufacturer are distributed directly to a customer or retail chain with minimal handling or storage time. The activity is taking place in distribution docking terminal which goes through the process of receiving products through an inbound dock and then transferring them across the dock to the outbound transportation dock. Such approach, however, needs to comply with due dates since tardy deliveries may cause the recipients to struggle with meeting their own due dates. Another example can be seen in computer assembly production line where a wide variety of computers can be assembled according to the customer's order. Make-to-order manufacturing companies assemble finished products from partly finished products and customized components after receiving customer orders. Customized components required for one's computer is placed in a tray, which goes to assembly line. Once ready, it is available to the distribution process.

Motivated by the MTO business trend nowadays, this study considers the production-distribution system that consists of one manufacturer and one or more customers. Under MTO, the company manufactures the end product according to the customer's order. This type of business production strategy allows customers to purchase products that are made to their specifications and this gives more flexibility in customization compared to purchasing from retailers' shelves. The manufacturing company produces the products once orders are received. This is followed by the distribution process that starts as soon as the products are available. In this way, inventory, maintenance and warehousing which incur significant costs within the supply management flow could be reduced or avoided altogether.

Boudia *et al* [16] reported that distribution planning, i.e. vehicle routing problems have been widely investigated under various inventory constraints and many more, but the production decisions were usually ignored at these stages. In many studies, production planning has been discussed in detail but distribution is not taken into account.

At the beginning of planning, each customer places an order with the manufacturer. Next, the manufacturer processes the orders and attempt to distribute the goods to the customer. Each order has specified date by which the customer expects to receive his/her order. However, delivery may be delayed due to variability in production schedule which may not meet the due date. Preferably, orders are delivered shortly after their completion. This facilitates the delivery process and prevent delays. On the other hand, such individualized deliveries may increase the distribution cost. Hence, the supplier will try to consolidate the deliveries for separate orders as much as possible to minimize the total distribution cost. This means that some completed orders may have to wait for other orders to be completed so that they can be delivered in the same vehicle. This results in a tradeoff between delivery timeliness and total distribution cost. Our study will focus on finding the joint schedule of order processing and delivery to optimize the tradeoff of the delivery timeliness and distribution cost.

Pundoor and Chen [109], stated that the maximum tardiness and total tardiness of orders are the two commonly used measurements of delivery timeliness which represent respectively the worst and average service levels with respect to meeting the order's due dates. Scheduling problems with optimized maximum tardiness had been studied extensively in machine scheduling in which only production operation was considered. As such a problem deals only with the production part, there exists an optimal solution by scheduling the jobs in nondecreasing order of their due date on a single machine. Some related studies have other elements to be considered such as jobs release date, due date and batch set-up time [102].

Since our study focuses on integration of order processing and delivery schedule, we do not consider the problem at the scheduling level. However, we may consider batching together a set of orders for delivery to reduce the total distribution cost. Potts and Kovalyov [105] stated that it may be cheaper and faster to process jobs in a batch than to a process them individually, which in our case is related to order delivery.

1.4 Vehicle Routing and Production Scheduling

Vehicle Routing

The study of distribution of goods, known by the generic name vehicle routing problem (VRP), is one of the most important components of production and distribution of goods. VRP is a generalization of the travelling salesman problem (TSP). In TSP a salesman has to visit a number of cities. He/She has to start from, and return to, the same city. The objective is to create a route which is the shortest in distance.

According to Eksioglu *et al.* [44], the first paper recorded in TSP literature was written by Dantzig, Fulkerson and Johnson [36]. That seminal paper presented a relatively large scale TSP involving a certain tour of 49 cities, one in each of the 48 states and Washington D.C. Heuristics for solving the TSP can be found in Nilsson [95], Gendreau *et al.* [53] and Lin and Kernighan [85]. Among these, Lin-Kernighan's procedure which demonstrated the effectiveness of local improvement was generally considered to be one of the most effective methods for symmetric TSP [63]. Some latest paper of TSP had been reviewed by Ahmed [1] and Asim [8].

TSP has been extended to involve multiple salesmen. The basic problem is the same; all salesmen need to visit or serve the cities or customers that are assigned to them. Again, they start at and return to the same city which is now known as the depot. With more than one salesman, the problem is known as m-TSP or vehicle routing problem [75],[15].

Generally, VRP deals with the efficient distribution of goods from depot to many customers. The process begins from the depot in which a fleet of homogenous vehicles are used to deliver the goods to a number of customers using a set of routes subject to operational constraints. The operational constraints may be of various forms, such as the current load cannot exceed the capacity of the vehicle, customers only require delivery or collection of goods, or the customer has some time intervals to be served. In general, it is assumed that each vehicle follows one route, starting and ending at the depot. Another assumption is that each customer is only served once and by one vehicle. The aim of the study is to find the best vehicle routing to satisfy the customers' demand.

Researches on vehicle routing show the large potential savings that attract the interest of industrial distributors in optimizing their distribution process in practice. This scenario can be observed by the large number of tools and software for planning and optimizing distribution and transportation systems based on operational research techniques.

Due to the importance of vehicle routing especially in production and distribution process, many researchers from different backgrounds and disciplines such as Mathematics, Computer Programming, Operational Research and Business Management have involved themselves in this area. With advancement in computer technology today, there is room for further development techniques yielding faster and more powerful solutions for more complex systems.

Production Scheduling

Scheduling is an important tool of the productivity of a process. It is a decision-making process which makes a major impact especially in manufacturing industries. Typically, scheduling is about allocation of resources to tasks over given time periods. A task is a basic work entity associated with the start time, S_i and/or the end time, E_i, where the execution time is given by duration p_i or known as processing time. This could be an operation in a production process, execution of a computer programs and so on. A resource is a technical or human mean needed for the execution of a task and generally available in a limited quantity and capacity such as machines, man, and so on.The aim of scheduling is to minimize the production time and costs.

Graves [61] defined a production scheduling as the allocation of available production resources over time to best satisfy some set of criteria. The purpose of production scheduling is to maximize the efficiency of the operation whilst reducing the costs. Indepth reading on scheduling can be found in Pinedo [103]. The book contains all details in scheduling system starting from the basic scheduling such as the framework and notation used and classes of scheduling. Further studies on production scheduling can also be found in Lopez and Roubellat [88].

Generally, customer orders released in manufacturing setting are translated into jobs in production line. Each of the jobs is associated with the due dates, or maybe the processing time. These jobs will then be processed by machines to determine order or sequence. Some delays during the processing process of some jobs may happen due to certain problems, such as pre-emptions when high priority jobs have to be given preference. The machine may be too busy; breakdowns may happen, and some jobs may require longer-than-expected processing times. These need to be taken into account as they may have a major impact on the whole schedules. Thus, it is really important to have a control of this operation to maintain the efficiencies of the process.

In a manufacturing process, the scheduling function needs to have interactions with others decision making functions too. Once a schedule has been generated, the ready dates of all jobs will be determined where all the necessary raw materials and resources also will be made available at the specific time. The objective is to minimize either the completion time of last job that leave the system, maximum lateness or total weighted completion time.

Integrated Production and Distribution Scheduling

There have been many studies on integration of production scheduling and vehicle routing. Chen [29], [27] has extensively reviewed the issue of integrated production and vehicle routing scheduling. In such an integrated system, linkage between production schedule and delivery process is extremely important. Chang and Lee [23], Chen and Vairaktarakis[28] and Pundor and Chen [109] showed that there was significant benefit by using the optimal integrated production-distribution schedule. A recent survey by Ullrich [133] divided VRP literature into two main areas; studies that focus on rather 'simple' delivery considerations and others which allow for vehicle routing.

Most of the papers in literature focused more on production scheduling than vehicle routing. These include Li and Vairaktarakis [81], Pundor and Chen [109], Cakici *et al.* [21] and Farahani *et al.* [48]. Relatively few papers gave more attention to vehicle routing. Among these, Chang and Lee [23] considered a single machine scheduling with a limited capacity vehicle which was assigned up to two destinations only. Chen and Vairaktarakis [28] extended the problem with single and parallel production machine with infinite number of capacitated vehicles. Ullrich [133] dealt with parallel machine scheduling with a fleet of vehicles where customer's time window was the constraint of the problem.

To fill the gap, this study will consider a single production line, focusing more on vehicle routing. The routing of a fleet of vehicles will take into account the availability of the product with reference to the time in which the products are released from the production line, and the latest time at which products need to arrive safely at customer's location.

1.5 Motivation and Contribution of the Research

As mentioned previously, there are generally three stages in supply chain before goods are delivered to the end customers.

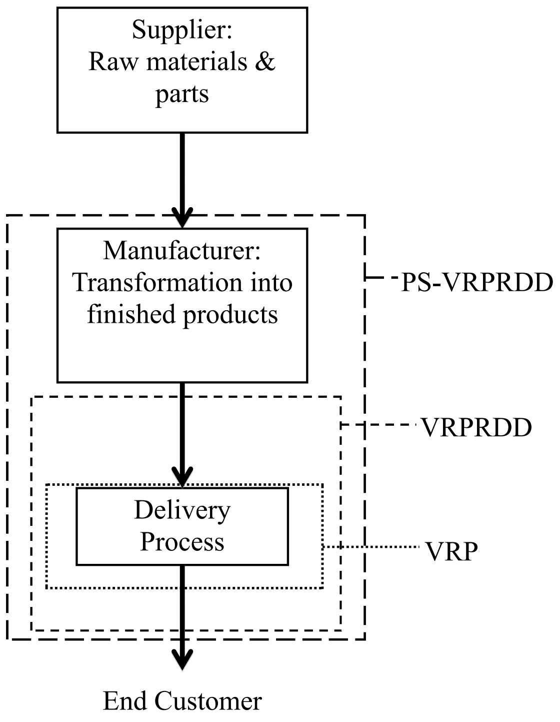


FIGURE 1.1: General Supply Chain Flow.

Referring to Figure 1.1, the manufacturer will transform the supplied raw materials into finished products. Once the finished products are ready, then delivery process will start to run. The standard VRP is working on this distribution section where it is concerned with designing the routes in order to fulfill an objective function and satisfy some constraints. In classical VRP, it is assume that all vehicles will leave the depot at the same time when orders are all ready to be dispatched. Customers need to be served before the due dates to avoid any lateness in the delivery process. However, in this study, taking from the vehicle routing perspective, we add a feature to standard VRP to make the problem more realistic. We shall consider the time of products to be released from the manufacturing process at different times. Thus, apart of the due dates, VRP now is taking into account the release dates of customer's order. The problem is known as Vehicle Routing Problem with Release and Due Dates (VRPRDD). Since in VRPRDD, we are dealing with the fixed release dates, then we extend the problem by looking at the production schedule as well which contributes to the determination of the release date. The problem is now known as Production Scheduling-VRPRDD (PS-VRPRDD).

Our research on vehicle routing will look at release and due date. Therefore, we deal with the vehicle routing problem with variable departure times of vehicle as these depend on the maximum release dates of orders scheduled on the routes. A customer can be served as soon as a vehicle arrives at customer's location. However, servicing should be done before the due date. Otherwise, penalty cost is imposed.

1.6 Objective of the Research

As explained in the previous section, the research deals with the availability of the orders at the start of the delivery process. Therefore, the objective of this research is to minimize the tradeoff of travelling and lateness cost. This can be done by solving the two problems starting with VRPRDD and then, PS-VRPRDD.

1.7 Outline of the thesis

In chapter 2, a more extensive review of VRP is provided since this is the main focus of our research. The chapter also includes variants of VRP since we will look into the different types of vehicle routing later. This will be followed by heuristics that are normally used to solve the vehicle routing problem in the literature and as the background to this research approach as well.

The research has been divided into two subproblems. The first problem will be VR-PRDD and then, PS-VRPRDD both of which will be described in Chapter 3. Chapter 3 also contains the problem description, formulation, data sets and problem design as well as modification of the instances. The VRPRDD then will be solved in Chapter 4 where heuristics used to solve the VRPRDD will be discussed in detail.

Chapter 5 deals with the second problem: PS-VRPRDD. This study is the extension of VRPRDD where improvement is made through the adjustment of the production sequence resulting in changes in release date of customer's order. This chapter will also look into two integrated approaches to develop the best algorithm for our problem. Results and discussions are presented for the proposed approaches.

Finally, conclusions and possible future works are given in Chapter 6. This chapter will summarize the work done in previous chapters.

Chapter 2

Review on Vehicle Routing Problem (VRP) and Its Solutions

2.1 Introduction

The chapter reviews the vehicle routing problem, its variants and heuristics for solution. The study of vehicle routing problem began more than 50 years ago. The first paper on VRP was presented by Dantzig and Ramser [37], giving the classical vehicle routing problem which was considered as a generalization of the TSP. Since than, a huge number of works dealing with VRPs had been published. Different variants may include some specific characteristics such as time windows, pick-up and delivery service, stochastic components and so on. This chapter describes and reviews some of the main VRP variants and heuristics in the literature.

2.2 Vehicle Routing Problem

Capacitated Vehicle Routing Problem (CVRP) which is considered as the classical version of VRP can be described as follow: Let $G = (V, E)$ be a complete graph with a set of vertices $V = \{0, ..., n\}$, where the vertex 0 represents the depot while others are customers. Each edge $\{i, j\} \in E$ has non-negative cost c_{ij} and each customer $i \in V' = v \backslash \{0\}$ has a demand d_i. Let $C = \{1, ..., m\}$ be the set of homogeneous vehicles with capacity Q. The objective is to determine a set of vehicle routes which satisfies all the customer demand with minimum overall cost or travelled distance.

In general, the existing VRP literature can be divided into theoretical study which focusing on the developments of the models or methods for the problem. Theoretical benchmark instances are used to measure the effectiveness of the proposed algorithms.

The other part of literature involving case-studies which dealing with real world applications such as in public library distribution system (Min [92]), solid waste collection (Kim *et al.* [71], Teixeira [128], Wy *et al.* [138]), or distribution in the perishable food industry (Cheng [30] [30], Hsu *et al.*[67], Chen *et al.* [24], Osvald and Stirn [97]).

Bodin *et al.* [15] provided a comprehensive review of the original vehicle routing problem, and published work related to algorithmic developments. More recent researches are found in the survey on the VRP and its variants presented by Kumar and Panneerselvam [72].

Below are some of the fundamental variants of the CVRP:

- VRP with Time Windows (VRPTW):
  Apart from the capacity constraints, each customer is associated with a time windows where the service should be done. According to Liu and Shen [87], Solomon [120] was the first to generalize VRP heuristics for solving VRPTW. Kim and Sahoo [71], Balakrishnan [9], Chen *et al.*[25], Min [92] and Tavakkoli-Moghaddam *et al.* [127] classified time window into two: soft (see Balakrishnan [9], Ioannou *et al.* [68], Xu *et al.* [139], Taş *et al.* [126]) and hard time windows (see Desrochers *et al.*[40], Liu and Shen [86], Chen *et al.* [26], Hong [66]). In soft time window (VRPSTW), a vehicle can break customer's time window with penalty cost included. Various penalty structures were described by Russell and Urban [115] as follows:

  - a constant penalty, reflecting a fixed compensation to the customer if punctuality was guaranteed (eg. a clause in a contract),
  - a penalty per unit time of early or tardy delivery (a linear loss function), and
  - a penalty based on the Taguchi loss function (a quadratic loss function).

  However, in VRP with hard time windows (VRPHTW), customer's time window has to be followed strictly[25].

  Comprehensive survey of VRPTW has been reported by Bräysy and Gendreau into two parts. The first part was on the traditional route construction heuristics [18] and the second part highlighted several important metaheuristics [19] used to solve VRPTW. Baldacci *et al.* [10] provides a recent exact algorithms for solving the vehicle routing problem with capacity and time window constraints. A summary of recent works on VRPTW can be found in Figliozzi [4].

- Split Delivery Vehicle Routing Problem (SDVRP)
  In SDVRP, customers may be served more than once and by more than one vehicle, contrary to what is usually assumed in the classical VRP. For this variant, each vehicle may deliver a fraction of customer's order. Therefore, demand of each

customer may be greater than the capacity of vehicle. According to Archetti *et al.* [7], SDVRP was introduced in the literature at 1989 and 1990 by Dror and Trudeu [41], [42]. Recent survey of SDVRP had been conducted by Archetti and Speranza [6]. The paper reviews the variants of SDVRP and in general all routing problems that consider split deliveries.

- Periodic Vehicle Routing Problem (PVRP)
  In classical VRPs, typically the planning period is a single day. In the case of the PVRP, the classical VRP is generalized by extending the planning period to a certain number of days. Thus a customer may request to be served more than once within the planning period. Francis*et al.* [51] provided a comprehensive survey on PVRP and its extension.

- VRP with Pickup-and-Delivery (VRPPD)
  In VRPPD, customers require simultaneous delivery and pick-up service. Deliveries are supplied from a single depot at the beginning of routing, while pick-up loads are taken to the same depot at the end of the service. VRPPD has received more attention from practitioners because of its great importance in practical applications and reverse logistics. The problem has been introduced by Min [91] to solve a problem of transporting books between library. Recent works on VRPPD can be found in Goskal *et al.* [60].

- Dynamic Vehicle Routing Problem (DVRP)
  In DVRP, decisions are made before all relevant information becomes available. Parts of the information are reviewed dynamically during the design or execution of the routes. Therefore, decisions must then be modified as new information is received. Early review can be found in Psaraftis [108] which provided the status and prospect of the DVRPs. However, recent review by Pillac *et al.* [101] has revealed a number or technological advances which led to a renewed interest in DVRPs.

- Stochastic Vehicle Routing Problem
  In classical VRPs, it is assumed that all data are deterministic. However, in SVRP, information on customer demand or travel times between customers is given by probability distributions. Recent works can be found in Lei *et al.* [80], Taş *et al.* [126] and Ehmke *et al.* [43].

- Location-Routing Problems (LRP)
  This is a combination of routing and locational decisions. The problem involves simultaneously locating a number of facilities among candidate sites and establishing delivery routes to a set of users in such a way that the total system cost is minimized. Earliest survey on LRP can be found in Laporte [73]. For more information about recent works on LRP, see the review paper of Prodhon and Prins [107].

- Inventory Routing Problem (IRP)
  In IRPs, each customer has a given consumption rate of good, a given initial stock and a given storage capacity. A customer may be visited any number of times during a multi-period planning horizon. This is to ensure that no customer runs out of stock. Bertazzi and Speranza [13] have introduced the IRPs with example, classified their characteristics and showed different models and policies for the class of problems.

Many real life problems have been modelled as VRPs, from general to the specific problem-case studies. Thus, there are many methods in literature which can be found in Toth and Vigo [130], Mole and Jameson [94], Wark and Holt [137], Gillet and Miller [55], Fisher [50], Altinkemer and Gavish [3] and others.

VRPs which can be represented as graphs or networks are usually formulated as mixed-integer linear programming (MILPs). Concerning the solution algorithms, two principal approaches can be distinguished: mathematical programming, and heuristics and meta-heuristics. Mathematical programming is based on the MILP models of VRP or can be categorized as exact solution. Theoretically, exact solution guarantees to find an optimal solution if one exists.

The vehicle routing problem, which is hard combinatorial problem, has therefore attracted considerable research attention and a number of algorithms has been proposed for its solution. To the literature it is categorized as a combinatorial optimization problem due to the difficulties encountered in solving real world applications of such problems. The goal is to find the possible solution and by solving this, a discrete set of possible solutions will be generated. The VRP is categorized as NP-hard [87],[110], which means that the computational time required for solving the problem increases exponentially as the problem size increases [77]. For such problems, heuristics arriving at approximation solutions are obtained which are fast enough and sufficiently accurate for the purpose.

2.3 VRP with Due Dates (VRPDD)

A number of works in VRPTW has been reviewed, where each customer needs to be served within their predefined times. However, for some reasons, the service could not be done if a vehicle arrives before the customer's earliest time. Thus, a penalty will be given which represent the cost of keeping goods before the time.

Due to the fierce competition companies have to face in today's global market, they need to utilize their resources efficiently. For example, in MTO business trend, custom-made products are manufactured and delivered within very short lead times. Therefore, there exist a situation where customer's earliest time is no longer important. As soon as an order is available, it must be immediately supplied to the customer without needing to wait for certain time. This applies to the time-sensitive products that have very short life-cycle such as in perishable food industry, industrial refuse collection, newspapers and mailing.

Concerning to the above, there is a need some research nowadays which only deals with the customer's latest time to be served, i.e Due Date. A recent study on VRPDD was done by Archetti *et al.* [5] where the problem conceptually lies between PVRP and IRP.

Park [99] studied vehicle scheduling problem with due dates and time deadlines. Penalty is given when the vehicle reaches a customer beyond its due time. Nevertheless, there was usually a deadline for service, i.e. latest allowable service times. The paper focused on solving the problem in order to minimize the total weighted travel time, tardiness and the fleet size.

Another study on vehicle routing problem with due times was carried by Kang *et al.* [70] which presented a heuristic for the vehicle routing problem with due times. Tabu search was used to solve a mixed integer programming formulation where a penalty cost is imposed if the service exceeds customer's due time. The objective was to minimize the weighted sum of the travelling time and total tardiness.

Both of the latter studies were focusing on development of algorithm by assuming deliveries were done at the same time from a depot. Our study on VRPDD will assume varied departure times as they depend on the availability of the orders to be dispatched.

2.4 Heuristics for VRPs

Many heuristics used to solve VRP can be classified as classical and meta-heuristics. Metaheuristics such as Simulated Annealing (SA), Variable Neighborhood Search (VNS), Ant Colony System (ACS), Genetic Algoritm (GA) and Tabu Search (TS) have received great attention in literature since 1990. It can be seen that those heuristics are capable of solving vehicle routing problems with large number of customers.

2.4.1 Classical Heuristics

Classical heuristic methods have limited exploration of the search space but can still produce good quality solutions within modest computing times [2]. This class of heuristics contains the most standard constructions and improvement procedure that has been used until today in commercial packages. Some of them are used in combinations with other techniques. Laporte and Semet [130] divided classical heuristic into three main categories, which are constructive heuristics, two-phase heuristics and improvement heuristics.

2.4.1.1 Constructive Heuristics

A constructive heuristic build the feasible solutions gradually. At the same time, this heuristic keeps an eye on the solution cost. The only drawback in this heuristic is it does not contain an improvement phase *per se*. Two main constructive techniques used in VRP are savings and insertion cost algorithms.

Clarke and Wright Savings Algorithm is known as the best route building heuristic [49]. The algorithm was introduced by Clarke and Wright [33] to solve a vehicle routing problem, related to the scheduling of vehicles from a depot to a number of delivery points. The algorithm starts by serving every customer individually by a separate vehicle. When there are n customers to be served, that means n routes had to be created initially which gave the highest cost for routing. The cost could then be reduced by combining any two of these single customer routes resulting in using one less vehicle.

Sequential heuristic method is classified as 'tour-building heuristic'. Basically, there are two phases in VRP. The first phase is to assign each of the unrouted customer to its best feasible insertion position based on the minima additional distance and time required. The second phase is the selected customer inserted in the emerging route using a maximum saving concept. Two algorithms based on the sequential were given in Mole and Jameson [94] and Christofides, Mingozzi and Toth [32]. In [94], the authors used a sequential insertion heuristic which expands one route at a time whilst [32] . applied both the parallel and sequential insertions.

2.4.1.2 Two Phase Method

In this method, the problem is decomposed into its two natural components; i.e. clustering of vertices into feasible routes, and the actual route construction. There are two methods, by which this is done. They are cluster-first route-second, and route-first cluster-second.

Cluster-first route-second method performs a single clustering of the vertex set and then determines a vehicle route on each cluster. Two classic algorithms were Sweep Algorithm which was popularized by Gillett and Miller in 1974 [55]. In this algorithm, geographical coordinates for each customer are needed. It is assumed that all customer locations can be represented by polar coordinates. The depot itself might be used as the origin in the coordinate system. By rotating a ray centered at the depot with an arbitrarily starting angle, all the customer locations can be ordered according to their angles in polar coordinates. Then, customers are assigned to several clusters, and each cluster must not exceed the capacity of the vehicle. There are two types of sweep algorithm, a forward-sweep and a backward-sweep. In forward-sweep, customers are ordered beginning with the location that has the smallest angle and the other way around for the backward-sweep algorithm.

Route-First Cluster-Second applies to problems with unlimited number of vehicles. In the first phase, the algorithm starts with the construction of giant TSP tour, disregarding the side constraints. Next, decompose this tour into feasible routes in the second phase. Beasley [11] considered route-first cluster-second method for solving vehicle routing problem. The construction of giant tour can be formed in a number of different ways. Without loss in generality, customers are ordered as usual. Customer 1 is the first customer on the directed tour after the depot (denoted as 0), customer 2 is the second visited customer of the route and so on.

2.4.1.3 Improvement Heuristics

Improvement heuristics are normally applied to improve a feasible solution obtained by using other methods. The heuristic will search for the best improvement solution if possible. Then, the solution is updated and the search for a new improved solution is repeated until no more improvement is found or stopping condition is met. Improvement heuristics can be divided into two; i.e. single-route improvement and multi-route improvement. The terminologies of both single-, and multi-route improvement were found in Laporte and Semet [74] which gave the same meaning of intra- and inter-route improvement which will be used throughout this thesis. The intra-route improvement performs changes to one route at a time, such as permute the customers within a route, while the latter involves exchanging and moving customers between two or more routes. A more complete survey of neighborhoods in heuristics can be found in Bräysy and Gendreau [18].

In single-route improvement, the methods 2-Opt and 3-Opt are commonly used by researchers on routing problems. These routines were introduced by Lin [84]. Both methods work by removing two (2-Opt) or three (3-Opt) edges and adding other two or three new edges within a route. The best edge exchange that improves the current route cost is updated. The process is repeated until no more improvement is found.

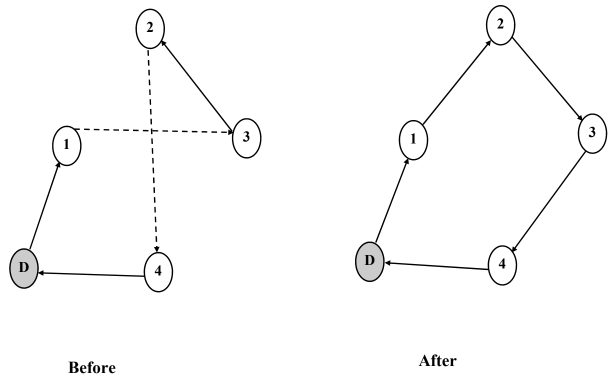


FIGURE 2.1: An example of 2-Opt.

Figure 2.1 shows an illustration of a 2-Opt move. Edges (1,3) and (2,4) are replaced by edges (1,2) and (3,4), thus reversing the direction of customers between 3 and 2. To improve the route cost, the move is accepted if the new cost is reduced compared to before.

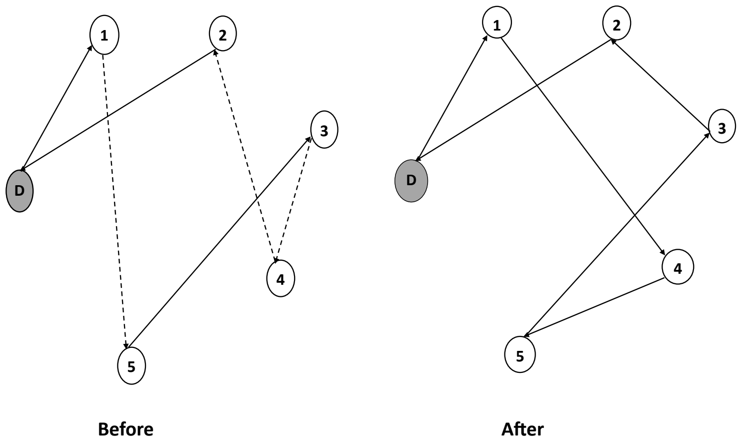


FIGURE 2.2: An example of 3-Opt.

Figure 2.2 shows an example of a 3-Opt move. Three edges; (1,5), (3,4) and (4,2) are removed and those customers are reconnected by three other new edges without changing the remaining directions.

In multi-route improvement, the method makes changes to more than a route. The insertion and λ-interchange are examples of routines that fall into this group. The insertion routine works by removing a node from one route and try to reinsert this node into other an route at the best position, while λ-interchange routine works by exchanging or swapping an equal number of nodes, λ, between two routes [96]. In both routines, the insertion or swapping is executed once an improvement is found. The search will start again until the stopping condition is met.

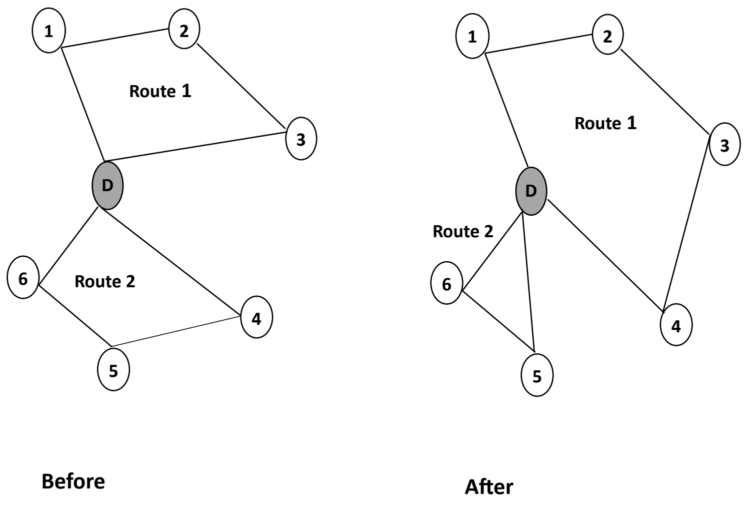


FIGURE 2.3: An example of insertion move.

Figure 2.3 shows an example of insertion move between Route 1 and Route 2. Customer 4 from Route 2 is inserted into Route 1 which improves the current solution cost.

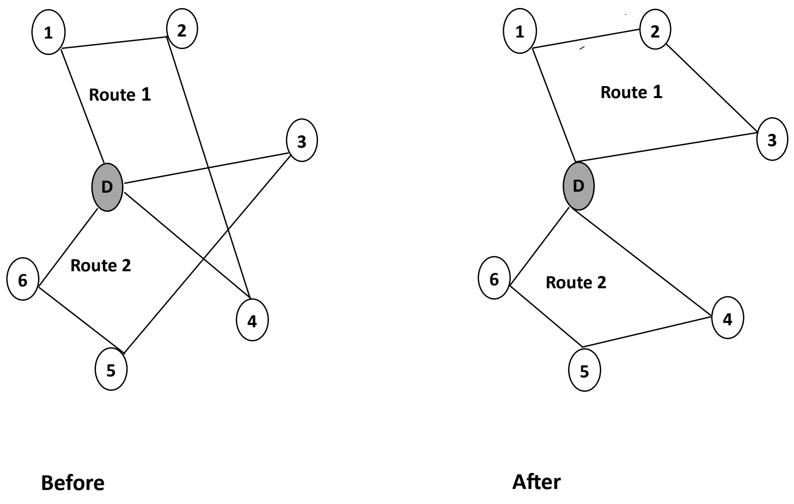


FIGURE 2.4: An example of swapping/exchange move.

Figure 2.4 shows an example of swapping move. Customer 3 in Route 2 is swapped with customer 4 in Route 1. Swapping of these customers is done if the move could improve the current solution cost.

Another algorithm which can be classified under improvement heuristics is variable neighborhood descent (VND). This algorithm was known as one of the successful algorithms that contain combinations of single-, and multi-routes improvement methods. In VND, single- or multi-routes move is considered as two different neighbourhoods. The idea of Hansen and Mladenović [93] which systematically changes the neighbourhood is used to avoid any local trap. VND algorithm is seen as a successful improvement heuristic.

VND is an enhanced local improvement strategy which is commonly used as a subordinate in other heuristics. VND is very popular due to its simplicity and its relatively short computational time to find the improvement solution. Basic VND iteratively explores neighborhoods $N_k, k = 1, ..., k_{max}$ and applies any search strategy to each of the neighborhoods. Although the solution found may have a risk to be trapped in the valleys, changing the neighborhood can result in better local optima. Thus, VND can be a very successful method, due to the fact that a local optimum within one neighborhood is not necessarily a local optimum of other neighborhood. By changing the neighborhoods, a solution can be found to escape from the valley.

The algorithm begins as follow. Starting with the first neighborhood N_1, local searches are performed until no further improvements are possible. From this local optimum, the search continues with the next neighborhood N_2. VND returns to the N_1 again if and only if an improvement solution is found in the neighborhood. Otherwise, the method proceeds with N_3, and so forth. If the last neighborhood $N_{k_{max}}$ has been applied and does not return any further improvement, the current solution represents a local optimum with respect to all used neighborhoods and VND terminates.

2.4.2 Local Search

To find a good quality solution, a heuristic or metaheuristic algorithm moves a solution to another through a searching process that is often called Local Search (LS). According to Derigs and Vogel [39], local search neighborhoods can be evaluated rather efficiently for the standard VRP since changes in distance, vehicle load and duration can generally be calculated in constant time. There are two search strategies that can be used in any method described previously. Each search strategy uses a rule or policy which indicates the order in which new solutions are searched. Two of the simplest strategies mentioned earlier are the best improvement and the first improvement strategy. The best improvement strategy examines all neighbours satisfying the criterion needed and selects the best among them. The other search strategy selects the first improved solution found in the neighborhood which is better than the current solution

[18],[65],[111].

LS heuristics are not constructive heuristics. Thus, they do not develop a solution, but instead need an input solution and then iteratively modify it to improve its quality. The strength of LS is the intensification of the search [39]. Local search heuristics modify an input solutions, s, by performing a sequence of operations on the current solution to produce a new improved solution. Those operations are referred to as moves. The moves used to transform a solution s into a set of other solutions, $N(s)$, also known as the neighborhood of s. If the move made leads to an improved solution, then the current solution is replaced by the improved solution and the process is repeated. Otherwise, the searching process stop.

2.4.3 Metaheuristics

Since 1990, metaheuristics have received a great attentions among VRP researchers. The methods have been applied to many VRP problems. An intensive survey was conducted by Bräysy and Gendreau [19] and presented in [131]. This solution procedure explores the solution space to identify near optimal solution which is much better than classical heuristic. Some of the standard route construction and improvement heuristics which have been described in previous section are embedded in the procedure.

In metaheuristics, a deep exploration in the most promising regions of the solution space is performed. In this section, several selected metaheuristics are examined. The methods which have been applied to our VRP works are Variable Neighborhood Search, Large Neighborhood Search and Tabu Search.

These methods were chosen since they had received a great deal of attention in solving VRP problems. The difference on the exploration techniques in finding solution becomes an advantage to each method. In VNS, there is no trajectory that needs to be followed. Once an improvement has been found, the solution jumps into the new one. A similar technique is used in LNS but can be in more complex neighborhood. LNS could make it possible to find a better candidate solution in each iteration and hence traverse a more promising path. However, in TS, solution space is explored by moving a solution x_t identified at iteration t to the best solution x_{t+1} at iteration t in a subset of the neighborhood of $N(x_t)$ of x_t. To prevent cycling over a sequence solutions, tabu mechanism is used which makes it unique compared to the two above.

2.4.3.1 Variable Neighborhood Search (VNS)

Variable Neighbourhood Search (VNS) is a recent metaheuristic which exploits systematically the idea of neighbourhood changes [93], both in the decent to local minima and in the escape from the valleys which contain them. It was an algorithm that combines local search strategy and the variable neighbourhood structure [66]. The idea of changing neighborhoods is based on the fact that different neighborhoods may have different local minima, is the key to the success of the VNS. According to Stenger *et al.* [123], a VNS which performs a local search on increasingly larger neighborhoods leads to an effective exploration of the solution space and helps to avoid getting stuck in local optima. Even though VNS algorithm can effectively solve VRPTW problems that have relatively wide time windows, its climbing ability is relatively poor. Hong [66] stated that combining LNS and VNS can implement the complementary advantage between them.

Contrary to other metaheuristics based on local search method, VNS does not follow a trajectory [62]. The search starts by exploring the search space of the incumbent solution x to generate a new solution. A local search is then applied to optimise it before the replacement decision is undertaken. The procedure of VNS is given below:

- Select the set of neighbourhood structures N_k, for $k = 1, ..., k_{max}$, that will be used in the search; find an initial solution x; and choose a stopping condition;
- Repeat the following sequence until the stopping condition is met:
    1. Set $k \leftarrow 1$
    2. Repeat the following steps until $k = k_{max}$:
        (a) *Shaking.* Generate a point $x^{'}$ at random from the kth neighbourhood of $x(x^{'} \in N_k(x))$;
        (b) *Local search.* Apply local search method with $x^{'}$ as initial solution; denote the obtained local optimum as $x^{''}$;
        (c) *Move or not.* If the local optimum $x^{''}$ is better than the incumbent x, move there ($x \leftarrow x^{''}$), and continue the search with $N_1(k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$;

In VNS algorithm, the neighborhood size k increases from 1 to a certain maximum value k_{max}. The most important stage is the shaking phase where the current solution x is shook within the current neighborhood $N_k(x)$ to generate a new solution, x'. It is a crucial issue to define the shaking neighborhoods that will allow enough perturbations to the current incumbent solution in obtaining a near optimal solution. A local search is applied at solution x' to intensify the search to obtain a local optimum, x''.

Next, the new solution x'' is compared with the current best solution, x. If x'' is better than x, the new solution is accepted and the search continues in the same neighbourhood in the next iteration. Otherwise, other shaking neighbourhoods will be used and the process is repeated. The procedure continues until all shaking neighborhoods have been considered and the search determines when the stopping condition is met.

According to [62], VNS procedure could be of varied nature. For example, the basic VNS procedure is a descent algorithm with the first improvement criterion which can be changed easily to the best improvement criterion. This could be done by selecting the best improvement among all k neighborhoods. Other variations include the starting neighborhood sizes which does not necessarily have to begin at $k = 1$.

The basic VND algorithm is simple and easy to understand. Apart from that, any well-known neighborhood moves and local search methods can be easily integrated in several ways and applied within the VNS framework. These factors have encouraged many researchers to use it for any combinatorial optimization problem such as TSP, VRP or facility location problem with almost no modification to the basic structure of the algorithm.

2.4.3.2 Large Neighborhood Search (LNS)

Bent and Hentenryck[12], Goel and Gruhn [57],[58],[59], Ropke and Pisinger [113],[112], Prescott-Gagnon *et al.*[106], and Lee *et al.*[76] have been successful in solving the vehicle routing problem using LNS.

LNS was first introduced by Shaw [119]. In LNS, an initial solution is gradually improved by alternately destroying and repairing the solution. It uses a process of continual relaxation and re-optimisation [76],[119]. In VRP, the current positions of some customers are relaxed, which mean these customers are removed from the current routing plan. Next, this routing plan is re-optimised over the relaxed positions by re-inserting the relaxed customers into the routing plan with the least insertion cost. Removal and re-insertion of customers in an iteration is considered as an examination of a neighborhood.

LNS improves the current solutions by using this searching process in large neighborhoods, which may contain potentially better solutions. This would increase the chance of finding high-quality solutions. As mentioned above, the algorithm works by removing a large number of customers from their current routes and re-schedule them into other routes at the least insertion cost. The process of removing and reinserting the removed customers into routes are also known as destroying and repairing procedure by Pisinger and Ropke [104]. It is also similar to the heuristic proposed by Schrimpf *et al.* [117] who calls this processes as ruin and create.

LNS starts with an initial solution x. Since LNS algorithm is very sensitive to the initial solutions, Hong [66] opined that it is very important for LNS to have a high quality initial solution to reduce computational time. The author adopted the Clarke-Wright algorithm to produce the initial solutions which can generate some feasible solutions in a short time. After generating an initial solution, k customers are removed from their current solution x in each iteration. The removed customers could be selected at random, or based on some relatedness measures which was introduced by Shaw [119]. Then, a new solution x' is generated by inserting those customers into the routes. The new solution x' is accepted as the next current solution if it shows an improvement from the previous best solution. The algorithm continues until a stopping condition is met.

The procedure below shows a general LNS algorithm.

- Find an initial solution x;
- Let $x^{best} \leftarrow x$
- Repeat the following sequence until the stopping condition is met:
    1. Remove k customers from x
    2. Re-Insert those k customers into routes to generate solution x'
    3. If x' better than x, accept the solution $x \leftarrow x'$
    4. If x' better than x^{best}, accept the solution $x^{best} \leftarrow x'$
- Return x^{best}.

The crucial components of the heuristic are the removal and the re-insertion processes. This is because the performance and robustness of the overall heuristic is highly dependent on the choice of the two procedures. Thus, the probability to find a better solution is strongly connected to the choice of neighborhood $N_k(x)$.

The most important choice when implementing the removal process is the degree of removal customers. If only a small part of the solution is destroyed i.e. a small number of customers removed, the heuristic may have difficulty exploring the search space as the effect of large neighborhood is lost. On the other hand, if too large a part of the solution is destroyed, LNS is seen as a repeated re-optimization process [104] which might be excessively time consuming, or yield a poor quality solution.

Ropke and Pisinger [112] stated three removal heuristics:

- Shaw's removal: The general idea is to remove requests that are somewhat similar, using a relatedness measures and thereby create new, perhaps better solutions.For example, remove the related customers which are geographically close to one another.
- Random removal: The algorithm simply removes the customers randomly.
- Worst removal: Remove the customers whose removal decreases the cost function the most.

and two insertion heuristics:

- Basic greedy heuristic: This is a simple heuristic which perform at most n iterations and the insertion is done one at one time.
- Regret heuristic: The heuristic tries to improve upon the basic greedy heuristic by incorporating a kind of look-ahead information when selecting the request to insert.

Lei *et al.* [80] developed four removal heuristics and four insertion heuristics. Since two removal heuristics (ie. expected worst removal and random removal) and one insertion heuristic (ie. greedy insertion) discussed by the authors are quite similar to the removal and insertion heuristics stated by Ropke and Pisinger [112], so we will discuss the other removal procedures. These removal procedures are as follows:

- Similarity removal heuristic: It is an extension of the Shaw's removal heuristic where the vertices that are similar to the first vertex which is randomly selected, in terms of proximity and time window, are removed based on a similarity measure.
- Feasibility-oriented removal (FOR): The heuristic concentrates on the m-infeasible solutions and moves toward feasibility by removing vertices to make the number of routes close to m.

The three insertion heuristics include:

- Feasibility-oriented insertion (FOI): The heuristic concentrates on the m-infeasible solutions and moves toward feasibility by inserting vertices to make the number of routes close to m.
- Demand and failure sorting insertion: The heuristic inserts the vertex with the largest expected demand from the removed vertices after sorting the removed vertices in decreasing order of their expected demand and the routes in increasing order of their probabilities of failure.
- Time and failure sorting insertion: Insert the vertex with smallest time window width after sorting the removed vertices in increasing order of the time window's width.

The strategy of choosing nodes to be removed makes LNS algorithm easily successful at finds local minimum points, but hard to search for the global optimal point. To overcome this shortcoming, Hong [66] adopted a variety of removing sets and the maximum-best-first inserting algorithm. The removed node-sets are the relatedness node-set, the violation time windows node-set, the worst node-set and the fewest-longest route node-set. We will not discuss the algorithm for removing relatedness node-set and worst node-set as they are similar to the Shaw's removal and worst removal respectively.

The removed node-sets are:

- Violation time window node-set: Resolved via the solving strategy (see [66] for detail) if violation of time windows occurs due to the insufficient quantity of vehicles in use. Otherwise, eliminate the node-set by the removing and the re-inserting approach if the violation of time window is due to incorrect service sequence of nodes.
- Fewest-longest route node-set: the node-set of a route will be selected if the route has the fewest nodes and the longest travel distance of all routes.

Whereas the max-best-fit inserting algorithm is described as follows:

- Maximum-best-first insertion algorithm: The removed node-set is randomly inserted into the corresponding best or second-best place according to the deviation degree that is defined to measure the extent to which a solution deviates from its best solution.

A basic procedures of "remove and insert" as discussed in Ropke and Pisinger [112] will be adopted in our work later.

2.4.3.3 Tabu Search (TS)

Tabu Search (TS) is a memory-based local search metaheuristic. It was proposed by Fred Glover in 1986 [56], which later becomes a popular method in finding solution to large combinatorial problems. The idea behind Tabu Search is to improve local search by accepting non improving moves within the neighbourhood. It was extended from the 'hill climbing' method where hill climbing is allowed to overcome local optima. In some cases, Tabu Search is reported to provide the closest solutions to optimality. Tabu Search is claimed to be among the most effective methods and the best to tackle difficult combinatorial problems at hand [90],[35].

Two basic elements which are important in tabu search are the definition of its search space and its neighbourhood structure. The search space contains all the possible solutions that can be considered during the search, while the neighbourhood structure is a set of neighbouring solutions in the search space. The method performs an exploration of the solution space by moving the current solution into other solution in the next iteration. The method or algorithm records a list of search known as *tabu list*. The list is used to record the move that has been recently examined. Since the algorithm accepts the non-improving solution, this tabu list mechanism is put in place to prevent the process from cycling over a sequence of solutions.

To avoid the cycling problem, bookkeeping is typically required. Thus, tabu search is also known as a memory-based search strategy. The memory can be classified as recency-based memory and frequency-based memory [125]. If we have a tabu list of size N that records the last N moves, this is known as recency-based memory, also called short-term memory. The other type of memory, which is frequency-based memory, provides some additional information of how many times the tabu moves have been attempted.

In tabu search, it is important to decide an appropriate size of tabu list. Different tabu list sizes would affect the quality of the solutions. If the tabu size is too small, the possibility of cycling solution might occur. On the other hand, if the size is too big then it is difficult to find a good solution in a narrow space. Thus, the proper size of tabu list is often determined empirically [78][132].

In order to improve the search, diversification and intensification processes are often implemented in the search exploration. Diversification is used to ensure that the search process is not limited to a certain solution space while intensification consists of performing an accentuated search around the best solutions [54]. By comparing with other meta-heuristic algorithms, Yu *et al.* [140] showed that Tabu search can explore new possible solution space and prevent the proposed algorithm from being trapped in local optimal. The algorithm stops when there is no improvement found after a certain number of iterations or when the maximum number of iterations is achieved or the allocated computing time is used.

The idea behind the general tabu search algorithm is as follows.

- Find an initial solution x;
- Let $x^{best} \leftarrow x$
- Repeat the following sequence until the stopping condition is met:
    1. Find a subset of $N(x)$, the neighbor of x which is not in the tabu list.
    2. Find the best one, x^* in $N(x)$
    3. If x^* better than x^{best}, accept the solution $x^{best} \leftarrow x^*$.
    4. Update the tabu list.
- Return x^{best}.

As mentioned, the method explores the search space of feasible solutions by making a series of moves. It moves the solution x_t at iteration t to the best solution, x_{t+1} found at iteration $t+1$ in a subset of the neighborhood $N(x_t)$ of x_t. Since the solution x_{t+1} is not necessarily better than x_t, the search need to be guided to avoid getting trapped in local optima. Therefore, a *tabu* mechanism is implemented to prevent the solution from cycling over a sequence of solutions. In other words, the solutions that were recently examined are forbidden or declared *tabu* for a certain number of iterations. However the tabu status of an attribute can be revoked if the cost of the solution is below a so-called aspiration level or criterion [39], [98]. As mentioned by Derigs and Vogel [39], the tabu search idea leaves many degrees of freedom for the actual implementation. For example the basic concept can be relaxed and only partial neighborhoods are scanned.

2.5 Summary

The chapter reviews related works on VRP with appropriate references for further reading. Included here is the VRP cases which only involve due dates which are seen to more competent in MTO business trend. This has been followed by a discussion of some heuristics that can be used to solve our VRP problem. In order to improve the solution found, a local search technique, VND, will be embedded in each of our approaches.

Chapter 3

Vehicle Routing Problem with Release and Due Dates (VRPRDD)

3.1 Introduction

Studies about VRP have led to one interesting practical application that is the subject of this thesis. This chapter provides vehicle routing problem which includes the customer's order release date and due date. It starts by describing the scenario intensively, followed by the formulation of the problem. Then, data sets and problem design that been used here are presented. Before going to the Chapter 4, a small example is shown in the last two sections validating the formulation given.

3.2 Problem Background

Transportation is an important domain of human activity which supports and makes possible most other social and economic activities. All human activities including the trade of wealth and productive labor involve transportation, whether it is the movement of goods, or the movement of people from their homes to their jobs, or their homes to the places where they shop. It seems that trade without transportation is not possible. Freight transportation, in particular, is one of today's most important activities, not only measured by the yardstick of its own share of a nation's gross national product (GNP), but also by the increasing influence that the transportation and distribution of goods have on the performance of virtually all other economic sectors [17].

VRP is an optimization problem where a set of geographically scattered customers has to be served by a fleet of vehicles to minimize the routing costs without violating the vehicle capacity constraints. Ideally, customers are to be served within a certain time interval which is crucial for the transporter. Any disruption will cause delays. In practice, the routing plans are sometimes disrupted by many factors such as vehicle breakdown, traffic congestion, delay from any service point before and many more. Delays mean losing reliability and trustworthiness and often penalties need to be paid, causing decreasing profits. Hence, the due dates associated with each customer should be considered. To avoid any penalty due to delay in the delivery process, it is assumed that customer's orders are loaded into a vehicle once it is available at the depot. The times where orders are available for the delivery process are known as release dates of customers' order.

This study is motivated by the Make To Order-business trend. This production-distribution system consists of one manufacturer and one or more customers. The manufacturer makes the products once orders are received. In this way, inventory, one of the major costs within the supply management flow, could be reduced or avoided altogether. Lesser inventory needs a closer linkage between production and distribution processes. This makes joint plannings necessary.

It needs to be mentioned here that our study does not consider the earliness even though this has been considered in the most papers of VRPTW. This is because we would like to highlight the necessity of an integrated planning of production and distribution for time-sensitive products that have a very short life-cycle such as perishable foods. This kind of products cannot be stored and must be delivered to the customers immediately after production.

In order to see the better efficiencies of production-distribution operations in integrated manner, this study will be divided into two; vehicle routing problem with release and due dates (VRPRDD) and production scheduling-VRPRDD (PS-VRPRDD). In VRPRDD, we focus on the vehicle routing only by assuming that the release dates are provided before the distribution process begin. Whilst in PS-VRPRDD, the production sequence is highlighted.

3.3 Description of VRPRDD

In this section we describe the model of the VRPRDD studied in this thesis. The system includes one manufacturer or depot and N customers denoted as $v_1, v_2, ..., v_N$ located at different locations in a given system. At the beginning of the process, the manufacturer receives N orders or jobs, requesting for processing in which each customer is associated with a job. Each job i has a known size: q_i, where $i = 1, 2, ..., N$. q_1 is the order's size of customer 1, q_2 is the order's size of customer 2 and so forth. Each order, i is associated with release date, r_i and due date d_i. Release date of order i is referred as the availability of order i to be in the delivery process. In order to avoid lateness in the delivery process, order i need to be delivered before its specified due date d_i which is determined by customer i.

It is important to emphasize here that all orders are being processed on a single production line of the manufacturer. It is assumed that one job is processed at a time and one is processed after another. No preemption is allowed. Hence, the sequence of jobs production is important to determine the release date of customer's order, r_i. However, in our VRPRDD, it is assumed that r_i is given. Further discussion on jobs processing and sequencing will be given in the chapter PS-VRPRDD later.

3.4 Problem Formulation

Before providing the mathematical formulation, we present here some notations and definitions used throughout the chapter.

Our vehicle routing problem can be written in a graph theoretical perspective. Let $G = (V, A)$ be a complete undirected graph with vertex set $V = \{v_0, v_1, ..., v_n\}$ and edge set $A = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$. Each vertex $v_i \in V$ is associated with:

i) a known order's size to be delivered, q_i ($q_0 = 0$ for vertex v_0).

ii) a release date of each customer's order, r_i.

iii) a uniform service time for servicing at customer's location, s_i.

iv) a due date for each customer, d_i, where d_i is the latest time for the i^{th} customer to be served. Any servicing beyond the date will be considered as lateness in delivery.

Mathematical Notations:

n number of customers

$N = \{1, 2, ..., n\}$ the set of n customers

$\overline{N} = \{0, 1, 2, ..., n\}$ the set of all nodes including the depot

h number of vehicles

$K = \{1, 2, ..., h\}$ the fleet of vehicles

Decision variables

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

y_j tardiness for customer j

z_{ik} arrival time of vehicle k at customer i

z_{0k} departure time of vehicle k from depot

Parameters

Q	capacity of vehicle
q_i	order size of customer i where $q_0 = 0$
s_i	service time of customer i where $s_0 = 0$
d_i	due date of customer i
r_i	release date of customer i's order
t_{ij}	travel time from customer i to customer j
c_{ij}	travel cost from customer i to customer j
w_i	tardiness penalty cost of customer i
M	large positive integer number

Note: We assume the travel time is proportional to distance, meaning that the vehicles run at constant average velocity.

Formulation

$$\min \ (\alpha)\left(\sum_{i\in\overline{N}}\sum_{j\in\overline{N}}\sum_{k\in K} c_{ij}x_{ijk}\right) + (1-\alpha)\left(\sum_{j\in N} w_j y_j\right) \tag{3.1}$$

subject to

$$\sum_{k\in K}\sum_{j\in\overline{N}} x_{ijk} = 1 \quad \forall i \in N \tag{3.2}$$

$$\sum_{j\in N} x_{0jk} \leq 1 \quad \forall k \in K \tag{3.3}$$

$$\sum_{j\in N} x_{0jk} = \sum_{i\in N} x_{i0k} \quad \forall k \in K \tag{3.4}$$

$$\sum_{i\in\overline{N}} x_{ipk} = \sum_{j\in\overline{N}} x_{pjk} \quad \forall p \in N, \forall k \in K \tag{3.5}$$

$$\sum_{i\in N}\sum_{j\in\overline{N}} q_i x_{ijk} \leq Q \quad \forall k \in K \tag{3.6}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ijk} \leq |S| - 1 \quad S \subset N, |S| \geq 2, \forall k \in K \tag{3.7}$$

$$z_{0k} \geq r_i \sum_{j\in\overline{N}} x_{ijk} \quad \forall i \in N, \forall k \in K \tag{3.8}$$

$$z_{ik} + s_i + t_{ij} + M\left(x_{ijk} - 1\right) \leq z_{jk} \quad \forall i \in \overline{N}, \forall j \in N, \forall k \in K \tag{3.9}$$

$$z_{jk} + s_j - d_j + M\left(\sum_{i\in\overline{N}} x_{ijk} - 1\right) \leq y_j \quad \forall j \in N, \forall k \in K \tag{3.10}$$

$$x_{ijk} = 0 \quad \forall i = j \in \overline{N}, \forall k \in K \tag{3.11}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i, j \in \overline{N}, \forall k \in K \tag{3.12}$$

$$y_j \geq 0 \quad \forall j \in N \tag{3.13}$$

$$z_{ik} \geq 0 \quad \forall i \in \overline{N}, \forall k \in K \tag{3.14}$$

Equation (3.1) is the objective function that is to minimize the weighted total costs of travelling and tardiness. In order to achieve this, we define a weighting factor α ($0 \leq \alpha \leq 1$). Weight α is based on the manufacturer's relative preference on these two costs; travelling and tardiness. It can be seen that when α is close to 0, more emphasis is given to the total tardiness cost. On the other hand, when α is close to 1, more emphasis is given to minimize the travelling cost. In situations where the relative preference on these two costs measurement is difficult to decide, we can simply solve the problem multiple times with varying values of α and pick one with the least cost.

The constraint (3.2) means that each customer is only visited once and only by one vehicle. Each vehicle used for routing originates from the depot as shown in constraint (3.3). Constraint (3.4) states that each vehicle originates from the depot, and goes back to the depot. Route continuity is represented by equation (3.5), i.e. if a vehicle enters a customer node, it must exit from that node. Capacity constraint for each vehicle is shown in equation (3.6) where the summation of order sizes of a route must be less than or equal to vehicle capacity limit. The set of constraints in (3.7) have the goal of avoiding subtour.

The constraint (3.8) shows that the departure time of vehicle from depot depends on the maximum release date of orders scheduled on the vehicle. Next, arrival time at each customer location is computed as in equation (3.9). Finally, the tardiness of each customer is computed in equation (3.10). Tardiness is defined as difference between arrival time of vehicle at customer's location and their due date. Decision variables are shown in (3.11)-(3.14).

For this model, there are $\{[(N+1)(N+1)(K)-(N+1)(K)]+N+K+(NK)\}$ number of variables and $\{N+K+K+NK+K+K+NK+(N+1)(NK)+NK\}$ constraints.

3.5 Data Sets and Problem Designed

3.5.1 Solomon's Instances

In this study, 56 VRPTW 100-customer instances of Solomon benchmark problem have been used. As no benchmark instances can be used for VRPRDD, Solomon's instances were chosen as they contain similar information that we need for the problem. These instances have six classes; C1, C2, R1, R2, RC1, and RC2. Each class contains between 8-12 individual instances. The C1 and C2 classes have customers located in clusters, R1 and R2 classes contain customers at random positions while RC1 and RC2 classes contain mix of both, clustered and random positions. All problems in each class have the same customer locations and the same customer demands, only the time windows differ. C1, R1 and RC1 basically have quite tight time windows, while C2, R2 and RC2 have wider time windows.

Each data instance contains a hundred customers, a central depot, customers' nodes, customers' demand, and earliest and latest customer's time windows. All distances between nodes are represented by the Euclidean distance. It is also assumed that the speed of all vehicles is unity which means it takes one unit of time to travel one unit of distance. Thus, numerically, travel cost c_{ij}, the travel time t_{ij} and the Euclidean distance between the customer nodes are equal to each other [124].

3.5.2 Generating An Initial Customer's Release Dates, r_i

To suit our problem, we initially set the earliest customer's time window as our customer order's release date, while the latest time window is assumed as our customer's due date. We then did a little modification to make it realistic in which there should have enough time between release date and due date. In other words, we need to have a realistic release date of customer's order. In order to do so, we follow the steps below:

1. Sort the customer's list in ascending order according to their due date, d_i. The purpose of doing this is to ensure that the order of customer with the smallest due date will be processed first. Thus, the list will be referred as an initial sequence of order's processing in a single production line.

2. According to the sorted list above, calculate the travel time, t_{0i} of each customer i from the depot. Depot is numbered as customer 0. The calculation is done on single trip from depot to a customer's location.

3. Next, calculate the difference between customer's due date, d_i and the travel time, t_{0i} to identify the largest slack, S.

$$S = max_{i \in V}\{d_i - t_{0i}\}$$

4. Next, we generate the processing time of order of each customer i, p_i randomly.

   $$p_i = Unif[1, 20], \forall i \in N$$

   One unit time of processing is chosen as in conventional setting in production scheduling whilst twenty unit time is used as a maximum time of processing the order. The number was chosen based on the method of try and error. This number cannot be too small for the vehicle routing to stay significant or too big to ensure the routing aspect will still be within control.

   It is important to note here that for the C classes, the summation of processing time of all customers' order should be less than the maximum slack, i.e. S. This can be written as

   $$\sum_{i \in N} p_i \leq S$$

   The condition above is necessary to ascertain that all orders have been released before reaching the latest due date. However, the condition cannot be applied to the R and RC classes as they have around seventy percent smaller due date durations compared to C classes.

5. Then, an initial release dates of customers' orders are identified based on the sequence in the production line and their processing times. The calculation used is as below:

   $$r_{i+1} = r_i + p_{i+1}$$

   where $p_0 = 0, r_0 = 0$. For simplicity, we denote i as sequence of order's processing.

   Note that, step 4 and step 5 are run simultaneously.

6. Next, we need to identify the maximum percentage of compression factor, F on processing times. We compressed the time where the rule $r_i + t_{0i} \leq d_i$ is broken, i.e. $d_i - t_{0i} > 0$. The compression factor is defined as:

   $$F = max_{i \in N}\{\frac{r_i - (d_i - t_{0i})}{r_i}\} \times 100\%$$

7. Once the maximum percentage has been identified, uniformly compress the processing times of all customers' orders by F. The calculation is as follow:

   $$\text{new } p_i = \text{old } p_i - (\text{old } p_i * F), \forall i \in N$$

   Using these new compression of processing time, re-calculate the new release date of each customer's order which will formally be used in our problem.

From the observation for the whole instances, we decided to expand the original latest time windows by 100 unit time for each customer. We are also changing the original service time given. The service time for C1 and C2 instances is 20 and the rest, i.e. R1, R2, RC1 and RC2 is 5. We were generating the processing time between 1 and 20 randomly. All calculations were done by using Microsoft Excel 2010.

3.5.3 Fleet-Size Cases

We introduce three different problem types which are based on three different values of α, in the objective function (3.1). We set the values of α as 0.9, 0.1 and 0.5 which represents minimizing the travelling cost, tardiness cost and the total cost respectively.

Apart from that, we also look at cases which have a limited number of vehicles. We are not looking at cases of not having enough vehicles to serve all customers. Rather, we consider the fleet size needed to minimize our objective functions. Because different orders may have different due dates, delivering more orders on time might require a large number of vehicles. Even though this will lead to a higher distribution cost (i.e. travelling cost), it may reduce the total tardiness cost. This will give some insight in finding the best joint schedule of production sequence and vehicle routing. A further investigation of fleet size will be carried out in the next chapter. Initially, an observation on the trade-off between travelling and tardiness costs will be made first.

In real business situations, some companies may want to use all of their available resources. This can be a strategy in maximizing the usage of resources so as to improve the overall costs. We hope to assess the impact of using a limited number of vehicles on our VRPRDD problem.

To investigate how much the company could benefit by maximizing the usage of their resources, different fleet-size problems have been designed. We refer to these cases as big, medium and small fleets.

The formula for determining the required number of vehicles, m, is as follows [49],[77],[83]:

$$m = \left\lceil \frac{\sum_{i \in N} q_i}{Q} \right\rceil \tag{3.15}$$

In order to design different fleet-size cases, we use the formula given in Wang *et al.*[136]. For each type of the fleet-size, we calculate m based on the percentage of utilization of vehicle capacity. Thus, we use the formula:

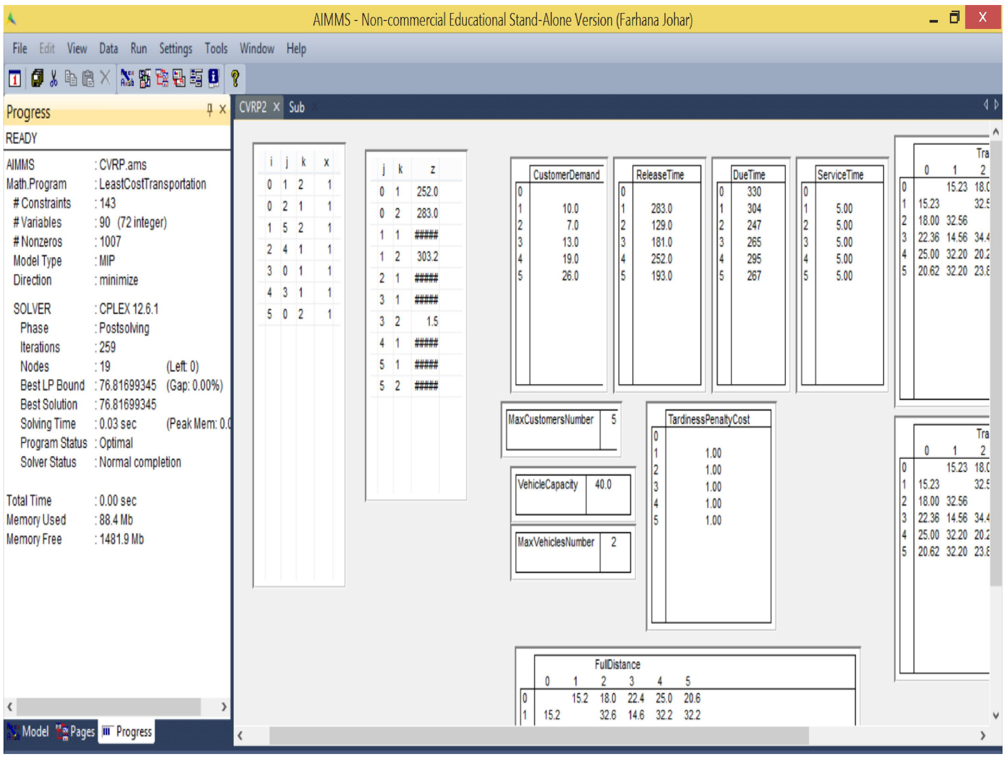
$$m = \left\lceil \frac{\sum_{i=1}^{n} q_i}{aQ} \right\rceil + 1 \tag{3.16}$$

where a is a parameter; $0 < a < 1$. The bigger the value of a is, the more constrained is the problem. In other words, the value of a is considered as how much of the capacity of each vehicle is being utilized. If a is close to 0, that's mean the vehicle is not fully loaded. Therefore more vehicles will be needed to deliver all customers' orders. If a is close to 1, more orders can be loaded into a vehicle. In that case a small fleet-size is needed for the deliveries.

We use three values for the parameter a corresponding to three fleet-size cases: $a = 0.3, 0.5$ and 0.7 which correspond to big, medium and small fleet-size cases respectively. Thus, our vehicle will consider all these three fleet-size cases as well as these three values for the weighting factor; 0.1, 0.5 and 0.9.

3.6 Model Validation

In order to check the validity of the problem formulation, we test on a small example which will be solved to optimality by the AIMMS 4.6.3.270-x64 software with CPLEX 12.6.1 solver. We have tried routing vehicle problems with five and seven customers. The results are shown as Figure 3.1 and 3.2 below.



```
READY
AIMMS            : CVRP.ams
Math.Program     : LeastCostTransportation
  # Constraints  : 143
  # Variables    : 90  (72 integer)
  # Nonzeros     : 1007
  Model Type     : MIP
  Direction      : minimize

  SOLVER         : CPLEX 12.6.1
    Phase        : Postsolving
    Iterations   : 259
    Nodes        : 19            (Left: 0)
    Best LP Bound   : 76.81699345   (Gap: 0.00%)
    Best Solution   : 76.81699345
    Solving Time    : 0.03 sec      (Peak Mem: 0.0
    Program Status  : Optimal
    Solver Status   : Normal completion

Total Time       : 0.00 sec
Memory Used      : 88.4 Mb
Memory Free      : 1481.9 Mb
```

i	j	k	x
0	1	2	1
0	2	1	1
1	5	2	1
2	4	1	1
3	0	1	1
4	3	1	1
5	0	2	1

j	k	z
0	1	252.0
0	2	283.0
1	1	#####
1	2	303.2
2	1	#####
3	1	#####
3	2	1.5
4	1	#####
5	1	#####
5	2	#####

	CustomerDemand	ReleaseTime	DueTime	ServiceTime	TardinessPenaltyCost
0			330		
1	10.0	283.0	304	5.00	1.00
2	7.0	129.0	247	5.00	1.00
3	13.0	181.0	265	5.00	1.00
4	19.0	252.0	295	5.00	1.00
5	26.0	193.0	267	5.00	1.00

MaxCustomersNumber	5
VehicleCapacity	40.0
MaxVehiclesNumber	2

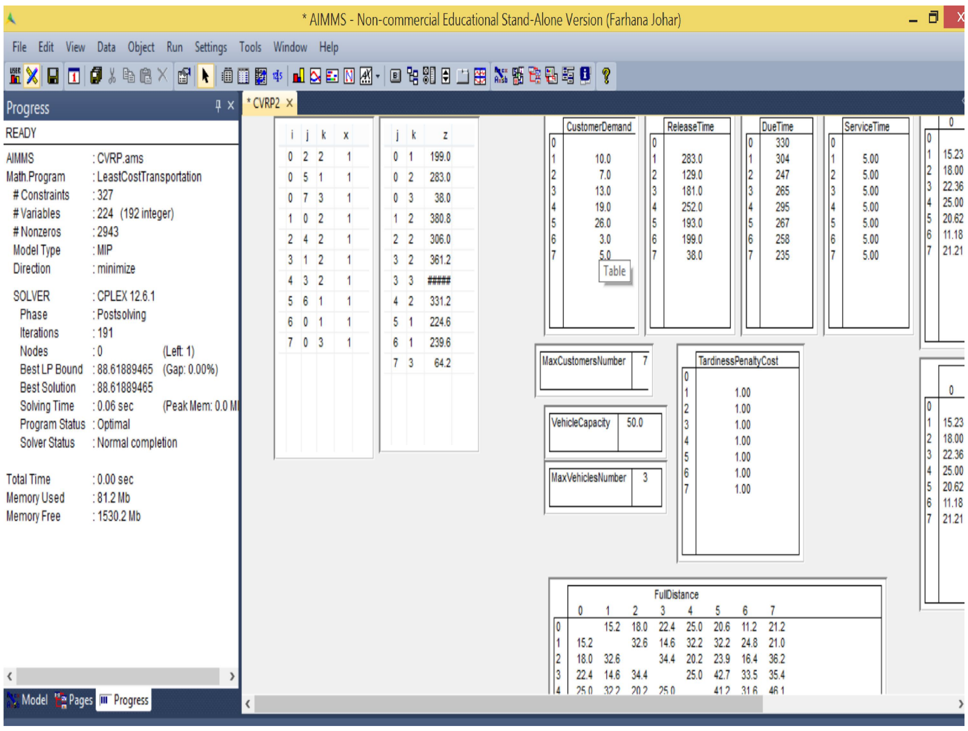
	Tra		
	0	1	2
0		15.23	18.0
1	15.23		32.5
2	18.00	32.56	
3	22.36	14.56	34.4
4	25.00	32.20	20.2
5	20.62	32.20	23.8

	Tra		
	0	1	2
0		15.23	18.0
1	15.23		32.5
2	18.00	32.56	
3	22.36	14.56	34.4
4	25.00	32.20	20.2
5	20.62	32.20	23.8

	FullDistance					
	0	1	2	3	4	5
0		15.2	18.0	22.4	25.0	20.6
1	15.2		32.6	14.6	32.2	32.2

FIGURE 3.1: Results for 5 customers.

Figure 3.1 shows the result of routing with five customers. Looking at generated result, the routes formed are: 0-2-4-3-0 and 0-1-5-0 which producing 76.817 unit cost of travelling distance and tardiness. The result is based on setting $\alpha = 0.5$ and vehicle capacity of 40. The optimal solution to the model was found in 0.03 seconds involving 143 constraints and 90 variables.



```
READY

AIMMS            : CVRP.ams
Math.Program     : LeastCostTransportation
  # Constraints  : 327
  # Variables    : 224   (192 integer)
  # Nonzeros     : 2943
  Model Type     : MIP
  Direction      : minimize

  SOLVER         : CPLEX 12.6.1
    Phase        : Postsolving
    Iterations   : 191
    Nodes        : 0             (Left: 1)
    Best LP Bound  : 88.61889465  (Gap: 0.00%)
    Best Solution  : 88.61889465
    Solving Time   : 0.06 sec     (Peak Mem: 0.0 M
    Program Status : Optimal
    Solver Status  : Normal completion

Total Time       : 0.00 sec
Memory Used      : 81.2 Mb
Memory Free      : 1530.2 Mb
```

i	j	k	x
0	2	2	1
0	5	1	1
0	7	3	1
1	0	2	1
2	4	2	1
3	1	2	1
4	3	2	1
5	6	1	1
6	0	1	1
7	0	3	1

j	k	z
0	1	199.0
0	2	283.0
0	3	38.0
1	2	380.8
2	2	306.0
3	2	361.2
3	3	#####
4	2	331.2
5	1	224.6
6	1	239.6
7	3	64.2

	CustomerDemand	ReleaseTime	DueTime	ServiceTime	0	TardinessPenaltyCost
0			330			
1	10.0	283.0	304	5.00	15.23	1.00
2	7.0	129.0	247	5.00	18.00	1.00
3	13.0	181.0	265	5.00	22.36	1.00
4	19.0	252.0	295	5.00	25.00	1.00
5	26.0	193.0	267	5.00	20.62	1.00
6	3.0	199.0	258	5.00	11.18	1.00
7	5.0	38.0	235	5.00	21.21	1.00

	0
0	
1	15.23
2	18.00
3	22.36
4	25.00
5	20.62
6	11.18
7	21.21

MaxCustomersNumber	7
VehicleCapacity	50.0
MaxVehiclesNumber	3

FullDistance

	0	1	2	3	4	5	6	7
0		15.2	18.0	22.4	25.0	20.6	11.2	21.2
1	15.2		32.6	14.6	32.2	32.2	24.8	21.0
2	18.0	32.6		34.4	20.2	23.9	16.4	36.2
3	22.4	14.6	34.4		25.0	42.7	33.5	35.4
4	25.0	32.2	20.2	25.0		41.2	31.6	46.1

FIGURE 3.2: Results for 7 customers.

The result for seven customers is shown as Figure 3.2. The routes formed are: 0-5-6-0, 0-2-4-3-1-0 and 0-7-0 which produce 88.619 unit cost of travelling distance and tardiness. The result is based on setting $\alpha = 0.5$ and vehicle capacity of 50. The optimal solution to the model was found in 0.06 seconds involving 327 constraints and 224 variables.

As results were successfully obtained for five and seven customers, the experiment is continued with different number of customers. We have tried out for 10 customers. Unfortunately, the result was not as expected as subtour occurs. The subtour constraint has been included, but the subtour problem is still there. Since the root of this subtour problem could not be found, thus, heuristics are the best option in order to find the solution.

Sousa *et al.*[121] has solved the capacitated vehicle routing problem with time window using mixed integer linear programming for small number of clients too. The problem solved was for six clients for up to three vehicles subject to only 10 constraints by GLPK software. As a conclusion, the author suggested that greater problems should be solved by other techniques like heuristics.

As mentioned in the previous sections, we will generate the solution for 56 instances of 100 customers with 3 different aims and fleet-cases. Heuristics will be used in order to get the near optimal solution within reasonable computing times. The heuristics for solving the VRPRDD with larger data will be dealt with in the next chapter.

3.7 Summary

This chapter reviews one of the problems studied, i.e. VRPRDD, beginning with the problem description for a small example and the mathematical formulation. Data from benchmark problem are modified to suit the requirements of the problem studied. Nine different vehicle routing problems have been designed to see the performance of the algorithms that will be proposed. These are routing problems with big-, medium- and small-fleet size cases, each of which is treated for three different aims; solving the problem by focusing on tardiness cost, travelling cost and both cost equally. Validation of the problem formulation has been done by using AIMMS software for the small instances which aim to optimize the costs equally. Due to the subtour problem encountered for 10 customers case, we decide to use heuristics as we need to solve 100 customer points with different aims and fleet-sizes case. The problem will involve thousands and hundreds of thousands of variables and constraints. We really need to go for heuristics which can produce near optimal solution within reasonable computing times. This will be presented in the next chapter in which detailed descriptions of the heuristics used will be explained.

Chapter 4

Heuristics for VRPRDD

4.1 Introduction

This chapter provides a detailed description of the particular heuristics used in solving VRPRDD in our study. The aim is to observe the trade-off between travelling and tardiness costs. The problem assumes that fixed schedule for production sequence is considered. In other words, release date for each customer's order is known at the start. The solution starts with the construction of an initial feasible solution by using the release and due date of customer's order. The parallel insertion method is used to generate an initial routing. This initial solution will be improved by three metaheuristics as discussed here. A pilot study will be used to determine the best parameter in each experiment. The final results for all data sets will then be presented in table form. Based on the results given, the best heuristic for the VRPRDD which generates the best routing cost will be identified.

4.2 Initial Solution: Parallel Insertion method

A heuristic for VRP typically uses a greedy approach to obtain a good initial solution in an efficient manner and then incrementally improve the solution by neighborhood exchanges or local searches. Solomon [120] divided VRP tour-building algorithms into either sequential or parallel methods. In sequential method, routes are constructed one at a time until all customers are scheduled. While in parallel procedure, routes are constructed simultaneously. The number of parallel routes may either be limited to a predetermined number, or formed freely.

Solomon [120] and Compbell and Savelsbergh [22] reported that insertion heuristic can efficiently handle different types of constraints, including time windows and multiple uses of vehicles. Since our problem involves an availability constraint; i.e. release date of customer's order, we adopt the insertion heuristic to construct our initial solution. The initial solution is generated by a parallel insertion heuristic based on the least-cost solution. The method works by constructing several routes at one time. A candidate inserted customer is added in all available routes temporarily without violating the vehicle capacity limit. The least insertion cost of that customer to the route will determine the best vehicle for the candidate customer. Note that the terms route and vehicle are used interchangeably.

One issue in finding an initial solution to a routing problem is to identify the first inserted customer. This customer is referred to as a seed customer. The most commonly-used criterion to determine the seed customer is either the farthest unrouted customer, customer that has earliest due date, or the customer that has earliest allowed arrival time[110]. In this study, the departure time of each vehicle depends on the maximum release dates of customers' order scheduled in the route. Therefore it would be preferable to assign the customers that have the closer release dates to be in the same vehicle. This could be done by sorting the customers according to their orders' release dates. But since the objective function of this study is to minimize the costs and not the number of vehicles used, parallel insertion method will be applied. In addition, Savelsbergh [116] reported that parallel insertion heuristic performed clearly better, in the sense that it succeeded in producing an initial feasible solution with the specified number of routes. Hence, a seed customer is chosen randomly.

As mentioned above, a customer is selected to be inserted on random basis. Analysis in the insertion process is done in two parts: the first deals with the feasibility of an insertion and the second deals with its overall costs. Let assume a route is given by $(0, 1, ..., i, ..., n, 0)$ where 0 represents a depot, n is the number of customers in the route and i is the i^{th} customer visited by the vehicle. Let u be an unrouted customer to be inserted, and i and $i + 1$ are the customers between which u is being inserted. The insertion of u between i and $i + 1$ definitely may affect the current vehicle loading, which may result in an unfeasible tour. Generally, it affects the departure time of vehicle as well as it depends on the maximum release date of scheduled customer's order. Indirectly it changes the arrival times at customers/vertices $i + 1, i + 2, ..., 0$, which may result in increasing the number of tardy jobs, i.e. tardiness cost.

As we are constructing the routes simultaneously, the required number of vehicles, k, will be determined first which is calculated as number of total customer orders divided by vehicle capacity, as equation (3.16). Initialize k routes R_t, where $t = 1, ..., k$. Let $J = \{R_1, ..., R_k\}$. For each customer i not yet associated with a route and for each feasible route $R_t \in J$, compute the route costs $Cost_{it}$, i.e, insertion cost of customer i in route t and $Cost_{i*t} = min_t\{Cost_{it}\}$. Associate customer i with R_t* and repeat the process until all customers are routed.

Customers will be added to these vehicles without exceeding the vehicle capacity limit, Q. The customer i will be inserted into the route R_t with the best position which gives the least insertion cost, $Cost_{i*t}$. Insertion cost is computed as a cost difference when unrouted customer is inserted to the route. The travel distance, time and cost are calculated from depot to the customer's location and back to depot. On each possible route, an inserted customer's position or possible sequence will be tested and insertion cost is calculated. Then, a customer will be inserted to the route with the best position which generate the least-cost. The insertion process is continued until all customers are routed.

If the available number of vehicles is not sufficient due to capacity constraints, then a new vehicle will be added. Having a few choices of vehicle is an advantage to the customer where the solution for the best insertion can be explored as a larger solution space gives more chances in determining the best. The insertion process can be captured in the following algorithm 1:

The algorithm 1 starts by determining a minimum number of vehicles as in equation (3.16). Then, we initialize the number of current vehicle as zero. As soon as the candidate of inserted customer is verified (randomly listed), an empty vehicle is called up. This is carried out in lines 5 to 7. At the beginning of the insertion process, the seed customer or the first inserted customer will be routed in the first constructed route. Then, the next customer will be routed. For the second customer onwards, a temporary insertion process will be done in several routes simultaneously. On each particular route, the vehicle departure time is determined and the travelling and tardiness costs are calculated at every possible position of the current tested route as long as the capacity constraint is not violated. Once the best position in a particular route is found based on the least incremental cost on the overall possible routings, the candidate customer will be scheduled in that particular vehicle. The process will be continued until all customers are routed.

Algorithm 1 Parallel Insertion Heuristic

```
1: Determine the minimum number of vehicles, $MinVeh$
2: Initial $VehicleNo = 0$
3: Randomly sort the customer's list.
4: for $Cust = 1$ to $MaxCust$ do
5:   Choose the inserted customer, $InsertCustomer[Cust]$
6:   if $VehicleNo < MinVeh$ then
7:     Add a new vehicle, $VehicleNo$++;
8:   end if
9:   if $Cust = 1$ then
10:    Insert $InsertCustomer[Cust]$ into the current route; i.e. $VehicleNo = 1$
11:    Determine the departure time of vehicle and calculate the travel and tardy
       costs.
12:    Update the route information.
13:    Proceed to the next $Cust$.
14:  end if
15:  ------------------------------------------------------------------------
16:  Trial of possible customer's insertion in all existing routes.
17:  ------------------------------------------------------------------------
18:  for $Veh = 1$ to $VehicleNo$ do
19:    if $Veh$ is an empty vehicle, then
20:      Calculate the insertion cost of customer $InsertCustomer[Cust]$ in route
         $Veh$.
21:    else
22:      $NewLoad[Veh] = Load[Veh] + Load[InsertCustomer[Cust]]$
23:      if $NewLoad[Veh] \leq VehicleCapacity$ then
24:        Try every possible position to insert the customer $InsertCustomer[Cust]$
           into route $Veh$.
25:        Determine the departure time of vehicle and calculate the travel and tardy
           costs at each position.
26:        The cheapest cost or the least-cost solution determine the best position of
           customer $InsertCustomer[Cust]$ in route $Veh$.
27:      end if
28:    end if
29:    Update the temporary current best route information.
30:  end for
31:  Determine the best route for $InsertCustomer[Cust]$, $BestVeh$ which generate
     the lowest incremental cost of total routings.
32:  Then, insert the $InsertCustomer[Cust]$ into the route $BestVeh$.
33:  Update the route $BestVeh$ information.
34: end for
```

4.3 Local Search Neighborhoods

In this section we reassess some neighborhoods proposed for our vehicle routing problem. The VRP neighborhoods can be divided into two major categories: single-route and multi-routes improvements. For this we use the terminology of Laporte and Semet [74] which gave the same meaning of intra- and inter-route improvement, respectively, which will be used in this thesis. The intra-route improvement performs changes to one route at a time such as permute customers within a route while the latter involves exchange and move customers between two or more routes. A more complete survey of neighborhoods can be read in Bräysy and Gendreau [18].

Using the above terminology, we define the moves that have been used in our local search neighborhoods. The first two are the moves that use inter-route improvement and the next two are the intra-route improvement. All the moves made are based on the best improvement strategy. Each neighborhood is searched in lexicographic order and the search is repeatedly applied until no more improvement to the current solution is possible.

Inter-Route Improvement

1. Relocate:
   A move consists of selecting a customer i, currently inserted in a route k, removing it from the route and reinserting it into a new k', which reduces the total cost of these two routes. It might be done one by one or simultaneously. There are a few versions used in this study as follows:

   - RelocateOrder
     The relocated customer will be chosen one by one in lexicographic order of each constructed route. Beginning with the first route, the algorithm tries to reallocate the customer in the selected route one by one into another route. Finally, the best insertion customer which generates the least cost insertion that improves the current solution will be chosen.
   - RelocateRD:
     Starting from the first vehicle which has been randomly ordered, the jobs in the chosen vehicle will be ordered according to their release dates (RD) in increasing order. Then, the gaps between release dates are calculated. The last half calculated gaps will be selected for comparison in which the biggest gap will be chosen. Once the gap has been identified, the algorithm will delete the jobs from the biggest gaps onward. This is illustrated in Figure 4.1.

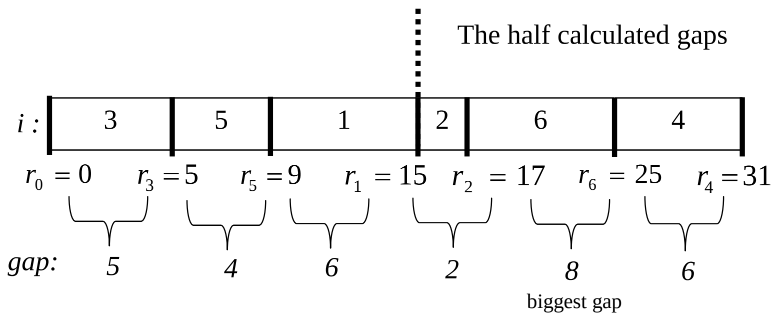


FIGURE 4.1: An example of RelocateRD version for one route/vehicle.

Figure 4.1 shows the scheduled customers in one route/vehicle. Notation i refers to the job number/customer, r_i is the release date of each job i and each gap is calculated by subtraction between two release dates. Select the biggest gap in the last half of calculated gaps range where the jobs 2, 4 and 6 will be the candidates for relocation.

- RelocateCost:
  In this neighborhood, we will relocate a customer which contributes the most cost in their route:

$$\text{Cost of job } i = \text{RouteCost with job } i - \text{RouteCost without job } i.$$

  The cost of the job i is calculated as the cost of the route the job i minus the route cost without the job i. Thus, in this neighborhood, we will only relocate the job at one time.

2. Swap:
   A customer from one route may be swapped with a customer from a different route. Any swapping that improves the total cost will be chosen.

Intra-Route Improvement

3. 2-Opt:
   In 2-Opt, two edges are removed from a route and reconnect two paths in a new way. In other words, the 2-opt operator improves a single route by replacing two of its arcs by two other arcs.

4. 3-Opt:
   It is quite similar to 2-Opt. Instead of two, three edges will be removed and three new paths created without changing the remaining directions.

4.4 Relative Percentage Deviation, *dev*

Before we proceed further, it is important to state here that the comparison for each heuristic will be based on the relative percentage deviation, *dev*. This is the percent deviation of one solution from the best solution found. These best solutions found are used as lower bounds for the investigated cases. The deviation is computed as follows:

$$dev = \frac{F_{Heuristic} - F_{Best}}{F_{Best}} \times 100\% \tag{4.1}$$

where $F_{Heuristic}$ is the best solution found by that particular heuristic and F_{Best} refers to the overall best solution found for the investigated cases. The performance of the algorithm will be measured using average deviation, A_{dev}, computed as:

$$A_{dev} = \frac{\sum\limits_{h=1}^{H} dev_h}{H} \tag{4.2}$$

where H is the number of investigated cases for each algorithm.

4.5 Heuristic 1: Variable Neighborhood Search

VNS is a metaheuristic for solving optimization problems based on systematic changes of structures within a search [20][62]. The basic VNS consists of both diversification and intensification stages. Diversification is accomplished by applying shaking, i.e. random moves in large neighborhoods, while the local search component is used for intensification. Algorithm 2 shows the procedure of VNS used in the problem studied.

According to the basic VNS scheme, a series of neighborhood structures, N_k, is first selected. These structures represent the neighborhoods around any point $x \in X$ of the solution space. Then the local search neighborhoods, N_l, is used and this leads to a local optimum x. For both types of neighborhoods, N_k and N_l, the move is accepted and the search will begin again at the first neighborhood $N_1(x)$ if and only if an improvement on the incumbent solution is found. Otherwise, the procedure is iterated using the next neighborhood. The searching process continues until a stopping condition is satisfied.

Algorithm 2 Variable Neighborhood Search

```
1: Initialization:
     • Select the set of neighborhood structures $N_k$ for $k = 1,, k_{max}$, that will be
       used in the shaking phase, and
     • identify the set of neighborhood structures $N_l$ for $l = 1,, l_{max}$ that will be
       used in the local search; then,
     • find an initial solution $f(x)$ and improve it by using local search neighbor-
       hoods;
     • choose a stopping condition;
2: for $k = 1$ to $k_{max}$ do
3:   Shake the solution to generate $x'$ solution from the $k^{th}$ neighborhood $N_k(x)$ of $x$
     ($x' \in N_k(x)$).
4:   for $l = 1$ to $l_{max}$ do
5:     Local optimum: Find the best neighbour $x''$ of $x'$ in $N_l(x')$.
6:     If an improvement is found, move ($x' \leftarrow x''$) and let $l \leftarrow 1$. Otherwise $l \leftarrow l+1$.
7:   end for
8:   Move or not: If local optimum is better than the incumbent $f(x)$, move ($x \leftarrow x'$)
     and $k \leftarrow 1$. Otherwise $k \leftarrow k + 1$.
9: end for
```

As known, the set of neighborhoods used for shaking is the heart of VNS [64]. To define a neighborhood for the incumbent solution an appropriate operator must be specified. In our VNS, a relocate operator is used to shake the solution. Instead of one customer, we relocate k customers simultaneously where $k = 1, ..., k_{max}$. For instance, $k = 1, 2, 3$ and 4 represent the first, second, third and fourth shaking neighborhoods respectively. These neighborhoods operate on the relocate operator basis. Again, for instance $k = 1, 2, 3$ and 4 also signify the neighborhood that relocate one, two, three and four customers simultaneously.

To diversify the search, these k customers are selected randomly. They are removed from their current routes one by one. The cost for each route is then updated after each removal. Next, those customers who were removed are re-inserted into routes selected randomly by using the least-cost insertion method. In this VNS, a customer may not be inserted into their previous route. We first search randomly for other possible routes to insert the customer without violating the vehicle capacity limit. As a result of the random selection, there is a possibility that a relocated customer cannot fit into the existing route due to capacity constraint. Therefore, a new empty vehicle is added if no possible insertion can be performed into any one of the existing vehicles.

A solution obtained at the shaking phase is submitted to the local search procedure to search for the local optimal solution. In our local search, several neighborhoods were used to improve the shaking solution.They were relocated with different procedures: swap, 2-Opt and 3-Opt as have been explained in the previous section. After going through the shaking and the local search improvement procedures, the result is compared to the incumbent solution. The comparison is done to determine whether the original solution should be replaced. A solution with an improved cost is accepted. The algorithm is terminated once the computer running time exceeds 30 seconds.

To examine several parameter settings for the heuristics used, six out of the 56 modified Solomon's instances with 100 customers were implemented as our pilot study. Those six instances were selected from each class instance, i.e. mod_C101, mod_C201, mod_R101, mod_R201, mod_RC101 and mod_RC201. We believe that by choosing different classes of problem instance with varying results will give us some general conclusion across all problem instances. By varying the values of the parameters in this pilot study, an excessive running times over all heuristics on all instances could be avoided.

4.5.1 Investigations of k_{max}

We first attempt to determine the maximum number of relocated customers, k_{max}, at the same time determine the number of shaking neighborhoods, N_k. In our VNS, we are trying to relocate several customers by searching potential neighborhoods which might return better possible solutions. We start with $k_{max} = 1, 2$ and 3. It is essential to note here that k_{max} represent the number of our shaking neighborhoods. For example, if $k_{max} = 3$, there are three shaking neighborhoods where N_1, N_2 and N_3 represent a move that relocates one, two and three customers at a time, respectively.

Note that the solution of each k_{max} is obtained based on average values of 10 runs due to the random numbers used. For each k_{max}, an average solution for those six instances is calculated. The result of each k_{max} is represented as an average over the average of 10 solutions. We would gradually increase the k_{max} value if it improves the result on the previous k_{max}. We stop the experiment when there is no sign of a better solution found after several trials of k_{max} values have been made. This has been shown clearly in Table 4.2 which will be explained later.

Appendix A shows the average solution of 10 runs done on our pilot study. The results generated are summarized and shown in Table 4.1 below.

TABLE 4.1: VNS Parameter Setting-Shaking Phase using Cost

k_{max}	$a = 0.3$			$a = 0.5$			$a = 0.7$		
	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$
1	$\underline{618.98}$	1363.72	$\underline{1740.71}$	$\underline{2155.66}$	2031.57	1760.21	4319.05	3073.05	$\underline{1837.38}$
2	623.92	$\underline{1359.75}$	1744.85	2158.86	$\underline{2026.95}$	1760.14	$\underline{4311.13}$	3071.41	1838.05
3	629.48	1364.11	1743.79	2159.66	2043.62	1765.57	4322.94	$\underline{3071.02}$	1844.83
4	622.37	1361.28	1743.58	2177.83	2046.39	1763.67	4327.22	3074.87	1842.61
5	624.28	1366.43	1745.00	2174.51	2043.96	1764.85	4318.55	3076.12	1841.59
6	628.81	1367.52	1746.43	2188.16	2037.87	$\underline{1758.92}$	4318.04	3075.54	1841.22
Min	618.98	1359.75	1740.71	2155.66	2026.95	1758.92	4311.13	3071.02	1837.38

Table 4.1 shows the average solution of our pilot study done on different fleet sizes; big, medium and small-fleet size ($a = 0.3, 0.5$ and 0.7 respectively) each having three different weights; $\alpha = 0.1, 0.5$ and 0.9 on their objective functions. For comparison, the results are shown as relative percentage deviation.

TABLE 4.2: VNS Parameter Setting-Shaking Phase using Deviation (%)

k_{max}	$a = 0.3$			$a = 0.5$			$a = 0.7$			Average
	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	Dev.(%)
1	0.00	0.29	0.00	0.00	0.23	0.07	0.18	0.07	0.00	0.09
2	0.80	0.00	0.24	0.15	0.00	0.07	0.00	0.01	0.04	0.14
3	1.70	0.32	0.18	0.19	0.82	0.38	0.27	0.00	0.41	0.47
4	0.55	0.11	0.17	1.03	0.96	0.27	0.37	0.13	0.28	0.43
5	0.86	0.49	0.25	0.87	0.84	0.34	0.17	0.17	0.23	0.47
6	1.59	0.57	0.33	1.51	0.54	0.00	0.16	0.15	0.21	0.56

Table 4.2 presents the average deviations of all problems which were used to draw some general conclusion that will be used in our VNS. In Table 4.2, the *Average Dev.* is calculated. It shows that the deviation among the tested k_{max} are found to be $< 1\%$. It can be seen that increasing the k_{max} in our shaking phase would not give any better solution. Thus, from now on, we will set $k_{max} = 1$ in our VNS shaking phase for our next investigation.

4.5.2 Local Search Order

Our next investigation is to look for a better order of local search neighborhoods, N_l. For this experiment, it can be summarized as Table 4.3. We use two different settings in our VNS, setting 1 and setting 2. According to the previous test, both settings will use the same shaking neighborhoods, i.e. $k_{max} = 1$ as it was found to be most beneficial. Setting 1 uses four neighborhoods while setting 2 uses five neighborhoods in their local search. Each of these neighborhoods has been explained in section 4.3. The neighborhood used for a setting is marked as 1 and 0 if not.

Table 4.3: VNS Settings-Local Search Neighborhoods

	Setting 1	**Setting 2**
Shaking Neighborhoods:		
k_{max}	1	1
VND Neighborhoods:		
Swap	1	1
2-Opt	1	1
3-Opt	1	1
Relocate (Lexicographic order)	1	0
RelocateRD	0	1
RelocateCost	0	1

Setting 1

We begin with VNS setting 1 which use swap, 2-Opt, 3-Opt and relocate as our VND implementation. As we have four neighborhoods, we try all 24 possible orders. For simplicity, we use 1, 2, 3, 4 to represent the neighborhoods swap, 2-Opt, 3-Opt and relocate respectively.

The full results are attached in Appendix B. It shows the average solutions of all fleet size problems, each with different weights. These results are summarized in Table 4.4 as relative percentage deviation.

Table 4.4: Neighborhood's Order: Setting 1 (Deviation %)

Order	$a = 0.3$			$a = 0.5$			$a = 0.7$			Average
	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	Dev.(%)
1 2 3 4	6.11	0.89	1.31	2.77	0.68	0.70	0.46	2.10	0.83	1.76
1 2 4 3	0.00	0.64	0.79	3.06	1.22	1.05	0.83	0.78	0.37	0.97
1 3 2 4	0.70	0.05	1.49	1.04	0.63	0.30	0.00	1.66	0.68	0.73
1 3 4 2	5.28	0.70	0.99	1.39	0.69	0.32	1.07	1.63	0.33	1.38
1 4 2 3	7.34	0.55	1.23	3.55	0.06	0.03	0.94	0.00	0.75	1.60
1 4 3 2	7.22	0.64	1.38	3.25	2.18	0.79	1.06	1.76	0.89	2.13
2 1 3 4	4.87	0.51	0.65	1.41	1.00	0.68	0.05	0.88	0.52	1.18
2 1 4 3	3.79	0.00	1.50	3.64	1.77	0.36	0.28	2.02	0.00	1.49
2 3 1 4	1.84	0.95	1.04	0.00	1.89	0.13	0.19	1.32	0.11	0.83
2 3 4 1	8.07	1.55	1.53	3.93	1.89	0.25	1.69	1.86	1.03	2.42
2 4 1 3	5.24	0.97	0.00	2.91	0.00	0.55	0.94	0.37	0.81	1.31
2 4 3 1	5.59	0.80	0.85	2.10	2.30	0.50	2.08	2.04	0.75	1.89
3 1 2 4	4.25	0.77	1.24	0.98	1.29	0.00	1.27	1.58	0.31	1.30
3 1 4 2	6.73	1.25	1.09	4.18	1.43	0.97	1.19	0.89	0.97	2.08
3 2 1 4	2.84	0.28	0.40	0.00	0.27	0.34	0.78	0.43	0.60	**0.66**
3 2 4 1	6.93	1.54	1.36	3.19	1.55	0.76	2.09	2.07	0.54	2.22
3 4 1 2	3.49	0.58	0.92	3.20	1.45	0.63	1.05	1.66	0.52	1.50
3 4 2 1	5.63	0.12	0.82	4.47	1.42	0.63	1.04	2.37	0.20	1.85
4 1 2 3	7.10	1.30	1.19	1.43	2.04	1.16	1.90	2.13	0.88	2.12
4 1 3 2	6.08	1.43	1.11	2.86	1.85	0.67	0.65	2.22	0.31	1.91
4 2 1 3	4.40	1.13	0.55	2.20	0.75	1.26	0.98	1.48	0.57	1.48
4 2 3 1	8.92	0.69	0.25	3.69	0.43	0.61	1.59	1.86	0.85	2.10
4 3 1 2	6.78	1.37	1.29	2.66	1.73	0.27	2.18	1.43	0.36	2.01
4 3 2 1	6.60	1.43	1.49	3.39	1.08	0.65	1.68	2.00	0.62	2.10

Table 4.4 shows the average solution cost across all weights in each fleet-size case. It can be seen that the order 3 2 1 4 which represent 3-Opt, 2-Opt, Swap and Relocate return the best order for our VND in setting 1.

Setting 2

Now, we will look into VNS with setting 2. We use five neighboods in setting 2; Swap, 2-Opt, 3-Opt, RelocateRD and RelocateCost represented as 1, 2, 3, 4 and 5 respectively. There are 120 possible permutations for five neighborhoods' order. Due to time constraints, it is not possible to try all of these 120 combinations. Hence, we used three strategies in our investigation in VNS Setting 2 as below:

- ***Best Order.***
  Based on the best order found in Setting 1, we generalized the results to five neighborhoods. We do this by keeping the subsequence 3 2 1 and extend it by the two extra combinations of 4 5 and 5 4. In other words we replace the Relocate with RelocateRD-RelocateCost and RelocateCost-RelocateRD. Hence, we will use these following orders; 3 2 1 4 5 and 3 2 1 5 4 which represent 3-Opt, 2-Opt, Swap, RelocateRD, RelocateCost and 3-Opt, 2-Opt, Swap, RelocateCost, RelocateRD respectively. The results are given in Appendix C and have been summarized in Table 4.5 below.

Table 4.5: Neighborhoods order on Best Order basis (cost based).

Fleet Size	weight, α	3 2 1 4 5	3 2 1 5 4
Big, $a = 0.3$	0.1	612.46	592.37
	0.5	1361.91	1359.28
	0.9	1742.26	1739.76
Medium, $a = 0.5$	0.1	2153.87	2139.02
	0.5	2010.94	2028.52
	0.9	1743.07	1765.19
Small, $a = 0.7$	0.1	4246.98	4286.55
	0.5	3049.03	3029.32
	0.9	1833.21	1826.10

- ***Best First.***
  For this strategy, we run our VNS by using VND neighborhood individually which will identify the value of each local search. Table C.10-C.18 in Appendix C show the performance of each local search. The results are summarized in Table 4.6 below.

TABLE 4.6: Best First basis: Value of each local search

Fleet Size	weight, α	Swap	2Opt	3Opt	RelRD	RelCost
Big, $a = 0.3$	0.1	633.91	636.03	637.02	635.51	624.98
	0.5	1388.86	1371.76	1373.08	1368.47	1365.86
	0.9	1750.82	1750.54	1753.02	1749.64	1737.81
Medium, $a = 0.5$	0.1	2197.94	2198.86	2198.75	2156.89	2176.23
	0.5	2056.28	2056.50	2055.53	2053.94	2036.57
	0.9	1775.32	1774.79	1775.44	1760.83	1765.69
Small, $a = 0.7$	0.1	4343.04	4363.27	4372.08	4328.98	4349.06
	0.5	3062.38	3088.89	3085.63	3069.24	3083.21
	0.9	1847.83	1849.27	1848.72	1846.30	1839.56
Average Sol.		2117.38	2121.10	2122.14	2107.75	2108.77
Order Best		3	4	5	1	2

Table 4.6 shows the solution cost generated on our pilot study, where VND neighborhood had been run individually. According to these values, we order the neighborhoods in term of the results which the best comes first i.e. RelocateRD, RelocateCost, Swap, 2-Opt, 3-Opt or can be written as 4 5 1 2 3. Next, the costs generated by local search order based on Best First basis is shown in Table 4.7 below.

TABLE 4.7: Best First Basis with Order 4 5 1 2 3

Big-Fleet Size, $a = 0.3$						
weight,α	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
0.1	1173.07	110.421	859.091	818.073	415.671	431.579
0.5	1994.12	586.455	1253.27	1400.29	1329.99	1619.23
0.9	2201.58	960.257	1557.38	1883.08	1740.77	2130.8
Medium-Fleet Size, $a = 0.5$						
weight,α	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
0.1	4037.45	104.815	2442.36	2816.57	1315.27	2371.5
0.5	3374.37	589.444	1964.9	2463.41	1614.04	2223.19
0.9	2341.92	957.715	1557.82	1855.27	1743.6	2139.45
Small-Fleet Size, $a = 0.7$						
weight,α	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
0.1	8766.19	105.307	4084.32	5297.76	3673.14	4162.46
0.5	5792.01	584.806	2938.17	3814.62	2182.64	3187.46
0.9	2557.71	957.838	1637.33	2012.36	1729.12	2108.33

- ***Random***.
  In this case the VND is ordered randomly. We reset the random order for VND after each shaking. We run for ten times and take the average from these ten solutions. Detailed results can be found in Appendix C and has been summarized in Table 4.8.

TABLE 4.8: Random Order Basis

Big-Fleet Size, $a = 0.3$						
weight,α	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
0.1	1121.91	110.13	833.70	792.54	414.59	430.34
0.5	1983.23	579.90	1272.60	1392.87	1329.66	1610.94
0.9	2197.35	957.68	1545.77	1886.78	1724.73	2138.85
Medium-Fleet Size, $a = 0.5$						
weight,α	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
0.1	4039.49	109.47	2414.64	2739.60	1319.73	2283.13
0.5	3406.84	582.32	1960.05	2418.19	1612.56	2217.75
0.9	2345.10	957.28	1553.70	1851.78	1716.30	2139.09
Small-Fleet Size, $a = 0.7$						
weight,α	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
0.1	8690.58	109.80	4114.85	5206.06	3591.99	4042.71
0.5	5748.65	583.21	2919.54	3785.44	2161.20	3137.70
0.9	2597.11	960.52	1629.76	2009.82	1723.40	2117.84

Table 4.8 shows the solution of those six instances of each fleet size and weight with random ordering. For comparison, an average solution of those six instances is calculated using these three strategies. These have been summarized in Table 4.9 below.

TABLE 4.9: Experiment on Local Neighborhoods Order of VNS Setting 2 (cost based)

Fleet Size	weight, α	Best Order		Best First	Random
		3 2 1 4 5	3 2 1 5 4	4 5 1 2 3	
Big, $a = 0.3$	0.1	612.46	<u>592.37</u>	634.65	617.20
	0.5	1361.91	<u>1359.28</u>	1363.89	1361.53
	0.9	1742.26	<u>1739.76</u>	1745.64	1741.86
Medium, $a = 0.5$	0.1	2153.87	<u>2139.02</u>	2181.33	2151.01
	0.5	<u>2010.94</u>	2028.52	2038.23	2032.95
	0.9	<u>1743.07</u>	1765.19	1765.96	1760.54
Small, $a = 0.7$	0.1	<u>4246.98</u>	4286.55	4348.20	4292.66
	0.5	3049.03	<u>3029.32</u>	3083.28	3055.96
	0.9	1833.21	<u>1826.71</u>	1833.78	1839.74

Table 4.9 contains the average solutions of VNS setting 2 generated by Best Order, Best First and Random strategies. From this relative percentage deviation of each case is calculated. This average deviation (in %) of all investigated cases for all instances are used to compare the overall performance of the strategies used. This is shown in Table 4.10.

TABLE 4.10: Comparison on Local Neighborhoods Order of VNS Setting 2 (deviation %)

Fleet Size	weight, α	Best Order		Best First	Random
		3 2 1 4 5	3 2 1 5 4	4 5 1 2 3	
Big, $a = 0.3$	0.1	3.39	0.00	7.14	4.19
	0.5	0.19	0.00	0.34	0.17
	0.9	0.14	0.00	0.34	0.12
Medium, $a = 0.5$	0.1	0.69	0.00	1.98	0.56
	0.5	0.00	0.87	1.36	1.09
	0.9	0.00	1.27	1.31	1.00
Small, $a = 0.7$	0.1	0.00	0.93	2.38	1.08
	0.5	0.65	0.00	1.78	0.88
	0.9	0.36	0.00	0.39	0.71
Average Dev.(%)		0.60	0.34	1.89	1.09
#best		3	6	0	0

From Table 4.10, it can be clearly seen that the smallest average deviation is found using Best Order strategy which use 3-Opt-2-Opt-swap-RelocateCost-RelocateRD as local search order that generate the least cost solution for the VNS setting 2. We observe that this neighborhood's order produce six least cost solution out of nine investigated cases with average deviation 0.34%. This also tells us that Best Order approach is the best strategy compared with Best First and Random. It is worth noting that the approach is extended from the full inspection on each possible local search neighborhood's order for VNS setting 1.

Comparison between Setting 1 and Setting 2

Based on the results for settings 1 and 2, we extend our investigation to see which setting gives the better solution. The comparison that we made is given in Table 4.11 where we compile the best order we found in both settings and make the comparison on the best solution only.

Table 4.11 shows that in six of the nine cases (each with different fleet sizes and weights), the best values are in setting 1. In addition, the average deviation of all nine cases shows that setting 1 is slightly better than setting 2. Indirectly it can be seen here that the relocate operator with searching method based on lexicographic order performs

TABLE 4.11: Comparison of VNS Setting 1 and Setting 2

Fleet Size	Weight, α	Setting 1	Setting 2
Big, $a = 0.3$	0.1	1.26	0.00
	0.5	0.00	0.45
	0.9	0.00	0.62
Medium, $a = 0.5$	0.1	0.00	1.68
	0.5	0.25	0.00
	0.9	0.00	0.42
Small, $a = 0.7$	0.1	0.00	1.68
	0.5	0.25	0.00
	0.9	0.00	0.42
Average Dev.(%)		0.25	0.36
#best		6	3

better than that based on the release date or cost; i.e. RelocateRD and RelocateCost, respectively. Thus, we will use the VNS with setting 1 to generate results when tested on the 56 instances which will be shown in the last section of this chapter.

4.6 Heuristic 2: Large Neighborhood Search

Shaw [119] introduced the LNS method for solving VRP with time window. LNS is based on a process of relaxation and re-optimization. The optimization method used is a simple greedy local search procedure. A set of customers visits are chosen and removed from the schedule and reinserted back to the schedule at minimum cost. The process of going back and forth characterizes the move. The moves made within the neighborhood are generated randomly. A move is accepted if it improves the cost solution, and rejected otherwise.

Apart from randomized choice, another possible method for choosing customers for relaxation is choosing the related customers' visit. The term "related" need to be defined according to the problem. Related customers' visit should allow most opportunity for the reinsertion to improve the cost solution. For example, related visits might be referring to customer's locations which are geographically close to one another, or have similar time windows. Further details on relatedness function could be found in [118].

Our LNS will select a set of customers to be removed at random and rebuild the solution by inserting these customers into routes one at one time by using a greedy heuristic, i.e. least-cost insertion method. The insertion with the least cost will be chosen and the process will be repeating until all customers have been inserted. The algorithm of our LNS here is illustrated in Algorithm 3 below.

Algorithm 3 Large Neighborhood Search

```
1: $BestSolutionSoFar, s^b \leftarrow InitialSolution, s$
2: Set $MaxRemoveCustomer = K$
3: Set $RemoveCustomer = 1$
4: **repeat**
5:   Delete $RemoveCustomer$ from their current route and update the cost, $d(s)$.
6:   Re-Insert $RemoveCustomer$ into the routes which give the least-cost insertion, $r(d(s))$.
7:   Update the overall routes cost as $s^t = r(d(s))$.
8:   **for** $l = 1$ to $l_{max}$ **do**
9:     Local optimum: Find the best neighbour $s^*$ of in $N_l(s^t)$.
10:    If an improvement is found, move $(s^t \leftarrow s^*)$ and let $l \leftarrow 1$. Otherwise $l \leftarrow l+1$.
11:  **end for**
12:  **if** $s^t < s^b$ **then**
13:    Move the solution, $s^b \leftarrow s^t$ and keep searching with the same number of $RemoveCustomer$.
14:  **else**
15:    $RemoveCustomer \leftarrow RemoveCustomer + 1$
16:    **if** $RemoveCustomer > K$ **then**
17:      Reset $RemoveCustomer = 1$
18:    **end if**
19:  **end if**
20: **until** $RunTime > 30s$
```

In line 1, the global best solution is initialized. This is followed by setting the maximum number of customers to be removed, $MaxRemoveCustomer$, denoted as K. In line 5, $RemoveCustomer$ customers are deleted from their current route. These customers are then re-inserted one by one into routes which returns the least insertion cost. In line 7, the new solution is evaluated, s^t by going through the local search process to find the local optima of s^t. In line 12, the heuristic determines whether the solution found in line 10 should become the best solution. If the solution is accepted, the searching process will continue by using the same number of $RemoveCustomer$. Otherwise, $RemoveCustomer$ is increased by one in the next iteration.

An upper limit of K was placed on the value of $RemoveCustomer$. In other words, the set of solutions can be reached from current solution, s by relocating at most K customers. This scheme increases $RemoveCustomer$ only when there is no improvement found in the current $RemoveCustomer$. The scheme used is similar to that of Shaw [119] except that the degree of destruction is gradually increased. The algorithm terminated once the CPU searching time is greater than 30 seconds.

In brief, we use the concept of VNS of systematically changing the neighborhoods to the number of customers to be moved.

4.6.1 Investigations of $MaxRemoveCustomer, K$

As in the VNS parameter setting, the first experiment we did is to determine the most number of relocated customer, K in LNS neighborhood. Here, we are trying to relocate a set of customers. As we know, the main idea behind the LNS heuristic is that the large neighborhood allows the heuristic to navigate in the solution space easily even if the instance is tightly constrained. Thus, we are trying to relocate as many as possible from the total number of customers.

We are using the same setting found to be the best in VNS in our LNS. As in VNS, local search technique is used to find the local optimum of the solution. To make it possible for comparison later, we used the same initial solution, the same local search neighborhoods and also the same neighborhoods' order as in VNS. Ten independent runs are executed for each instance on different fleet sizes and weights. The results representing the average of ten solutions can be found in Appendix D.

Tables D.1, D.2 and D.3 show the averages of least-cost solutions for each fleet size and weight. We start with $K = 5$ to 85 by an increment of 5 to see if any better solution could be found. The relative percentage deviation (%) of each case is calculated and presented in the Tables 4.12, 4.13 and 4.14.

TABLE 4.12: Big Fleet Size, $a = 0.3$

Focusing on Tardiness Cost, $\alpha = 0.1$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	6.65	1.51	9.06	20.56	4.65	13.45
10	17.31	1.31	9.64	32.77	4.76	15.22
15	15.49	1.51	10.07	13.34	4.76	15.16
20	11.96	0.48	7.55	32.77	4.76	15.22
25	18.32	1.51	6.87	32.77	0.00	15.22
30	5.66	1.51	12.28	30.18	4.76	15.22
35	0.00	1.51	9.37	32.77	4.76	15.22
40	10.40	1.51	4.83	32.77	4.76	1.80
45	1.58	1.51	8.97	17.99	4.76	12.72
50	9.11	1.36	9.13	1.70	4.76	0.99
55	6.90	1.51	12.28	5.47	2.89	0.00
60	19.49	1.51	12.28	31.54	4.76	12.48
65	19.49	0.53	7.77	0.00	4.55	15.22
70	18.02	1.51	0.00	19.78	4.76	3.12
75	17.55	1.51	4.93	32.77	4.76	6.41
80	19.49	0.00	5.33	24.09	3.03	15.22
85	3.13	1.51	12.28	13.16	4.76	11.70
Focusing on Solution Cost, $\alpha = 0.5$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	3.09	4.30	3.29	3.66	1.34	0.22
10	0.50	0.00	3.29	5.37	1.10	3.65
15	2.27	3.63	0.22	5.37	1.34	4.33
20	3.09	4.47	3.29	5.37	0.06	3.11
25	0.85	3.98	3.11	2.51	1.34	2.21
30	2.51	4.21	0.84	3.11	1.34	3.24
35	3.09	2.48	2.73	5.37	1.34	4.60
40	1.01	1.62	3.29	4.98	0.19	1.03
45	3.09	3.02	2.44	5.37	1.14	0.00
50	3.09	4.47	0.07	2.88	0.29	4.60
55	1.88	2.03	2.63	5.37	1.34	3.65
60	3.09	0.79	2.11	3.70	1.34	4.60
65	0.84	3.07	0.75	5.37	1.34	3.52
70	1.13	2.05	3.29	3.65	1.34	4.53
75	0.00	1.40	1.77	0.00	0.53	4.60
80	1.55	3.57	0.00	1.23	0.00	4.00
85	1.11	4.30	2.65	0.72	0.62	0.22
Focusing on Travelling Cost, $\alpha = 0.9$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	0.86	1.37	0.69	1.14	4.97	0.31
10	1.15	1.37	0.85	2.16	1.86	0.67
15	0.76	1.23	0.86	3.11	3.93	0.00
20	1.20	1.37	1.08	2.16	3.97	0.37
25	1.20	0.33	1.08	1.81	1.22	0.30
30	1.20	1.00	0.95	3.02	2.35	0.67
35	0.94	1.37	0.93	3.11	4.36	0.40
40	0.92	1.37	0.39	2.63	5.19	0.67
45	0.66	1.37	0.87	1.34	4.42	0.25
50	1.20	1.37	0.26	0.77	4.17	0.45
55	1.16	1.25	1.08	3.11	4.54	0.63
60	1.20	1.37	1.08	0.00	1.09	0.67
65	0.00	0.50	1.08	1.78	2.82	0.58
70	0.68	0.78	1.08	3.11	4.66	0.67
75	1.20	0.00	0.00	2.45	1.54	0.67
80	0.79	0.87	1.08	2.57	0.00	0.67
85	1.20	0.45	0.69	3.00	3.38	0.67

Table 4.13: Medium Fleet Size, $a = 0.5$

Focusing on Tardiness Cost, $\alpha = 0.1$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	5.75	0.78	6.35	8.91	3.40	0.00
10	4.06	1.36	5.22	8.86	0.00	12.40
15	5.75	1.49	7.18	2.59	4.11	12.93
20	4.27	1.49	7.34	8.91	4.11	8.98
25	5.75	0.46	7.96	0.00	2.64	9.06
30	5.75	1.37	7.20	8.27	4.11	8.29
35	5.75	1.49	7.96	8.91	4.11	5.84
40	5.08	1.49	0.00	7.88	2.08	2.26
45	3.89	1.49	7.96	8.91	2.83	8.94
50	2.78	0.00	0.69	0.37	4.11	12.39
55	0.93	1.10	4.74	8.24	4.11	11.67
60	5.75	1.49	4.12	4.66	1.47	12.93
65	0.00	1.24	4.83	6.06	4.11	8.57
70	2.43	1.49	7.96	8.91	4.11	11.65
75	2.98	1.49	7.45	0.58	4.11	6.28
80	5.75	1.44	4.49	8.91	2.43	3.63
85	5.14	1.49	7.96	8.73	4.11	10.63
Focusing on Solution Cost, $\alpha = 0.5$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	4.91	4.20	4.27	5.23	3.90	4.40
10	2.59	4.20	2.11	1.56	3.90	4.40
15	5.38	0.00	4.40	5.03	3.90	4.40
20	4.79	4.20	1.28	6.79	3.90	4.40
25	5.21	4.15	3.00	7.33	3.90	2.94
30	5.38	2.33	2.64	5.19	3.90	4.40
35	2.89	0.70	0.00	5.62	3.90	4.39
40	0.00	2.39	2.90	3.71	3.90	2.53
45	3.11	1.15	3.20	3.74	2.02	4.40
50	4.44	1.37	3.87	0.00	0.51	2.47
55	2.02	2.90	3.20	1.69	3.90	1.72
60	4.99	2.46	2.42	6.55	3.90	4.40
65	5.38	3.14	4.06	4.60	3.90	1.40
70	5.26	4.15	2.80	0.53	2.00	4.40
75	5.38	2.02	2.50	1.36	3.90	4.40
80	5.38	1.63	2.78	5.00	0.00	4.40
85	5.38	0.33	4.40	2.76	3.02	0.00
Focusing on Travelling Cost, $\alpha = 0.9$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	3.01	0.90	1.82	3.52	4.05	1.78
10	1.16	0.90	1.71	3.61	4.19	1.78
15	3.01	0.90	1.82	3.69	3.28	1.31
20	3.01	0.48	1.82	0.00	1.27	0.04
25	3.01	0.90	0.00	3.74	0.00	1.71
30	0.86	0.22	1.70	3.93	1.87	1.59
35	0.00	0.90	1.26	3.87	4.40	1.43
40	3.01	0.90	0.81	2.18	2.22	1.48
45	2.95	0.61	1.82	3.84	3.21	1.35
50	0.70	0.90	1.05	2.78	3.48	0.45
55	1.38	0.57	1.47	3.93	2.99	1.53
60	1.36	0.61	0.31	2.65	5.46	1.28
65	1.05	0.90	1.82	0.43	2.91	1.78
70	2.89	0.90	1.80	1.28	3.63	0.00
75	0.87	0.61	1.26	3.55	0.15	1.78
80	3.01	0.00	0.87	3.61	2.85	1.22
85	2.18	0.90	0.85	3.93	2.18	1.78

Table 4.14: Small Fleet Size, $a = 0.3$

Focusing on Tardiness Cost, $\alpha = 0.1$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	2.05	1.68	0.91	3.50	11.43	7.89
10	4.32	1.68	3.11	6.00	10.81	9.68
15	2.54	0.35	0.00	6.00	2.54	10.11
20	4.06	0.65	2.32	3.85	9.56	0.00
25	0.48	1.57	2.57	2.30	5.72	3.73
30	3.17	1.68	0.45	4.15	12.14	9.98
35	2.19	1.68	1.75	6.00	13.51	4.60
40	4.32	1.68	0.62	4.75	11.27	9.72
45	2.51	0.69	0.09	0.00	6.83	6.74
50	4.00	1.68	1.42	3.62	0.00	6.45
55	3.62	0.00	2.68	5.85	6.14	10.51
60	2.77	1.60	3.67	6.00	6.77	10.22
65	0.00	0.83	0.29	1.93	7.06	7.27
70	2.56	0.97	3.79	4.68	8.83	4.58
75	0.67	0.93	3.36	4.97	5.33	9.17
80	3.25	1.68	2.69	1.24	6.98	10.51
85	4.16	1.68	1.51	5.98	7.14	2.96
Focusing on Solution Cost, $\alpha = 0.5$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	2.50	1.49	0.91	6.10	4.89	3.79
10	2.51	3.68	4.21	4.46	5.93	1.63
15	2.51	3.68	4.02	5.80	5.01	3.79
20	2.49	0.79	2.43	7.30	5.93	2.99
25	0.77	3.68	0.79	0.03	5.93	3.54
30	2.51	2.11	1.80	0.00	4.51	3.79
35	2.30	0.99	1.18	3.99	2.04	0.00
40	2.32	2.51	3.08	4.47	5.62	3.34
45	2.31	2.08	4.11	2.79	2.84	3.79
50	2.51	1.85	3.36	7.24	5.93	3.79
55	2.51	3.68	2.20	7.30	5.29	3.79
60	1.61	2.17	3.77	5.77	5.06	3.79
65	0.00	3.68	0.00	4.05	0.00	2.66
70	1.99	2.78	0.66	0.26	0.08	3.79
75	1.83	2.56	2.64	7.30	4.79	3.79
80	1.52	0.00	3.75	7.30	2.98	3.79
85	2.33	3.68	4.29	6.41	5.02	3.79
Focusing on Travelling Cost, $\alpha = 0.9$						
K	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
5	4.45	3.17	0.75	3.65	1.18	0.00
10	4.73	3.17	0.52	3.65	0.48	1.07
15	4.99	2.03	1.16	2.78	2.90	1.49
20	4.40	3.17	1.23	3.65	3.18	1.49
25	3.94	3.17	1.17	0.00	3.18	1.01
30	2.90	3.17	1.23	3.65	0.00	1.38
35	2.68	2.88	0.71	3.55	0.60	1.37
40	2.93	2.11	1.07	3.44	1.50	1.49
45	3.60	2.64	0.04	3.46	2.24	1.49
50	3.67	3.17	0.00	3.65	0.43	1.01
55	1.84	2.11	0.80	3.65	1.09	1.49
60	2.82	3.17	0.81	3.46	2.21	0.50
65	3.96	1.74	1.23	0.61	0.76	1.12
70	3.73	2.45	1.13	2.76	1.06	1.49
75	3.88	3.17	1.23	3.46	1.14	1.49
80	0.00	2.26	1.23	3.65	3.18	1.49
85	4.87	0.00	0.33	3.48	3.18	1.49

The average deviations for various values of K are shown in Table 4.15 below.

Table 4.15: Average Deviation of All Investigated Cases

K	Average Dev.(%)
5	3.86
10	4.42
15	4.08
20	4.24
25	3.70
30	4.11
35	3.89
40	3.53
45	3.43
50	<u>2.72</u>
55	3.34
60	4.37
65	3.01
70	3.48
75	3.60
80	3.67
85	3.58

Setting $K = 5$, the average deviation for all generated solution on all fleet sizes and weights cases is about 3.86%. From this initial value, the first three increments of K at 10,15 and 20 do not show any better deviation than $K = 5$. However, an improvement appears at $K = 25$, with a deviation of 3.70%. Continueing, a low deviation appears at $K = 50$. The deviation fluctuates through various values of K. At $K = 85$, it is 3.58% at which we decide to stop searching. The smallest percentage deviation appears at $K = 50$. Therefore it can be concluded that relocating or relaxing 50 customers at a time could give the least-average cost solution. Therefore, $K = 50$ will be used to generate the results of all 56 instances later.

4.7 Heuristic 3: Tabu Search

Tabu Search is a metaheuristic that aims to proceed from a local optimum by allowing non-improving moves. The method examines a trajectory sequence of solutions and moves to the best neighbor of the current solution. Solutions that were recently visited are forbidden or called Tabu for a number of iterations. A memory structure is used in the form of a tabu list to prevent the same sequence of solutions being revisited. Intensification and diversification strategies are embedded in the tabu search procedure to improve the effectiveness of the search. Further exploration of the search space as well as focusing on the more desirable areas increases the possibilities of obtaining the better solution.

In our proposed optimization algorithm, a move used is defined as moving one customer from its original route to another route or as a swapping of two customers that are scheduled in two different routes. These are considered two different neighborhoods procedures. At each step of the search process, a subset of a neighborhood is explored and the neighbor that produces the minimum solution cost becomes the new current solution independently of the fact that the generated cost is better or worse than the current solution. Diversification is applied by allowing the non-improving move which enables further exploration of the search space. Then, the intensification process is done through the 2-Opt and 3-Opt procedures.

Tabu list, i.e., a list of the moves recently executed is used to forbid cycling around a local neighbourhood in the solution space. The number of moves in the list is determined by the tabu list size, denoted by $TabuListMax$. Our tabu list structure makes use of first-in first-out (FIFO) principle to store the tabu moves. This simple structure is shown as below:

$$\langle TabuCust, TabuRoute\rangle$$

This tells us that any move shifting the customer $TabuCust$ into route $TabuRoute$ is forbidden for several times depending on the tabu list size, i.e. $TabuListMax$.

At each iteration the two different move operators are applied to the current solution. From both of these neighborhoods, only the best non-tabu move is chosen. If the number of tabu moves in the tabu list is less than $TabuListMax$, the move is then set as tabu and will not be considered as a valid move until the tabu status is expired.

In tabu search, all kinds of solutions are accepted even if it is worse than before but the best solution found is kept until the end. The heuristic stopped after reaching a maximum CPU time, i.e. 30 seconds. The heuristic is summarized in Algorithm 4.

Algorithm 4 Tabu Search Heuristic

```
1: Set the parameter $TabuListMax$
2: $ListTabu = 0$
3: $BestSolutionSoFar \leftarrow InitialSolution$
4: repeat
5:    Search the best non-tabu move in neighborhood 1 (Relocate); and optimize the
      solution found by 2-Opt and 3-Opt.
6:    Search the best non-tabu move in neighborhood 2 (Swap); and optimize the
      solution found by 2-Opt and 3-Opt.
7:    From both neighborhoods, select the best move that produces the $BestSolution$.

8:    Make move and update the routes information.
9:    if $BestSolution < BestSolutionSoFar$ then
10:       $BestSolutionSoFar \leftarrow BestSolution$.
11:   end if
12:   if $ListTabu < TabuListMax$ then
13:       Add the move in the tabu list.
14:   else
15:       Remove the oldest move in the list.
16:       Add the current move to the list.
17:   end if
18: until $RunTime > 30s$
```

4.7.1 Investigation of $TabuListMax$

In this section, we are looking for the appropriate tabu list size i.e. $TabuListMax$. Thus, we start experimenting with several numbers of $TabuListMax$. The purpose of doing this is to find the best tabu list size for all problem instances.

Our standard tabu search does not involve many parameters. Thus we set $TabuListMax$ parameter equal to 5, 10, 15, 20 and so on. The value will increase in steps of 5 if an improvement can be found. Otherwise, the experiment stops after several setting numbers which show no further improvements.

The test is done for each of our fleet-size and weight. We use the same stopping condition as our VNS and LNS to provide a basis for comparison. The detail results for all settings are given in Appendix E and is summarized in Table 4.16 below. The solutions given are the average cost for six instances used as our pilot study.

TABLE 4.16: Tabu Search Parameter Setting-$TabuListMax$ (cost based)

$TabuList$	Big Fleet Size			Medium Fleet Size			Small Fleet Size		
Max	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$
5	615.74	1365.76	1971.04	2258.30	2043.08	1793.45	4366.30	3104.50	1837.74
10	621.93	1368.56	1954.56	2187.64	2038.12	1777.65	4293.95	3037.46	1809.81
15	606.26	1370.86	1944.24	2199.31	2026.37	1770.93	4347.78	3011.76	1806.73
20	599.95	1358.47	1943.88	2189.11	2017.48	1768.56	4285.07	3018.09	1800.31
25	608.66	1359.07	1938.92	2253.16	2037.13	1762.81	4300.66	3040.00	1800.35
30	606.44	1366.06	1949.91	2238.79	2026.72	1761.44	4314.26	3020.84	1802.85
35	606.54	1363.17	1950.25	2268.80	2035.56	1772.09	4347.86	3040.25	1821.67
40	595.80	1368.28	1952.10	2284.57	2046.61	1765.55	4385.95	3084.19	1814.66

Table 4.16 shows the average solution cost found for each fleet size and weight case. The average solution is shown for each value of $TabuListMax$. To determine the best $TabuListMax$,we calculate the averages of relative deviations for each $TabuListMax$. These are shown in Table 4.17.

TABLE 4.17: Relative Percentage Deviation-$TabuListMax$ (deviation %)

$TabuList$	Big Fleet Size			Medium Fleet Size			Small Fleet Size			Average
Max	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$	
5	3.35	0.54	1.66	3.23	1.27	1.82	1.90	3.08	2.08	2.10
10	4.38	0.74	0.81	0.00	1.02	0.92	0.21	0.85	0.53	1.05
15	1.75	0.91	0.27	0.53	0.44	0.54	1.46	0.00	0.36	0.70
20	0.70	0.00	0.26	0.07	0.00	0.40	0.00	0.21	0.00	**0.18**
25	2.16	0.04	0.00	2.99	0.97	0.08	0.36	0.94	0.00	0.84
30	1.79	0.56	0.57	2.34	0.46	0.00	0.68	0.30	0.14	0.76
35	1.80	0.35	0.58	3.71	0.90	0.60	1.47	0.95	1.19	1.28
40	0.00	0.72	0.68	4.43	1.44	0.23	2.35	2.40	0.80	1.45

From Table 4.17, we can see the average of relative percentage deviation for each fleet size and weight for $TabuListMax$ from 5 to 40. It appears that the tabu list size affects the solution. For the first four values of $TabuListMax$, at 5, 10, 15 and 20, the averages show a decreasing trend generally. After 20, it is observed that there is no more improvements for the next few values. Therefore, we stop the experiment. Note here that both of neighborhoods used to search the solution did not use any random number i.e. searching for each move is done in lexicographic order which is based on the best improvement strategy. Thus, we will set $TabuListMax = 20$ in our proposed tabu search to generate the results for the 56 instances later.

4.8 A Small Example

We use a small example to illustrate the procedure and flow of our proposed heuristics; VNS, LNS and TS. Lets assume Table 4.18 shows an initial solution, x to the problem discussed in which all procedures will start from the same initial solution;

TABLE 4.18: Problem of VRPRDD with 5 customers

Vehicle	Route	Demand	Travelling Cost (A)	Tardiness Cost (B)	Route Costs $(\alpha)(A) + (1-\alpha)(B)$
1	0-2-4-3-0	39	85.6	167.40	126.50
2	0-1-5-0	38	68	38	53
3	0-0	0	0	0	0
Obj.Solution:					179.50

Let say we have 3 available vehicles but only 2 vehicles are used. The other information used are shown in Tables 4.19 and 4.20.

- Number of customers, $n = 5$
- Vehicle Capacity, $Q = 40$
- Penalty Cost=1
- Service Time=5
- weight scale, $\alpha = 0.5$

TABLE 4.19: Problem of VRPRDD with 5 customers

Cust No.	Demand	Relese Time	Due Time
1	10	283	304
2	7	129	247
3	13	181	265
4	19	252	295
5	28	193	267

TABLE 4.20: Euclidean distance between depot and customers' location

Dist.	0	1	2	3	4	5
0	-	15.2	18	22.4	25	20.6
1	15.2	-	32.6	14.6	32.2	32.2
2	18.0	32.6	-	34.4	20.2	23.9
3	22.4	14.6	34.4	-	25	42.7
4	25.0	32.2	20.2	25	-	41.2
5	20.6	32.2	23.9	42.7	41.2	-

Note: Re-calculation on departure time of vehicles, travelling and tardiness cost were based on equations (3.8), (3.9) and (3.10), respectively. This needs to be done once changes occur in the route. There is a local search improvement embedded in all heuristics approached but it is not shown here. Due to the long calculation needed, the steps described for each algorithm is just for one iteration only.

4.8.1 VNS

Based on the initial solution, x found as shown in Table 4.18, the procedure is as follow:

- The search for an improvement begins by choosing a route randomly, say, R_1 and a customer in R_1, say, customer 2 for shaking process.
- Customer 2 is deleted from R_1 and will be inserted into other possible routes, either into R_2 or R_3. Inserted back into R_1 is prohibited. Currently, route for R_1 is 0-4-3-0.
- Lets say the algorithm then randomly choose R_3 for re-insertion process. Check the feasibility of the solution. If the capacity vehicle is not violated then proceed by inserting customer 2 into R_3 which only one possible position, 0-2-0. New calculation for the routes is shown as Table 4.21 below.

TABLE 4.21: Problem of VRPRDD with 5 customers solved by VNS

Vehicle	Route	Demand	Travelling Cost (A)	Tardiness Cost (B)	Route Costs $(\alpha)(A) + (1-\alpha)(B)$
1	0-4-3-0	32	72.4	42	57.20
2	0-1-5-0	38	68	38	53
3	0-2-0	7	44.80	0	22.40
Obj.Solution:					132.60

From the Table 4.21, it can be seen that R_2 cannot be chosen for the re-insertion process due to the infeasible solution, i.e, capacity of vehicle does not permitted.

- Shaking solution is calculated which is the weighted sum of route cost of R_1, R_2 and R_3. The shaking solution, x' is 132.60 cost unit.
- It is clear that $x' < x$, which means the current solution is better than the incumbent (for this iteration, it refers to the initial solution). Therefore, $x \leftarrow x'$ and searching is continued in this 1st neighborhood again.

4.8.2 LNS

The algorithm starts by setting the maximum number of removed customer, say, $K = 3$. To begin, let variable $Remove = 1$. As explained, LNS will remove up to K customers. The number of $Remove$ will gradually increase if no better improvement is found. However, it will reset $Remove = 1$ once better improvement has been made. The remove and insert processes are done on random basis.

For initial solution, refer to the Table 4.18.

- Randomly choose *Remove* customer to be removed. Say, customer 4 from R_1. Thus, R_1: 0-2-3-0. Update the route cost after removing it.

- Then, insert customer 4 in R_2 and R_3. Try every possible position in those routes, R_2: 0-4-1-5-0 or 0-1-4-5-0 or 0-1-5-4-0 and R_3: 0-4-0. Feasibility of the solution is checked first. Due to the capacity limit, customer 4 can not be in R_2. Therefore, it should be in R_3.

- Update the overall routes cost for the re-insertion of customer 4 into R_3. This is shown as Table 4.22 below.

TABLE 4.22: Insertion cost of customer 4 into R_3

Vehicle	Route	Demand	Travelling Cost (A)	Tardiness Cost (B)	Route Costs $(\alpha)(A)+(1-\alpha)(B)$
1	0-2-3-0	20	74.80	0	37.40
2	0-1-5-0	38	68	38	53
3	0-4-0	19	50	0	25
Obj.Solution:					115.40

- From Table 4.22, the solution x' yields 115.40 cost unit.

- Compared to the incumbent solution, x, it can be seen that x' is a better solution. Thus, $x \leftarrow x'$. Since a better improvement is found, therefore searching continues with the same number of *Remove* customer. Otherwise, *Remove* is increased by 1.

4.8.3 TS

For the small example here, we need to set the parameter for maximum tabu list, say, $TabuListMax = 2$. In this algorithms, two move operators, i.e. relocate and swap are used. Any move which has been in the tabu list is prohibited. Every possible re-insertion for each neighborhood not violating the capacity constraint is considered and the best solution found will be chosen.

Referring to the Table 4.18, the following moves are made in neighborhood 1 and neighborhood 2.

- Table 4.23 shows the relocation of customer 1, 2, 3, 4 and 5 into possible route.

  Possibilities of moves in neighborhood 1 (relocate) are:

TABLE 4.23: Moves in Neighborhood 1

RelocateCust	Routes	Demand	Travelling Cost	Tardiness Cost
1	R_1:0-2-4-3-0	39	85.60	0
	R_2:0-5-0	28	41.20	0
	R_3:0-1-0	10	30.40	0
Obj.Solution:$(\alpha)(TravelCost) + (1-\alpha)(TardinessCost)$				**78.60**
2	R_1:0-4-3-0	32	72.40	0
	R_2:0-1-5-0	38	68	68.40
	R_3:0-2-0	7	36	0
Obj.Solution:$(\alpha)(TravelCost) + (1-\alpha)(TardinessCost)$				122.40
3	R_1:0-2-4-0	26	63.20	0
	R_2:0-1-5-0	38	68	68.40
	R_3:0-3-0	13	44.80	0
Obj.Solution:$(\alpha)(TravelCost) + (1-\alpha)(TardinessCost)$				122.20
4	R_1:0-2-3-0	20	74.80	0
	R_2:0-5-0	28	41.20	0
	R_3:0-4-0	19	50	0
Obj.Solution:$(\alpha)(TravelCost) + (1-\alpha)(TardinessCost)$				115.40
5	R_1:0-2-4-3-0	39	85.60	0
	R_2:0-1-0	10	30.4	0
	R_3:0-5-0	28	41.20	0
Obj.Solution:$(\alpha)(TravelCost) + (1-\alpha)(TardinessCost)$				115.40

From Table 4.23, re-insertion of customer can only be done in R_3 due to the capacity limit. It can be seen that relocating customer 1 into R_3 is cheaper compared to the others. Thus, the best solution found in neighborhood 1 is 78.60 unit cost.

- Table 4.24 shows a swapping move of each customer in R_1 and R_2. Since this case only involves two routes, thus, moving shown are only between R_1 and R_2. No new route is constructed.

  Possibilities of moves in neighborhood 2 (swap) are:

  From Table 4.24, it can be seen that swapping customer 4 and 5 is more beneficial. Thus, the best solution found in neighborhood 2 is 121.20 unit cost.

- The best solution in each neighborhood is then compared. For this case, relocating customer 1 into the new route, R_3 generating the least-cost solution, x'. Thus, the move is done.

- The solution x' is compared to the incumbent solution, x, in which $x \leftarrow x'$.

- The move, customer 1 from R_2 into R_3 is inserted into tabu list. So that, for the next iteration, relocating customer 1 into R_2 is not allowed.

Table 4.24: Moves in Neighborhood 2

Swap Cust. in		Routes	Demand	Travelling Cost (A)	Tardiness Cost (B)	Objective Solution $(\alpha)(A)+(1-\alpha)(B)$
R_1	R_2					
2	1	R_1:0-1-4-3-0	42	94.80	140.80	149.05
		R_2:0-2-5-0	35	62.50	0	
	5	R_1:0-5-4-3-0	60	109.20	113.20	188.50
		R_2:0-1-2-0	17	65.80	88.80	
4	1	R_1:0-2-1-3-0	30	87.60	181.80	206.20
		R_2:0-4-5-0	47	86.80	56.20	
	5	R_1:0-2-5-3-0	48	107	22.60	**121.20**
		R_2:0-1-4-0	29	72.40	40.40	
3	1	R_1:0-2-4-1-0	36	85.60	144.60	157.95
		R_2:0-3-5-0	41	85.70	0	
	5	R_1:0-2-4-5-0	54	100	97.60	151.30
		R_2:0-1-3-0	23	52.20	52.80	

- Searching is continue in the neighborhood of x', $N(x')$.

4.9 Computational Results

All methods used here are coded in C++ and run on a PC computer with an Intel PentiumCore, 2.66Ghz processor with 16.0GB of RAM. The benchmark VRPTW of 56's Solomon (1987) instances with some modifications (explained earlier) are used as a platform for testing the algorithms. The problems are categorized into six classes, namely C1, C2, R1, R2, RC1 and RC2. Problems set with C categories are clustered data, meaning that locations of customer are geographically close to each others. Problem with R categories are uniformly distributed data, and those from RC classes are hybrid problems that have features of both C and R categories. In addition, C1, R1, and RC1 problems sets have narrower time between release date and due date, whereas the other problem sets have wider time between the dates.

We then compare the findings and draw some conclusions. Note that for both heuristics, i.e. VNS and LNS, which use random numbers in the algorithm, the final solution is calculated as an average of ten solutions. In other words, each of these algorithms is run 10 times and an average of these ten solutions was taken as the solution to the algorithm. To have a fairer comparison, the iterations will stop searching after 30s of CPU computing time and the current best solution obtained is the solution to the method. It is understood that each method is develop on varying nature, so the duration of the test could vary. It means that having the best solution cost among others but taken much longer time is not the indicator to conclude a particular method is the best. Therefore we use the same stopping condition in all three heuristics.

The solutions found in each algorithm have been shown in the following Table 4.25.

TABLE 4.25: A snap comparison of the heuristics.

Big Fleet Size: $a = 0.3$									
Class	Focus on Tardiness, $\alpha = 0.1$			Focus on Equally, $\alpha = 0.5$			Focus on Travelling, $\alpha = 0.9$		
	VNS	LNS	TS	VNS	LNS	TS	VNS	LNS	TS
C1	581.09	587.26	572.87	1493.05	1484.63	1436.04	1846.23	1846.40	1897.75
C2	107.63	108.19	109.98	539.40	539.68	523.52	956.93	961.40	953.79
R1	418.26	413.45	407.93	1001.70	1006.34	1036.68	1398.49	1387.93	1395.39
R2	189.35	186.32	195.06	806.66	799.23	826.28	1272.81	1272.97	1290.91
RC1	442.88	447.62	446.73	1249.09	1253.26	1272.63	1597.93	1606.71	1635.91
RC2	368.49	369.45	375.94	1257.42	1261.28	1285.04	1833.07	1847.33	1907.96
Medium Fleet Size: $a = 0.5$									
Class	Focus on Tardiness, $\alpha = 0.1$			Focus on Equally, $\alpha = 0.5$			Focus on Travelling, $\alpha = 0.9$		
	VNS	LNS	TS	VNS	LNS	TS	VNS	LNS	TS
C1	2185.00	2201.17	2265.07	2309.91	2310.95	2330.99	1912.01	1921.68	1918.90
C2	108.26	108.48	107.58	539.31	541.68	529.88	951.01	957.11	974.09
R1	1485.68	1497.27	1479.78	1431.19	1436.15	1439.08	1391.34	1401.08	1397.73
R2	371.16	373.75	371.82	873.04	882.44	885.05	1265.23	1275.28	1270.98
RC1	1366.80	1367.79	1374.72	1508.49	1544.69	1549.96	1584.38	1602.04	1615.85
RC2	1082.31	1087.83	1106.36	1535.63	1572.54	1570.12	1837.25	1844.29	1837.64
Small Fleet Size: $a = 0.7$									
Class	Focus on Tardiness, $\alpha = 0.1$			Focus on Equally, $\alpha = 0.5$			Focus on Travelling, $\alpha = 0.9$		
	VNS	LNS	TS	VNS	LNS	TS	VNS	LNS	TS
C1	5347.30	5290.05	5411.78	3975.90	3986.07	4036.76	2142.91	2141.38	2146.42
C2	106.75	106.75	109.98	539.25	539.37	558.94	954.67	958.85	960.03
R1	2822.64	2798.82	2859.09	2223.69	2249.90	2325.22	1447.89	1437.73	1464.98
R2	833.24	819.46	843.15	1163.27	1146.93	1160.85	1303.51	1301.17	1303.20
RC1	2933.30	2945.85	2980.39	2127.50	2146.84	2145.31	1602.23	1590.91	1618.86
RC2	2259.27	2299.29	2336.51	2271.60	2231.06	2248.07	1848.26	1858.99	1868.65

Table 4.25 is a summary of Appendix F. The table shows the solution for three different fleet sizes; i.e., big, medium and small fleets each of which focuses on different cost solution in the objective function, i.e. focus on tardiness, travelling and both costs equally.

For the case of big fleet, the number of available vehicles is more than needed. As mentioned in our problem description before, minimizing the vehicle number is not the aim in this case. In fact, we try to maximize the usage of available vehicles through the fleet size-cases that had been designed which might improve the solution cost. From Table 4.25, it can be seen that the smaller size of fleet contributes to higher costs. This can be seen in all type of vehicle routing cases.

As expected, the problem sets with the wider time between dates generate the least cost than the sets with the narrower time. For example, in the small fleet case using heuristic VNS, the average cost 5347.30 and 106.75 are obtained for C1 and C2 respectively. The same situation happens in other heuristics and problem classes. However, different circumstances occur in the problem sets of RC1 and RC2. RC1 generates less cost than RC2 when the problem focuses on travelling cost and as well as both costs equally.

For our experiment, it is empirically found that solutions generated in the big fleet case produce the least costs. This can be seen in all type of weights, α, in which focusing on tardiness in the objective function yields the best solution. However, unlike in other two fleet cases i.e. medium and small-fleet sizes, focusing on travelling, i.e. $\alpha = 0.9$, generates the least cost to each of the problem classes compared to other weights. Logically, by fully utilizing the resources without incurring more vehicle cost might improve the solution cost, especially in respect to tardiness. However, with limited vehicles we have, focusing on distance might be the best strategy. Overall, solution in class C2 does not show any significant differences even though different fleet sizes have been used. This is due to the nature of instances which has been classified as clustered and having wider time windows.

Table 4.26 gives the clear picture of the performance of each algorithm. The relative percentage deviation was calculated based on the average solution of all classes as in the previous Table 4.25.

TABLE 4.26: Comparison of Algorithms (deviation %)

Fleet-Size	Weight, α	VNS	LNS	TS
Big, $a = 0.3$	0.1	0.00	0.22	0.04
	0.5	0.05	0.00	0.56
	0.9	0.00	0.19	1.98
Medium, $a = 0.5$	0.1	0.00	0.56	1.61
	0.5	0.00	1.11	1.31
	0.9	0.00	0.67	0.83
Small, $a = 0.7$	0.1	0.30	0.00	1.97
	0.5	0.01	0.00	1.42
	0.9	0.11	0.00	0.79
Average Dev.		**0.05**	0.31	1.17

Table 4.26 shows an average of percentage deviation on the average cost solution for all six classes. To enable comparison on fair basis, each heuristic is limited to the same processing time. Clearly it can be seen that VNS is the most effective heuristic, solving the problems with 100 customers to the near optima and achieving the least average solution cost in 5 out of 9 solutions. It is followed by LNS which gives the least cost in the rest of 4 solutions. Hence, VNS is the choice to be adopted in the next problem involving coordination of production scheduling and vehicle routing.

4.10 Summary

In this chapter, we present the description of each heuristic used in our study. Based on traditional basic algorithms, we investigate the parameters used in each method. Experiments to determine the best parameter for each algorithm used were done on six instances due to the time constraint. To draw some general conclusion of the algorithm performance, these six instances are chosen to represent the six different classes. By using the parameter settings found, the algorithms were run on all the 56 instances of the modified Solomon's data. Detail solutions found are reported in Appendix F. Finally, we present the performance of the methods used in which VNS comes out as the best performer and hence the one which will be considered in our next problem which deals with the coordination of production scheduling and vehicle routing.

Chapter 5

Coordination of Production Sequence and VRPRDD (PS-VRPRDD)

5.1 Introduction

This chapter presents a study on a problem of production and distribution planning. The problem considers a make-to-order production-distribution system with one manufacturer and many customers. The manufacturer (at the depot) produces the products once customer orders have been received. Distributions to the customers associated with a set of orders should be done before the customer's due date to avoid any penalty for delay. The problem is extended from the problem in the previous chapter where deliveries start depending on the fix release dates. To achieve optimum efficiency in planning, it makes sense to coordinate production schedule with distribution planning. This is the subject of this chapter. Two approaches have been employed in the study; the classical approach which treats these two planning sequentially, and the coordinated approach which integrates these two subproblems as one. VNS is used as optimization method since it shows to be the most competent heuristic in the previous problem.

5.2 Production Sequence and Vehicle Routing

Motivated by the make-to-order-business trend nowadays, the study considers the production-distribution system which consists of one manufacturer and one or more customers. The supplying company produces the products once orders are received. The distribution process starts as soon as the orders are available and can be released.

In this way, inventory, maintenance and warehousing which contribute significantly to the total of the supply management flow could be reduced or avoided altogether.

Boudia *et al* [16] reported that vehicle routing problems have been widely investigated, even for stochastic cases, with inventory constraints and under varied conditions. But the production decisions prior to distribution were usually ignored. In other studies, production planning has been discussed in detail but distribution is not taken into account.

At the beginning of planning, each customer places an order with the manufacturer. Next, the manufacturer processes the orders and attempts to distribute the goods to the customers. Each order has a specified date by which the customer expects to receive his/her order. However, orders may be delivered beyond their due dates due to variability in production schedule which may not meet the due date. Preferably, orders are delivered shortly after their completion. This facilitates the delivery process and prevents delays. On the other hand, this may increase the distribution cost. Hence, the manufacturer will try to consolidate the order delivery as much as possible to minimize the total distribution cost. This means that some completed orders may have to wait for other orders to be completed so that they can be delivered in the same vehicle. This results in a tradeoff between delivery timeliness and total distribution cost. Our study will focus on finding the joint schedule of order processing and delivery to optimize the tradeoff of the delivery timeliness and distribution cost.

Pundoor and Chen [109], stated that the maximum tardiness and total tardiness of orders are the two commonly used measurements of delivery timeliness which represent the worst and average service levels with respect to meeting the order's due dates, respectively. Scheduling problems with optimized maximum tardiness had been studied extensively in machine scheduling in which only production operation was considered. As such a problem deals only with the production part, there exists an optimal solution by scheduling the jobs in nondecreasing order of their due date on a single machine. In addition, some related studies also considered other elements such as jobs release date, due date and batch set-up time [102].

Since our study focuses on the integration of order processing and delivery schedule, we do not consider the problem at the detailed scheduling level. However, we may consider batching together a set of orders for delivery to reduce the total distribution cost. Potts and Kovalyov [105], in their review, stated that it may be cheaper and faster to process jobs in a batch than process them individually; which in our case is related to order delivery.

In this study, the initial release dates of the jobs are assumed to be given. Therefore, processing time of each order, p can be calculated by:

$$p_{s+1} = r_{s+1} - r_s \tag{5.1}$$

where $s = 1, 2, .., N-1$ represents the sequence of jobs processing.

To have a clear picture of this, the following table shows our data input for the problem studied.

Table 5.1: VRPRDD with six customers

customer/order i	**size of order,** q_i	**release date** r_i	**due date** d_i
1	60	15	20
2	20	17	25
3	50	5	20
4	100	45	40
5	40	9	15
6	80	25	35

Table 5.1 above shows a small size problem ; i.e. six customers in a system, for a VRPRDD study. Each customer/order i is associated with the size of order, q_i, release date r_i and due date, d_i. An initial sequence can be determined through the specified release date, r_i. From the given information, the jobs can be ordered as follows.

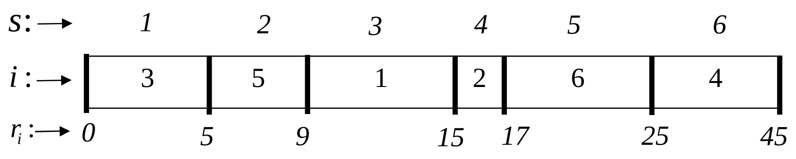


Figure 5.1: An initial sequence of small problem size of VRPRDD in a single production line.

From the Figure 5.1 above, i is the processed job's number while s is the sequence of jobs processing. The given (initial) release dates lead us to the initial sequence of that six jobs processing in a production line. Job 3 associated with customer 3 will be produced first and this is followed by jobs 5, 1, 2, 6 and 4 associated with customers 5, 1, 2, 6, and 4 respectively. From the generated sequence found, processing time of each job i can be determined by Equation 5.1. It can be seen that processing times of jobs 3, 5, 1, 2, 6 and 4 are 5, 4, 6, 2, 8 and 20 units respectively.

Completed orders will be delivered shortly to the customers by K available vehicles which have same capacity limit, Q. The vehicle capacity limit refers to the maximum total size of the jobs that can be delivered. It is also assumed that each vehicle is only using one route where it starts and ends at the depot, v_0. Each customer is only served once and only by one vehicle.

Although a released order can be immediately loading into vehicle, one job might have to wait for others that has been scheduled in the same vehicle due to varied release dates. Therefore, the vehicle can leave the depot only after all orders scheduled on the route are available. Hence, there are possibilities of some late deliveries for which penalties are imposed. The penalty is a cost which can represent the cost of lost sales, or goodwill due to the customer inconvenience for not meeting the customer's due date. In this study, a linear loss function will be used, i.e. a penalty per unit time of tardy delivery.

In this chapter, we are looking at the cost of integration between production scheduling and vehicle routing. We have dealt with the vehicle routing with release and due date (VRPRDD) in the previous chapter. For that problem, it is assumed that release dates are fixed which corresponds to the fixed schedule of order processing. VRPRDD is solved by three metaheuristics for which VNS is found to be the best among them. We will attempt to improve the solution found by adjusting the release dates in relation to their order processing sequence which corresponds to the production sequence/schedule.

5.3 The Model

The mathematical formulation for the VRPRDD has been presented in Chapter 3. However to improve the solution found, it is worth to coordinate the production sequence/schedule with the distribution planning. In other words, we are looking at PS-VRPRDD.

As mentioned in the description of VRPRDD, each job is ordered by a specific customer and it has to be delivered directly to that customer. As PS-VRPRDD is an extension from the previous VRPRDD, the same initial solution will be used; i.e., parallel insertion method based on the fixed production sequence given. There are a total of N jobs which correspond to N customers. Jobs that are scheduled in the same vehicle will be considered as a batch. For instance, say the route for vehicle 1 is 0-2-6-3-0 which means the delivery starts from the depot, then customers 2, 6 and lastly to customer 3 before going back to the depot. Jobs numbers 2, 6 and 3 which corresponds to the customers 2, 6 and 3 will be batched as B_1. We note that q_i corresponds to the size of the job i. Jobs and batches are numbered according to customer's orders and vehicle number respectively. Batches will be used for the process of finding the best sequence in a production line.

Each job selected must be processed one after another continuously from its start in the production line to its completion, without any interruption between jobs, and without any waiting between the finish time of job $i-1$ and starting time of job i. Associated with each job is the processing time and its due date for delivery specified by a customer.

Thus, the main task here is to deal with the problem of selecting and scheduling the orders to be processed by a manufacturer for immediate delivery to the customers. It is important to determine the best order of processing for departure time of vehicle through the generated release dates. The mathematical formulation is same as described in VRPRDD. However, in PS-VRPRDD the processing time of each customer's order is assumed to be known before routing process which can be calculated as shown in previous Figure 5.1. Thus, each customer's vertex is associated with:

i) a known order's size to be delivered, q_i ($q_0 = 0$ for vertex v_0).

ii) a processing time of each customer's order, P_i.

iii) a uniform service time for servicing at customer's location, s_i.

iv) a due date for each customer, d_i.

Due to the production sequence that has been accounted in PS-VRPRDD, the positions of jobs (customer orders) are very important to determine the best release date for each customer. Without loss of generality, we assume that the vehicles are indexed so that the jobs of vehicle k appear as a batch of job sequenced in position k of the production schedule. Thus, the formulation of PS-VRPRDD is given as below:

Mathematical Notations:

n number of customers

$N = \{1, 2, ..., n\}$ the set of n customers

$\overline{N} = \{0, 1, 2, ..., n\}$ the set of all nodes including the depot

h number of vehicles

$K = \{1, 2, ..., h\}$ the fleet of vehicles

Decision variables

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

y_j tardiness for customer j

z_{ik} arrival time of vehicle k at customer i

z_{0k} departure time of vehicle k from depot

Parameters

Q capacity of vehicle

q_i order size of customer i where $q_0 = 0$

s_i service time of customer i where $s_0 = 0$

d_i due date of customer i

r_i release date of customer i's order

P_i processing time on the machine to manufacture the order for customer i, where $P_0 = 0$

t_{ij} travel time from customer i to customer j

c_{ij} travel cost from customer i to customer j

w_i tardiness penalty cost of customer i

M large positive integer number

Formulation

$$\min\ (\alpha)\left(\sum_{i\in\overline{N}}\sum_{j\in\overline{N}}\sum_{k\in K} c_{ij}x_{ijk}\right) + (1-\alpha)\left(\sum_{j\in N} w_j y_j\right) \tag{5.2}$$

subject to

$$\sum_{k\in K}\sum_{j\in\overline{N}} x_{ijk} = 1 \quad \forall i \in N \tag{5.3}$$

$$\sum_{j\in N} x_{0jk} \leq 1 \quad \forall k \in K \tag{5.4}$$

$$\sum_{j\in N} x_{0jk} = \sum_{i\in N} x_{i0k} \quad \forall k \in K \tag{5.5}$$

$$\sum_{i\in\overline{N}} x_{ipk} = \sum_{j\in\overline{N}} x_{pjk} \quad \forall p \in N, \forall k \in K \tag{5.6}$$

$$\sum_{i\in N}\sum_{j\in\overline{N}} q_i x_{ijk} \leq Q \quad \forall k \in K \tag{5.7}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ijk} \leq |S| - 1 \quad S \subset N, |S| \geq 2, \forall k \in K \tag{5.8}$$

$$z_{0k} = \sum_{k'=1}^{k}\sum_{i\in\overline{N}}\sum_{j\in N} P_j x_{ijk} \quad \forall k \in K \tag{5.9}$$

$$z_{ik} + s_i + t_{ij} + M\left(x_{ijk} - 1\right) \leq z_{jk} \quad \forall i \in \overline{N}, \forall j \in N, \forall k \in K \tag{5.10}$$

$$z_{jk} + s_j - d_j + M\left(\sum_{i\in\overline{N}} x_{ijk} - 1\right) \leq y_j \quad \forall j \in N, \forall k \in K \tag{5.11}$$

$$x_{ijk} = 0 \quad \forall i = j \in \overline{N}, \forall k \in K \tag{5.12}$$

$$x_{ijk} \in \{0,1\} \quad \forall i,j \in \overline{N}, \forall k \in K \tag{5.13}$$

$$y_j \geq 0 \quad \forall j \in N \tag{5.14}$$

$$z_{ik} \geq 0 \quad \forall i \in \overline{N}, \forall k \in K \tag{5.15}$$

Since the improvement of the solution of this VRPRDD is based on the assumption that the release dates of jobs may vary, this mathematical formulation is the same as given in the previous chapter VRPRDD. Equation (5.2) is the objective function that is to minimize the weighted total costs of travelling and tardiness. The constraint (5.3) means that each customer is only visited once and only by one vehicle. Each vehicle used for routing originates from the depot, as shown in constraint (5.4). Constraint (5.5) states that once a vehicle leaves the depot, it must go back to the depot again. Route continuity is represented by equation (5.6), i.e. if a vehicle enters a customer node, it must exit from that node. Capacity constraint for each vehicle is shown in equation (5.7) where the summation of order sizes of a route must be less than or equal to vehicle capacity limit. The set of constraints in (5.8) has the goal of avoiding subtour.

The constraint connecting the VRPRDD and PS-VRPRDD is written as constraint (5.9), in which departure time of vehicle is summation of all processing time of the order scheduled in vehicles 1 to k. This can be done as we assume that orders scheduled in one vehicle are considered as one batch in production scheduling. Next, arrival time at each customer location is computed in equation (5.10). Finally, the tardiness of each customer is computed in equation (5.11). Tardiness is defined as difference between arrival time of vehicle at customer's location and its due date. Decision variables are shown in (5.12)-(5.15).

5.4 Solution Approaches

There are two approaches used here which will be compared later to see the advantage of coordinating production scheduling and vehicle routing plan. Before we go further to the algorithms, it is necesary to explain the terms used.

- Batch Formation
  A batch is a set of orders/jobs which will be delivered in the same vehicle. Thus, all the orders scheduled on vehicle k are considered as orders belonging to batch k where $k = 1, 2, 3, ..., K$.

- Latest Start Time (LST)
  Latest start time of vehicle k is the latest time where vehicle k should depart from the depot so that the delay in the distribution activity can be minimized. Assume batch 1 contains jobs of customers 5, 2, 7 and 4.

TABLE 5.2: Batch 1: Arrival and Due Dates of Jobs

Job i	Arrival Date	Due Date
5	8	13
2	11	20
7	13	24
4	23	26

Table 5.2 above shows the arrival and due date of each job in batch 1. The lateness of each job can simply be calculated by subtracting the due date from the arrival time of vehicle for each location as in Table 5.3 below.

TABLE 5.3: Batch 1: Lateness of Jobs

Job i	Arrival Date	Due Date	Lateness
5	8	13	-5
2	11	20	-9
7	13	24	-11
4	23	26	-3

Table 5.3 above shows the lateness time, L, of each job. It can be seen clearly that maximum L is -3. It can be concluded that LST of batch 1 is 3 which can be written as $LST_1 = 3$. For this example, if vehicle 1 leaves the depot at time, $t < 3$ the distribution of customer's orders can be done on time (initially start time from depot is equal to 0).

In order to construct a production sequence, σ, order the jobs (in a batch) in non-decreasing LST_k value.

Note that swapping and relocating batches in a production line are done to improve the current solution cost based on the new generated release date which has been embedded in both approaches.

5.4.1 PS-VRPDD1: Alternate Approach

We begin to solve the problems of production schedule and vehicle routing as an integrated problem. The alternate approach decomposes the problem as two subproblems to be tackled one after the other and merge the resulting solutions into an overall solution to the integrated problem. The flow of the algorithm is given in Figure 5.2. Briefly, the algorithm should improve the production sequence order without changing the route order and vice versa. There exists some integration, where improvement on production schedule is done based on the current best routing. The same goes for routing schedule improvement which is based on the current best sequence of production line. The improvement is done separately but depending on the current best schedule.

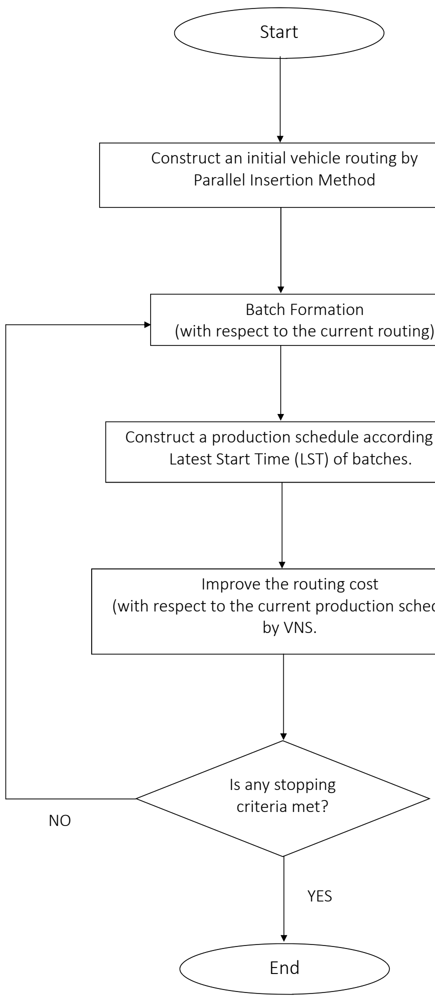

Start
Construct an initial vehicle routing by Parallel Insertion Method
Batch Formation (with respect to the current routing)
Construct a production schedule according to Latest Start Time (LST) of batches.
Improve the routing cost (with respect to the current production schedule) by VNS.
Is any stopping criteria met?
NO
YES
End

FIGURE 5.2: Framework of the Alternate Version.

Figure 5.2 describes the interactive solution framework 1 used to solve the integrated problem. From the best initial routing found, we start the batch formation.

The process of re-batching, re-scheduling and re-routing is continued until stopping condition is met. Improvement process is continued if it shows better results than the current best solution. If there is no improvement, the procedure is terminated. The algorithm is shown as follow:

The initial production scheduling is obtained through the Latest Start Time (LST) of vehicle. As previously mentioned, a production sequence, σ is constructed through the order of the jobs (in a batch) in non-decreasing LST_k value. Please refer to Figure 5.1 for an example. Sets of customer orders or batches will be sequenced according to the non-decreasing order of LST. In other words, the batch with the smallest LST will be produced first to enable it to be delivered before its due date. The sequence of customers in the route is considered as the sequence of those customer's processing

Algorithm 5 Algorithm of PS-VRPRDD1: Alternate Version

```
1: Construct an initial routing by Parallel Insertion Heuristic.
2: for $loop = 1$ to $MaxLoop$ do
3:     Batch Formation: Consider a set of customer's order in the same shipment as
       a batch.
4:     for $Vehicle = 1$ to $MaxVehicle$ do
5:         for $CustNo = 1$ to $MaxCust\,[Vehicle]$ do
6:             $Batch\,[Vehicle]\,[CustNo] \longleftarrow Route\,[Vehicle]\,[CustNo]$.
7:         end for
8:     end for
9:     To construct a production schedule:

         i) Calculate the Latest Start Time (LST) of each vehicle.

         ii) Sort LST in non-decerasing order i.e. set of customer's order with smallest
             LST will be in 1st batch on production line and so forth. Generate the
             new generated release date.

         iii) Then, re-calculate the new routing cost based on the current release date
             which is based on the obtained production schedule.

         iv) Improve the production schedule by swapping and insert/delete among and
             between batches. Re-calculate the release date and new routing cost.

10:    Improve the routing cost found above by VNS with the current release date.
11:    if met stopping condition then
12:        STOP the iteration.
13:    end if
14: end for
```

order in each particular batch. Therefore, these sequences will be considered for our initial production scheduling. The new sequence of orders in production line determines the new release date of each order. Then the new routing cost is calculated based on the new generated release dates. According to the new routing cost found, production scheduling is improved by using swapping, and, inserting and deleting operators between batches.

Next, we attempt to improve the routing cost by VNS which has previously been proven as an effective algorithm for solving VRPRDD. Different release dates may affect the departure time of each vehicle causing changes to the cost of the objective function. Any update of the vehicle departure time and delivery orders creates a potential for improving the quality of the solution cost through re-routing them without increasing the total transportation cost.

However, it is noted here that even if these two subproblems are solved to optimality, it does not guarantee an overall optimum. Thus, we investigate the second approach.

5.4.2 PS-VRPRDD2: InOneMove Approach

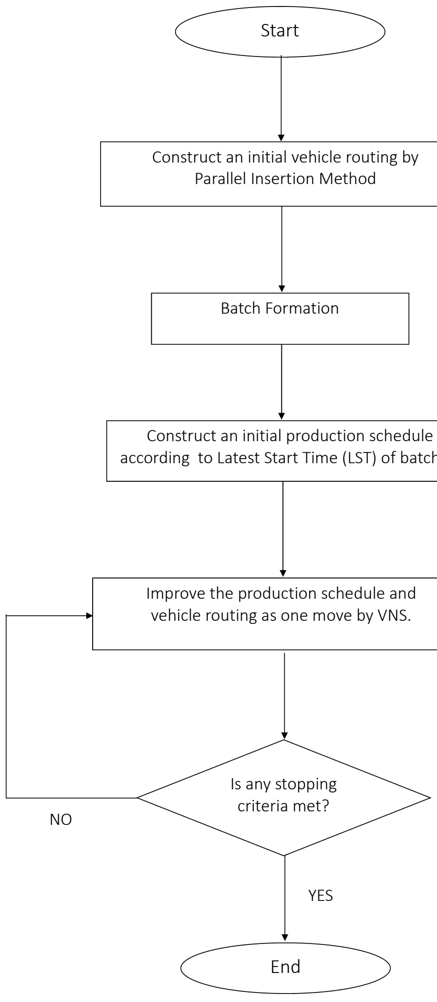

Start
Construct an initial vehicle routing by Parallel Insertion Method
Batch Formation
Construct an initial production schedule according to Latest Start Time (LST) of batches.
Improve the production schedule and vehicle routing as one move by VNS.
Is any stopping criteria met?
NO
YES
End

FIGURE 5.3: Framework of InOneMove Version.

Figure 5.3 describes the interactive solution framework 2 to solve our integrated problem. The solution has a similar framework as solution 1. However, instead of using sequentially approach, solution 2 framework coordinates between job production and delivery. Decision on job production and delivery is done simultaneously as one move. This means that once a job sequence is updated, it must be followed by checking the routing as well. A VNS algorithm has been used to tackle this integrated problem as a whole.

The solution starts with the initial routing which creates the batches. By using the LST, the best initial production schedule is obtained. We then re-calculate the new release dates based on this best sequence of the production schedule. It is important to note here that improving on the best sequence was done through the swapping and the insert/delete operators. Once a move has been made, the release dates will be updated and the total cost of the routes is recalculated.

Improvement on both production scheduling and vehicle routing is done concurrently by VNS. In order to find better overall solution, customers may be swapped or relocated in both routing and production schedule. Therefore, the release date is continuously updated, affecting the departure time of the vehicles and the cost of the routes.

Algorithm 6 below shows the integration of production sequence and vehicle routing in which improvement is done simultaneously.

Algorithm 6 Algorithm of PS-VRPRDD2: InOneMove Version

```
1:  Construct an initial routing by Parallel Insertion Heuristic.
2:  Batch Formation: Consider a set of customer's order in the same shipment as a
    batch.
3:  **for** $Vehicle = 1$ to $MaxVehicle$ **do**
4:      **for** $CustNo = 1$ to $MaxCust\,[Vehicle]$ **do**
5:          $Batch\,[Vehicle]\,[CustNo] \longleftarrow Route\,[Vehicle]\,[CustNo]$.
6:      **end for**
7:  **end for**
8:  To construct an **initial** production schedule.
     i) Calculate the Latest Start Time (LST) of each vehicle.
     ii) Sort LST in non-decerasing order i.e. set of customer's order with the smallest
         LST will be in 1st batch on production line and so forth. Next, new release
         dates are generated.
     iii) Then, re-calculate the new routing cost based on the current release date which
         based on the obtained production schedule.
     iv) Improve the production schedule by swapping and insert/delete among and
         between batches. Re-calculate the release date and new routing cost.
9:  **for** $loop = 1$ to $MaxLoop$ **do**
10:     Improve the production scheduling and vehicle routing in one move by VNS
        with the current release date.
11:     **if** stopping condition is met **then**
12:         STOP the iteration.
13:     **end if**
14: **end for**
```

The algorithm above shows the integration of the production sequence and the vehicle routing. It can be seen that the initial production sequence comes from the initial routing found by parallel insertion method. Next, the cost is improved through VNS where, in each iteration, the best routing is searched. This is followed by improving the sequence of production as well. The iteration continues until a stopping condition is met.

5.5 A Small Example

In this section, a small example is used to illustrate the proposed algorithms; PS-VRPRDD1: Alternate and PS-VRPRDD2: InOneMove. Let assume Table 5.4 shows an initial solution, x to the problem discussed in which all procedures will start from the same initial solution;

TABLE 5.4: Problem of PS-VRPRDD with 5 customers

Vehicle	Route	Demand	Travelling Cost (A)	Tardiness Cost (B)	Route Costs $(\alpha)(A) + (1-\alpha)(B)$
1	0-2-4-3-0	39	85.6	48.20	66.90
2	0-1-5-0	38	68	35.40	51.70
3	0-0	0	0	0	0
Obj.Solution:					118.60

It is assumed that we have 3 available vehicles but only 2 vehicles are used. The other information used are shown as Tables 5.5 and 5.6.

- Number of customers, $n = 5$
- Vehicle Capacity, $Q = 40$
- Penalty Cost=1
- Service Time=5
- weight scale, $\alpha = 0.5$

TABLE 5.5: Problem of PS-VRPRDD with 5 customers

Cust No.	Demand	Relese Time	Due Time	ProcessingTime
1	10	43	100	13
2	7	32	120	6
3	13	13	80	13
4	19	55	130	11
5	28	19	60	12

TABLE 5.6: Euclidean distance between depot and customers' location

Dist.	0	1	2	3	4	5
0	-	15.2	18	22.4	25	20.6
1	15.2	-	32.6	14.6	32.2	32.2
2	18.0	32.6	-	34.4	20.2	23.9
3	22.4	14.6	34.4	-	25	42.7
4	25.0	32.2	20.2	25	-	41.2
5	20.6	32.2	23.9	42.7	41.2	-

Note: Re-calculation on departure time of vehicles, travelling and tardiness cost were based on equations (3.8), (3.9) and (3.10), respectively. This needs to be done once changes occur in the route. There is local search improvement embedded but it is not shown here. For brevity, the steps described for each algorithm is just for iteration one only.

5.5.1 Alternate

As the algorithm 5 shown before, an initial solution as Table 5.4 is referred.

- Batch Formation:
  We consider a set of customer's order in the same shipment as a batch. Therefore, currently we have 2 batches, $B_1 : 2, 4, 3$ and $B_2 : 1, 5$.
- Construct a production schedule:

1. Calculate the Latest Start Time (LST) for each vehicle. The calculation used is as Section 5.4. Table 5.7 shows the latenss time of each job. It can be seen that LST_1 is 48.20 and LST_2 is 35.40.

TABLE 5.7: Lateness of Jobs

Batch	Job i	Arrival Time	Due Time	Lateness
B_1	2	73	120	-47
	4	98.20	130	-31.80
	3	128.20	80	48.20
B_2	1	58.20	100	-41.80
	5	95.40	60	35.40

2. Sort the LST in non-decreasing order in which the smallest LST will be in the 1st batch of production line. The order of production sequence become: 1 5 2 4 3. Hence, new release date is generated as Table 5.8 below.

TABLE 5.8: The new job's release date

Batch	Job i	Processing Time	Release Date
B_1	1	13	13
	5	12	25
B_2	2	6	31
	4	11	42
	3	13	55

3. Then, re-calculate the new routing cost based on the new release date generated as above. This can be seen as Table 5.9 below.

TABLE 5.9: New calculation of PS-VRPRDD of 5 customers

Vehicle	Route	Demand	Travelling Cost (A)	Tardiness Cost (B)	Route Costs $(\alpha)(A) + (1-\alpha)(B)$
1	0-1-5-0	38	68	17.40	42.70
2	0-2-4-3-0	39	85.6	48.20	66.90
3	0-0	0	0	0	0
Obj.Solution:					109.60

4. Then, improve the production schedule by swapping or relocate between batches.

   (a) Since, there are only two batches here, there is only one possibility where swapping the batches or relocate the batch may gives the same result as Figure 5.4.

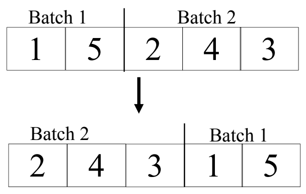


FIGURE 5.4: Swapping batch 1 and 2 in a production sequence.

   (b) Based on the new sequence above, then, new release date is calculated for each job. This is shown in Table 5.10 below.

   (c) Based on the new release date calculated above, we update the route costs as shown in Table 5.11 below:

TABLE 5.10: The new job's release date after batch swapping

Batch	Job i	Processing Time	Release Date
B_2	2	6	6
	4	11	17
	3	13	30
B_1	1	13	43
	5	12	55

TABLE 5.11: New calculation after batch swapping

Vehicle	Route	Demand	Travelling Cost (A)	Tardiness Cost (B)	Route Costs $(\alpha)(A)+(1-\alpha)(B)$
1	0-2-4-3-0	39	85.6	23.20	45.60
2	0-1-5-0	38	68	47.40	66.50
3	0-0	0	0	0	0
Obj.Solution:					112.10

5. Based on the current routing we have now in Table 5.11 and their respective release dates, VNS is used to update the routing cost until a stopping condition is met.

- Based on the best new routing found through VNS, the procedure starts again by forming a batch. This is repeated until the stopping condition, i.e. maximum number of iteration is met.

5.5.2 InOneMove

The same initial solution, x, shown in Table 5.4 is adopted.

- Batch Formation:
  We consider a set of customer's order in the same shipment as a batch. Therefore, currently we have 2 batches, $B_1 : 2, 4, 3$ and $B_2 : 1, 5$.

- Construct a production schedule:
  The steps used is exactly as in Alternate version. Refer to Table 5.7 to 5.11 for the calculation and result.

- Based on the current routing we have now as Table 5.11 and their respective release date, VNS is used to improve the production scheduling and vehicle routing simultaneously with the current release dates. We use the latest result in Table 5.11 to illustrate the process of updating the routing and sequence simultaneously.

1. Shaking: Randomly choose a customer and a route to reinsert. Say, customer 3 and R_3. This customer 3 will be inserted into R_3. This means that, the production sequence becomes as shown in Figure 5.5.

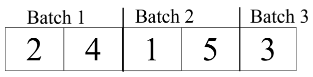


FIGURE 5.5: The production sequence of jobs.

Based on the new sequence of job processing, then, new release date is obtained as shown in Table 5.12 below.

TABLE 5.12: The new job's release date after batch swapping

Batch	Job i	Processing Time	Release Date
B_1	2	6	6
	4	11	17
B_2	1	13	30
	5	12	42
B_1	3	13	55

2. Shaking Solution: New routing cost is calculated based on this new release date is shown as Table 5.13.

TABLE 5.13: Updating new routing cost

Vehicle	Route	Demand	Travelling Cost (A)	Tardiness Cost (B)	Route Costs $(\alpha)(A) + (1 - \alpha)(B)$
1	0-2-4-0	26	63.20	0	31.60
2	0-1-5-0	38	68	34.4	51.20
3	0-3-0	13	44.8	0	22.40
Obj.Solution:x'					105.20

3. Local search is embedded in the algorithm but not shown in this example. Comparing the results, it can be seen that x' is better than incumbent solution, x. Hence, $x \leftarrow x'$. Searching is continued in the 1st neighborhood.

- Searching process continues until stopping condition is met.

5.6 Computational Results

The performance of the Alternate and the InOneMove algorithms has been tested using our modification of 56's Solomon instances as mentioned in the previous chapters. Again, 30 s of CPU time is used as our stopping condition of the iterations. The best solution obtained within the time is considered as the solution for the method. The experiment is done on three fleet-size cases with three different weights each. Consequently 504 cases are tested.

For each case, the algorithm is run for 10 times, and the average of the solution is taken. The full results can be seen in Appendix G, summarized in Table 5.14 below.

TABLE 5.14: Comparison of the two approaches.

Weight	Classes	Big Fleet $a = 0.3$		Medium Fleet $a = 0.5$		Small Fleet $a = 0.7$	
		Alternate	InOneMove	Alternate	InOneMove	Alternate	InOneMove
Focusing on Tardiness, $\alpha = 0.1$	C1	404.15	310.36	1358.16	813.89	3496.53	2132.48
	C2	103.30	146.89	102.72	141.95	103.35	108.25
	R1	286.08	210.11	771.94	484.09	1609.61	1065.19
	R2	206.98	184.22	406.45	332.97	776.17	545.53
	RC1	270.01	255.08	651.23	421.60	1337.95	840.67
	RC2	287.49	259.06	715.95	457.66	1399.97	888.58
Focusing on Both Costs, $\alpha = 0.5$	C1	1253.84	1175.93	1649.81	1297.77	2637.92	2001.37
	C2	513.66	530.83	509.81	528.93	510.60	528.79
	R1	886.95	838.30	989.78	835.17	1369.37	1094.99
	R2	786.82	791.03	857.33	823.32	1061.78	920.22
	RC1	993.17	1019.52	1049.06	910.04	1300.20	1081.36
	RC2	1122.84	1157.03	1240.81	1069.79	1625.17	1278.57
Focusing on Travelling, $\alpha = 0.9$	C1	1570.98	1527.24	1556.10	1479.40	1642.09	1519.41
	C2	905.10	950.04	899.97	951.19	901.19	949.12
	R1	1171.06	1053.78	1168.88	1066.57	1141.01	1084.24
	R2	1184.90	1196.50	1169.39	1115.04	1168.77	1096.31
	RC1	1243.46	1235.69	1241.59	1202.88	1241.25	1196.80
	RC2	1485.47	1463.49	1479.39	1448.48	1443.98	1361.74

Table 5.14 shows the average solution cost of each class of problem. The solutions are obtained by the two proposed algorithms. The average costs obtained indicates the performance of each integrated approach on the different fleet-size cases, weights and instance classes.

It can be seen clearly that all classes in a big fleet case which focus on tardiness cost produce the least average cost solution compared to the other cases and weights. It is proven that having a sufficiently large number of vehicles gives more opportunities in generating better solutions. Waiting time for other scheduled jobs in the same vehicle can be minimized as a job can be delivered to its destination immediately after it is completed so that services beyond the due dates might be avoided as well.

It is observed that for those instances of class C2 for all fleet-size cases and weights, both approaches do not show noteworthy effects of the fleet-size. For instance, focusing on tardiness cost by setting, $\alpha = 0.1$, the average solution cost for both approaches across the different fleet-sizes is between 100 and 150. The same goes to cases for $\alpha = 0.5$, and $\alpha = 0.9$, showing ranges between 500 and 530, and, 900 and 950, respectively. However, it also shows that the average cost is increasing when the weight is increased as well. In other words, for all fleet-size cases, it seems that focusing on the tardiness generates better instances of C2 compared to other weights. This is due to the characteristics of the instances of C2 which deal with clustered customers with wider time windows.

In contrast, average cost generated through the instances of C1 has shown the opposite effect. Different fleet-sizes strongly affect generated solutions for instances C1. The table shows that for big-fleet case, it is best if management focuses on tardiness cost rather than traveling due to a sufficiently large number of available vehicles which can be fully utilized to reduce the cost. On the other hand, having a small number of available vehicles, focusing on travelling cost becomes a good strategy. This is true for both algorithms.

For the instances with customers in random positions, i.e. R1, R2, RC1 and RC2, the average cost solution shows the same conclusion for weight 0.1 and 0.5, across all fleet-sizes. It is clear that the cost increases corresponding to decreasing numbers of available vehicles. However, the contrary situation happens for the problem that focuses on the traveling cost. In this case, the solution produced is slightly reduced with a decreasing number of available vehicles.

In general, the InOneMove shows to be a more effective approach when compared to the Alternate one. InOneMove achieves less average cost for almost all instances. Table 5.15 depicts the performance of both algorithms as shown by the relative average percentage deviations.

TABLE 5.15: Relative percentage deviation of two appraoches.

Weights.	Classes	Big-Fleet Size $\alpha = 0.3$			Medium-Fleet Size $\alpha = 0.5$			Small-Fleet Size $\alpha = 0.7$		
		Alternate	InOneMove	%Deviation.	Alternate	InOneMove	%Deviation.	Alternate	InOneMove	%Deviation.
Focusing on Tardiness, $\alpha = 0.1$	C1	404.15	310.36	-23.21	1358.16	813.89	-40.07	3496.53	2132.48	-39.01
	C2	103.30	146.89	42.20	102.72	141.95	38.19	103.35	108.25	4.75
	R1	286.08	210.11	-26.55	771.94	484.09	-37.29	1609.61	1065.19	-33.82
	R2	206.98	184.22	-10.99	406.45	332.97	-18.08	776.17	545.53	-29.72
	RC1	270.01	255.08	-5.53	651.23	421.60	-35.26	1337.95	840.67	-37.17
	RC2	287.49	259.06	-9.89	715.95	457.66	-36.08	1399.97	888.58	-36.53
	Avg.			**-5.66**			**-21.43**			**-28.58**
Focusing on Both Costs, $\alpha = 0.5$	C1	1253.84	1175.93	-6.21	1649.81	1297.77	-21.34	2637.92	2001.37	-24.13
	C2	513.66	530.83	3.34	509.81	528.93	3.75	510.60	528.79	3.56
	R1	886.95	838.30	-5.48	989.78	835.17	-15.62	1369.37	1094.99	-20.04
	R2	786.82	791.03	0.54	857.33	823.32	-3.97	1061.78	920.22	-13.33
	RC1	993.17	1019.52	2.65	1049.06	910.04	-13.25	1300.20	1081.36	-16.83
	RC2	1122.84	1157.03	3.04	1240.81	1069.79	-13.78	1625.17	1278.57	-21.33
	Avg.			**-0.35**			**-10.70**			**-15.35**
Focusing on Travelling, $\alpha = 0.9$	C1	1570.98	1527.24	-2.78	1556.10	1479.40	-4.93	1642.09	1519.41	-7.47
	C2	905.10	950.04	4.96	899.97	951.19	5.69	901.19	949.12	5.32
	R1	1171.06	1053.78	-10.01	1168.88	1066.57	-8.75	1141.01	1084.24	-4.98
	R2	1184.90	1196.50	0.98	1169.39	1115.04	-4.65	1168.77	1096.31	-6.20
	RC1	1243.46	1235.69	-0.62	1241.59	1202.88	-3.12	1241.25	1196.80	-3.58
	RC2	1485.47	1463.49	-1.48	1479.39	1448.48	-2.09	1443.98	1361.74	-5.70
	Avg.			**-1.49**			**-2.97**			**-3.77**

Table 5.15 demonstrates the significance of the proposed approaches through their relative percentage deviation. Since there is no benchmark to compare with, the calculated deviation is based on the solution generated by Alternate algorithm. It is noted that Alternate is the classical sequential approach which decomposes a problem into two subproblems; production scheduling and vehicle routing treated one after another. Therefore, it is worth to see the performance of InOneMove algorithm by comparing it with the classical approach.

Generally, the negative percentage deviations show that InOneMove algorithm generates lower costs than Alternate in all instance classes except for class C2. When focusing in tardiness cost, Alternate algorithm shows about 5% to 42% reduction against InOneMove algorithm with the biggest difference found in big-fleet case. However, InOneMove algorithm generates around 40% better costs compared with Alternate in class C2 in the case of $a = 0.5$ and $\alpha = 0.1$.

For many instances, InOneMove algorithm which treats the production scheduling and vehicle routing as a whole displays a significant potential for performance improvements. Numerical experiments reveal that InOneMove algorithm yields average relative improvements of between 0.4% and 40%. Overall it can be seen that the reduction in total operating cost from coordination ranges from 0.3% to 29%.

5.7 Summary

Two proposed integrated approaches are explained and applied to three fleet-size, and three different weights each with 56 instances. Full results are presented in Appendix G. The results are summarized in Tables 5.14 and 5.15. Comparison between these two approaches shows that the average total cost obtained by the InOneMove approach is lower than Alternate approach on all runs except for class C2. This shows that integrating production schedule and routing vehicle is significantly better compared to the classical approach.

Chapter 6

Conclusion and Future Work

The chapter provides an overall conclusion of the findings presented in this work and also gives an outline of some research avenues which are worthwhile investigating in the future.

6.1 Conclusion

In classical VRP, it is assumed that all vehicles depart from depot at the time when all orders are available to be dispatched. In this study, orders are available at different points of time due to the variation in manufacturing process. Hence, a vehicle need to wait until all orders scheduled in the vehicle are ready. However, waiting for one order to be done may delay the delivery time of other orders. When a customer has set the latest time for goods received as to be, penalty cost maybe imposed when delays occur. In order to avoid any lateness cost, a vehicle needs to depart as soon as all orders scheduled in their vehicles are released from the production line. Therefore, departure time of a vehicle is dependent on the release date of customer's order.

It is clear that release dates causes changes to the problem, directly affecting the distribution/routing planning. Hence, we need to find the best schedule for production and routing. Therefore, coordinating these two stages is very important.

Research on integration of production scheduling and vehicle routing has been reviewed extensively. However those studies generally focus more on production scheduling level than vehicle routing (e.g. Pundor and Chen [109]). Relatively few researchers focus more on vehicle routing than production scheduling. Our study deals with single production line and focus more on vehicle routing.

The study is conducted in two stages. The aim is to minimize the total weighted cost of travelling and tardiness. In the first stage, the contribution made to the vehicle routing problem which involves the customer's order release dates and their due dates (VRPRDD). As mentioned, the departure time of a vehicle is dependent on the release dates of customers' orders that are scheduled for each vehicle. In this problem, each customer's order release date is determined from a sequence of order processing in a single machine production. It is assumed that the orders are processed one by one sequentially. Hence, no preemption is allowed. Once the order has been completed, it will be released from the line and be made available for distribution. If the delivery is made right after each order has been released, it contributes to higher distribution cost, while tardiness cost might be avoided. To minimize the distribution cost, several orders need to be batched together and deliver as one trip in a vehicle. On the other hand, the tardiness cost may increase if a vehicle needs to wait for other orders that are scheduled in the same route to be available. At this stage, it is assumed that release dates of each customer's order are fixed and given. The investigation of the VRPRDD has been explored in Chapter 4. Further investigation on VRPRDD has been carried out by different approaches. Since there is no benchmark to compare with, we need to use several approaches in order to see the performances of the proposed algorithms.

Prior to analysis in Chapter 4, we have generated the problem instances in Chapter 3. Chapter 3 presents the detail description of the problem, where the mathematical formulation, designation of fleet-size cases and modification of the benchmark problem are given. To the best of our knowledge, there are no benchmark instances for vehicle routing problem with release and due date. Therefore, for the problem discussed, we modified the Solomon benchmark VRPTW instances with 100 customers. The instances contain six different classes which can be categorized as clustered (C), randomized (R) and combination of both customer's position (RC). The modifications are based on the timing attributes of the problem. The fleet-size cases which are big-,medium- and small-fleet size are designed to see the impact of approaches on different kinds of routing problems.

The modified instances have been used to test the proposed algorithms. In this study we use metaheuristic algorithms rather than exact solution methods. The reason is, for large and multi-faceted optimization problems, the large number of variables and constraints in models makes it untenable to exact mathematical methods such as linear, and especially, integer programming. Of course, such methods do (if successful) lead to the true optimum result given the input values. In this study, heuristic approaches such as variable neighbourhood search, large neighbourhood search and tabu search are used. The results obtained may not be optimum, but are still reasonably good.

Initial feasible solution for VRPRDD has been generated by parallel insertion method. Parallel insertions will start with K available vehicles which are stationed at the depot. The value of K will determine the fleet-size. Assuming that the initial release date are given, 100 customers are routed through these K available vehicles. Parallel insertions will route the customers one at a time with these K vehicles. The best vehicle/route which is the best for an inserted customer is determined through the least cost insertion. The process is continued until all 100 customers are routed. The solution then is improved by a local search procedure. This improved feasible initial solution will be used as initial routing for our proposed algorithms, namely VNS, LNS and TS as described in detail in sections 4.5, 4.6 and 4.7 respectively. In each section, various tests have been carried out in order to determine the best parameter for each algorithm. The tests were done on different fleet-sizes and weighted objectives. For the sake of time, the tests for each parameter were done on six instances; C1, C2, R1, R2, RC1 and RC2 which represent clustered, randomized and combination of both customer position with narrow and wide time windows (sign 1 and 2). The different instances were used to draw a general conclusion. The full results are in Appendices A-E and have been summarized in tables in Chapter 4.

Once the parameters for each algorithms have been set according to the best solution found, then computational results are produced for all 56 instances through 3 different fleet cases and weighted objectives totaling 504 vehicle routing problems. In VNS and LNS, random numbers have been used in the searching process. Each instance is run 10 times to get the average solution. The numerical results of these three proposed algorithms; VNS, LNS and TS are then compared. For fair comparisons, the same stopping condition is used. The iteration stops after 30s of CPU time. Analysis is done based on relative percentage deviation. The numerical results showed that the average percentage deviation of VNS is the least compared to others. Therefore, VNS is adopted in our next problem which is our second stage of study, given Chapter 5.

Extending from VRPRDD, we are interested in the next contribution where we could control the release dates. As discussed, release date is generated from the manufacturing process, i.e. the completion time of order's processing. Therefore, apart of routing, we are looking at the production schedule as well. The fact is release dates do changes to the problem nature. For example, even if we have customers in a clustered locations, routing planning with the 'wrong' release dates will not work.

Hence, our second stage is related to a problem with production and distribution planning. In order to improve the solution obtained in VRPRDD, an adjustment needs to be done at the stage of production planning. Such an adjustment to the sequence of one line production planning will generate different sets of release dates. Coordination between production planning and vehicle routing is the main feature of Chapter 5, known as PS-VRPRDD. Two approaches have been proposed; the classical decomposition which treats these two sub-problems, production planning and vehicle routing sequentially, and the other which integrates both planning as a whole. The approaches have been named as Alternate and InOneMove, respectively. VNS has been used as the optimization method for both approaches since it shows the most promising results for VRPRDD.

Generally, the Alternate method dealt with the production planning and vehicle routing separately. Once the best vehicle routing is found, then by using the routes obtained, an adjustment to the sequence in production line is done until it improves the current weighted objective functions. Next, once the best sequence of order processing is determined, searching for the best routing by considering the new release date is made. The searching process for both sub-problems is continued until the stopping condition is met. This classic decomposition approach that successively deals with the associated sub-problems are often found to produce poorly coordinated overall results.

In contrast, InOneMove algorithm tackles the integrated problem as a whole is compared to the Alternate algorithm. The same 504 test cases have been run using these two algorithms. Numerical results show that the Alternate approach, which breaks down the overall problem, successively solves the sub-problems, and aggregates the solutions to the sub-problems, but yields relatively poor results compared to InOneMove algorithm. On average, for most instances, the InOneMove algorithm which coordinates the two decisions as one show a significant performance improvement. The results showed the improvement can reach to 28.58% using InOneMove as compared to Alternate.

6.2 Suggestions for Future Research

For the most part, this study has pursued the aim of minimizing the costs of travelling and tardiness. However,there are many more matters that need to be considered in dealing with realistic problems. For example, we have focused our discussion on a single production machine, a single-plant and a single-product scenarios. Thus, a possible future study is to consider multiple plants or machines that potentially produce a set of different products requiring higher levels of integration among production and routing planning.

Moreover, in some problem, it is probably more meaningful to consider an integrated schedule for multiple-periods. In this case, the algorithms should be capable to track demands of each customer over several planning periods.

While the test results that we obtained showed a significant potential cost savings, it would still be interesting to study different heuristics or even exact solution methods such as branch and bound which had been used as solver for commercial optimization software that we used, i.e., AIMMS. This exact solution can be used for the model validation at least for smaller instances.

An enhancement can be incorporated into our approach which might improve the solution quality and making it more efficient by varying execution times. As has been mentioned earlier, all algorithms used the same stopping condition that is allowing maximum 30s of CPU time to search for the optimal solution. It is possible that better final solution may appear with longer computation time. Therefore, it is worthwhile to investigate the solution quality by varying the maximum searching time in phases.

It is important to note that the basic objective of our computational study was to compare heuristics in terms of the quality of solutions they generated. As such, no special effort was made to reduce either the number of vehicles or running times of the algorithm to a minimum. We observed that it is sensible to evaluate the performance of heuristics by comparing the ratios of solution quality and time.

6.3 Final Remarks

Overall, we believe that the work done in this thesis contributes positively to scientific research in vehicle routing and scheduling. This study provides simple idea on constraint handling mechanism that can be used in designing effective and robust algorithms. The methods approached in this research can be easily integrated within larger frameworks which can be used as an optimization tool to help improve distribution planning.

Bibliography

[1] Ahmed, Z. H. The ordered clustered travelling salesman problem: A hybrid genetic algorithm. *The Scientific World Journal 2014* (2014).

[2] Altinel, I., and Oncan, T. A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society* (2005), 954–961.

[3] Altinkemer, K., and Gavish, B. Parallel savings based heuristics for the delivery problem. *Operations Research* (1991), 456–469.

[4] Andres Figliozzi, M. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review 48*, 3 (2012), 616–636.

[5] Archetti, C., Jabali, O., and Speranza, M. G. Multi-period vehicle routing problem with due dates.

[6] Archetti, C., and Speranza, M. Vehicle routing problems with split deliveries. *International transactions in operational research 19*, 1-2 (2012), 3–22.

[7] Archetti, C., Speranza, M. G., and Hertz, A. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science 40*, 1 (2006), 64–73.

[8] Asim, M., Gopalia, R., and Swar, S. Travelling salesman problem using genetic algorithm.

[9] Balakrishnan, N. Simple heuristics for the vehicle routeing problem with soft time windows. *Journal of the Operational Research Society 44*, 3 (1993), 279–287.

[10] Baldacci, R., Mingozzi, A., and Roberti, R. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research 218*, 1 (2012), 1–6.

[11] Beasley, J. Route-first cluster-second methods for vehicle routing. *Omega 11*, 4 (1983), 403–408.

[12] Bent, R., and Van Hentenryck, P. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science 38*, 4 (2004), 515–530.

[13] Bertazzi, L., and Speranza, M. G. Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics 1*, 4 (2012), 307–326.

[14] Blumenfeld, D., Burns, L., and Daganzo, C. Synchronizing production and transportation schedules. *Transportation Research Part B: Methodological 25*, 1 (1991), 23–37.

[15] Bodin, L., Golden, B., Assad, A., and Ball, M. Routing and scheduling of vehicles and crews: The state of the art. *COMP. & OPER. RES. 10*, 2 (1983), 63–211.

[16] Boudia, M., Louly, M., and Prins, C. Fast heuristics for a combined production planning and vehicle routing problem. *Production Planning and Control 19*, 2 (2008), 85–96.

[17] Bräysy, O., and Gendreau, M. Tabu Search Heuristics for Vehicle Routing Problem with Time Windows. *Sociedad de Estadistica e Investigación Operativa Top 10*, 2 (2002), 211–237.

[18] Bräysy, O., and Gendreau, M. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science 39*, 1 (2005), 104–118.

[19] Bräysy, O., and Gendreau, M. Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science 39*, 1 (2005), 119–139.

[20] Burke, E., and Kendall, G. *Search methodologies: introductory tutorials in optimization and decision support techniques.* Springer Verlag, 2005.

[21] Cakici, E., Mason, S., and Kurz, M. Multi-objective analysis of an integrated supply chain scheduling problem.

[22] Campbell, A., and Savelsbergh, M. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation science 38*, 3 (2004), 369–378.

[23] Chang, Y., and Lee, C. Machine scheduling with job delivery coordination. *European Journal of Operational Research 158*, 2 (2004), 470–487.

[24] Chen, H., Hsueh, C., and Chang, M. Production scheduling and vehicle routing with time windows for perishable food products. *Computers and Operations Research 36*, 7 (2009), 2311–2319.

[25] Chen, J. C., H. W. H. C. C., and Chen, C. Hybrid meta-heuristics for vehicle routing problem with time window constraints. *Unpublished* (2009).

[26] Chen, J., Chen, C., Chen, B., Lay, M., Liu, L., Lee, B., Huang, S., Chang, W., and Huang, D. Application of Vehicle Routing Problem with Hard Time Window Constraints. In *The 29th International Conference Computers and Industrial Engineering* (2000).

[27] Chen, Z. Integrated production and outbound distribution scheduling: Review and extensions. *Operations Research 58* (2010), 130–148.

[28] Chen, Z., and Vairaktarakis, G. Integrated scheduling of production and distribution operations. *Management Science* (2005), 614–628.

[29] Chen, Z.-L. Integrated production and distribution operations. In *Handbook of quantitative supply chain analysis*. Springer, 2004, pp. 711–745.

[30] Cheng, C. B., and Wang, K. P. Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm. *Expert Systems with Applications 36* (2009), 7758–7763.

[31] Chiang, W., Russell, R., Xu, X., and Zepeda, D. A simulation/metaheuristic approach to newspaper production and distribution supply chain problems. *International Journal of Production Economics 121*, 2 (2009), 752–767.

[32] Christofides, N., M. A., and Toth, P. *Combinatorial Optimization*. Wiley, Chichester, UK, 1979, ch. The vehicle routing problem, pp. 315–338.

[33] Clarke, G., and Wright, J. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* (1964), 568–581.

[34] Cóccola, M., Zamarripa, M., Méndez, C. A., and Espuña, A. Toward integrated production and distribution management in multi-echelon supply chains. *Computers & Chemical Engineering 57* (2013), 78–94.

[35] Cordeau, J., and Laporte, G. Tabu search heuristics for the vehicle routing problem. *Metaheuristic Optimization via Memory and Evolution* (2005), 145–163.

[36] Dantzig, G., Fulkerson, R., and Johnson, S. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America 2*, 4 (1954), 393–410.

[37] Dantzig, G., and Ramser, J. The truck dispatching problem. *Management Science* (1959), 80–91.

[38] Dawande, M., Geismar, H., Hall, N., and Sriskandarajah, C. Supply chain scheduling: distribution systems. *Production and Operations Management 15*, 2 (2006), 243–261.

[39] Derigs, U., and Vogel, U. Experience with a framework for developing heuristics for solving rich vehicle routing problems. *Journal of Heuristics 20*, 1 (2014), 75–106.

[40] Desrochers, M., Desrosiers, J., and Solomon, M. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research 40*, 2 (1992), 342–354.

[41] Dror, M., and Trudeau, P. Savings by split delivery routing. *Transportation Science* (1989), 141–145.

[42] Dror, M., and Trudeau, P. Split delivery routing. *Naval Research Logistics (NRL) 37*, 3 (1990), 383–402.

[43] Ehmke, J. F., Campbell, A. M., and Urban, T. L. Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research 240*, 2 (2015), 539–550.

[44] Eksioglu, B., Vural, A., and Reisman, A. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering 57*, 4 (2009), 1472–1483.

[45] Ertogral, K., Wu, S., and Burke, L. Integrating Production and Transportation Logistics in a Supply Chain Environment: A Lagrangean Decomposition Approach. *IMSE Technical Reprot 98T 10*.

[46] Fahimnia, B., Luong, L., Marian, R., et al. Optimization/simulation modeling of the integrated production-distribution plan: an innovative survey. *WSEAS TRANSACTIONS on Business and Economics 3*, 5 (2008).

[47] Farahani, P., Grunow, M., and Gunther, H. A heuristic approach for short-term operations planning in a catering company. In *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on* (2010), IEEE, pp. 1131–1135.

[48] Farahani, P., Grunow, M., and Günther, H.-O. Integrated production and distribution planning for perishable food products. *Flexible services and manufacturing journal 24*, 1 (2012), 28–51.

[49] Fisher, M. Vehicle routing. *Handbooks in operations research and management science 8* (1995), 1–33.

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

[51] Francis, P. M., Smilowitz, K. R., and Tzur, M. The period vehicle routing problem and its extensions. In *The vehicle routing problem: latest advances and new challenges*. Springer, 2008, pp. 73–102.

[52] Garcia, J., and Lozano, S. Production and delivery scheduling problem with time windows. *Computers & Industrial Engineering 48*, 4 (2005), 733–742.

[53] Gendreau, M., Hertz, A., Laporte, G., and Stan, M. A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research 46*, 3 (1998), 330–335.

[54] Gendreau, M., Iori, M., Laporte, G., and Martello, S. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks 51*, 1 (2008), 4–18.

[55] Gillett, B., and Miller, L. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research* (1974), 340–349.

[56] Glover, F., and Laguna, M. Tabu search. 1997.

[57] Goel, A., and Gruhn, V. Large neighborhood search for rich vrp with multiple pickup and delivery locations. In *Proceedings of the 18th Mini EURO Conference on VNS (MEC-VNS). ISBN* (2005), pp. 84–689.

[58] Goel, A., and Gruhn, V. Solving a dynamic real-life vehicle routing problem. *Operations Research Proceedings 2005* (2006), 367–372.

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

[60] Goksal, F. P., Karaoglan, I., and Altiparmak, F. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering 65*, 1 (2013), 39–53.

[61] Graves, S. A review of production scheduling. *Operations Research* (1981), 646–675.

[62] Hansen, P., and Mladenović, N. Variable neighborhood search: Principles and applications. *European journal of operational research 130*, 3 (2001), 449–467.

[63] Helsgaun, K. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research 126*, 1 (2000), 106–130.

[64] Hemmelmayr, V., Doerner, K., and Hartl, R. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research 195*, 3 (2009), 791–802.

[65] Hinton, T. G. *A Thesis regarding the vehicle routing problem including a range of novel techniques for its solution.* PhD thesis, Department of Computer Science, University of Bristol, September 2010.

[66] Hong, L. An improved lns algorithm for real-time vehicle routing problem with time windows. *Computers & Operations Research 39*, 2 (2012), 151–163.

[67] Hsu, C., Hung, S., and Li, H. Vehicle routing problem with time-windows for perishable food delivery. *Journal of Food Engineering 80*, 2 (2007), 465–475.

[68] Ioannou, G., Kritikos, M., and Prastacos, G. A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society* (2001), 523–537.

[69] Kanda, A., Deshmukh, S., et al. Supply chain coordination: Perspectives, empirical studies and research directions. *International Journal of Production Economics 115*, 2 (2008), 316–335.

[70] Kang, K. H., L. B. K. L. Y. H., and Lee, Y. H. A heuristic for the vehicle routing poblem with due times. *Computers & Industrial engineering 54* (2008), 421–431.

[71] Kim, B. I., K. S., and Sahoo, S. Waste collection vehicle route problem with time window. *Computers & Operational Research 33* (2006), 3624–3642.

[72] Kumar, S. N., and Panneerselvam, R. A survey on the vehicle routing problem and its variants. *Intelligent Information Management 4* (2012), 66.

[73] Laporte, G. Location routing problems. In *Vehicle Routing: Methods and Studies*, A. A. Golden, B.L., Ed. North-Holland, Amsterdam, 1987, pp. 163–198.

[74] Laporte, G., and Semet, F. Classical heuristics for the capacitated vrp. *The vehicle routing problem* (2002), 109–128.

[75] Larsen, J. *Parallelization of the vehicle routing problem with time windows.* Institute of Mathematical Modelling, Technical University of Denmark, 1999.

[76] Lee, H., Cha, S., Yu, Y., and Jo, G. Large neighborhood search using constraint satisfaction techniques in vehicle routing problem. *Advances in Artificial Intelligence* (2009), 229–232.

[77] Lee, T., and Ueng, J. A study of vehicle routing problems with load-balancing. *International Journal of Physical Distribution and Logistics Management 29* (1999), 646–658.

[78] Lee, Y., Kim, J., Kang, K., and Kim, K. A heuristic for vehicle fleet mix problem using tabu search and set partitioning. *Journal of the Operational Research Society 59*, 6 (2007), 833–841.

[79] Lee, Y., and Kim, S. Production-distribution planning in supply chain considering capacity constraints. *Computers & Industrial Engineering 43*, 1-2 (2002), 169–190.

[80] Lei, H., Laporte, G., and Guo, B. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research 38*, 12 (2011), 1775–1783.

[81] Li, C.-L., and Vairaktarakis, G. Coordinating production and distribution of jobs with bundling operations. *IIE transactions 39*, 2 (2007), 203–215.

[82] Li, K., Ganesan, V., and Sivakumar, A. Synchronized scheduling of assembly and multi-destination air-transportation in a consumer electronics supply chain. *International journal of production research 43*, 13 (2005), 2671–2685.

[83] Lim, A., and Zhang, X. A Two-Stage Heuristic with Ejection Pools and Generalized Ejection Chains for the Vehicle-Routing Problem with Time Windows. *INFORMS Journal on Computing 19*, 3 (2007), 443–457.

[84] Lin, S., et al. Computer solutions of the traveling salesman problem. *Bell System Technical Journal 44*, 10 (1965), 2245–2269.

[85] Lin, S., and Kernighan, B. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research* (1973), 498–516.

[86] Liu, F., and Shen, S. A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. *European Journal of Operational Research 118*, 3 (1999), 485–504.

[87] Liu, F., and Shen, S. An overview of a heuristic for vehicle routing problem with time windows. *Computers & industrial engineering 37*, 1-2 (1999), 331–334.

[88] Lopez, P., and Roubellat, F. *Production scheduling.* Wiley Online Library, 2008.

[89] Marchetti, P. A., Gupta, V., Grossmann, I. E., Cook, L., Valton, P.-M., Singh, T., Li, T., and André, J. Simultaneous production and distribution of industrial gas supply-chains. *Computers & Chemical Engineering 69* (2014), 39–58.

[90] Marinakis, Y., and Migdalas, A. Heuristic solutions of vehicle routing problems in supply chain management. *SERIES ON APPLIED MATHEMATICS-WORLD SCIENTIFIC PUBLISHING COMPANY- 14* (2002), 205–236.

[91] Min, H. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General 23*, 5 (1989), 377–386.

[92] Min, H. A multiobjective vehicle routing problem with soft time windows: the case of a public library distribution system. *Socio-Economic Planning Sciences 25*, 3 (1991), 179–188.

[93] Mladenović, N., and Hansen, P. Variable neighborhood search. *Computers & Operations Research 24*, 11 (1997), 1097–1100.

[94] Mole, R., and Jameson, S. A sequential route-building algorithm employing a generalised savings criterion. *Operational Research Quarterly* (1976), 503–511.

[95] Nilsson, C. Heuristics for the traveling salesman problem. *Theoretica Computer Science Reports, Linkoping university* (2003).

[96] Osman, I. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research 41*, 4 (1993), 421–451.

[97] Osvald, A., and Stirn, L. A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food. *Journal of Food Engineering 85*, 2 (2008), 285–295.

[98] Paquette, J., Cordeau, J.-F., Laporte, G., and Pascoal, M. Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological 52* (2013), 1–16.

[99] Park, Y. A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. *International Journal of Production Economics 73*, 2 (2001), 175–188.

[100] Park, Y. An integrated approach for production and distribution planning in supply chain management. *International Journal of Production Research 43*, 6 (2005), 1205–1224.

[101] Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. A review of dynamic vehicle routing problems. *European Journal of Operational Research 225*, 1 (2013), 1–11.

[102] Pinedo, M. *Scheduling: theory, algorithms, and systems.* Springer Verlag, 2008.

[103] Pinedo, M. L. *Scheduling: theory, algorithms, and systems.* Springer Science & Business Media, 2012.

[104] Pisinger, D., and Ropke, S. Large neighborhood search. *Handbook of Metaheuristics* (2010), 399–419.

[105] Potts, C. N., and Kovalyov, M. Y. Scheduling with batching: a review. *European Journal of Operational Research 120*, 2 (2000), 228–249.

[106] Prescott-Gagnon, E., Desaulniers, G., and Rousseau, L. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks 54*, 4 (2009), 190–204.

[107] Prodhon, C., and Prins, C. A survey of recent research on location-routing problems. *European Journal of Operational Research 238*, 1 (2014), 1–17.

[108] Psaraftis, H. N. Dynamic vehicle routing: Status and prospects. *annals of Operations Research 61*, 1 (1995), 143–164.

[109] Pundoor, G., and Chen, Z. Scheduling a production-distribution system to optimize the trade-off between delivery tardiness and distribution cost. Published online in Wiley InterScience, July 2005. DOI 10.1002/nav.20100.

[110] Robert, J., and Claasen, S. A sequential insertion heuristic for the initial solution to a constrained vehicle routing problem. *ORiON 22(1)* (2006), 105–116.

[111] Ropke, S. *Heuristic and exact algorithms for vehicle routing problems.* PhD thesis, Department of Computer Science, University of Copenhagen, December 2005.

[112] Ropke, S., and Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science 40*, 4 (2006), 455–472.

[113] Ropke, S., and Pisinger, D. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research 171*, 3 (2006), 750–775.

[114] Russell, R., Chiang, W., and Zepeda, D. Integrating multi-product production and distribution in newspaper logistics. *Computers & Operations Research 35*, 5 (2008), 1576–1588.

[115] Russell, R., and Urban, T. Vehicle routing with soft time windows and Erlang travel times. *Journal of the Operational Research Society 59*, 9 (2007), 1220–1228.

[116] Savelsbergh, M. A parallel insertion heuristic for vehicle routing with side constraints. *Statistica Neerlandica 44*, 3 (1990), 139–148.

[117] Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics 159*, 2 (2000), 139–171.

[118] Shaw, P. 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* (1997).

[119] Shaw, P. Using constraint programming and local search methods to solve vehicle routing problems. *Principles and Practice of Constraint Programming—CP98* (1998), 417–431.

[120] Solomon, M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* (1987), 254–265.

[121] Sousa, J. C., Sousa, J. C., et al. A multi objective approach to solve capacitated vehicle routing problems with time windows using mixed integer linear programming. *International Journal of Advanced Science and Technology 28* (2011), 1–8.

[122] Stadtler, H. Supply chain management and advanced planning—-basics, overview and challenges. *European journal of operational research 163*, 3 (2005), 575–588.

[123] Stenger, A., Vigo, D., Enz, S., and Schwind, M. An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science 47*, 1 (2013), 64–80.

[124] Tan, K., Lee, L., and Ou, K. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. *Engineering Applications of Artificial Intelligence 14*, 6 (2001), 825–837.

[125] Tan, K., Lee, L., Zhu, Q., and Ou, K. Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering 15*, 3 (2001), 281–295.

[126] Taş, D., Dellaert, N., Van Woensel, T., and De Kok, T. Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research 40*, 1 (2013), 214–224.

[127] Tavakkoli-Moghaddam, R., Gazanfari, M., Alinaghian, M., Salamatbakhsh, A., and Norouzi, N. A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing. *Journal of manufacturing systems 30*, 2 (2011), 83–92.

[128] Teixeira, J., Antunes, A., and de Sousa, J. Recyclable waste collection planning—-a case study. *European Journal of Operational Research 158* (2004), 543–554.

[129] Thomas, D., and Griffin, P. Coordinated supply chain management. *European journal of operational research 94*, 1 (1996), 1–15.

[130] Toth, P., and Vigo, D. *The vehicle routing problem.* Society for Industrial and Applied Mathematics, 2002.

[131] Toth, P., and Vigo, D. *The vehicle routing problem.* Society for Industrial and Applied Mathematics, 2002.

[132] Tsubakitani, S., and Evans, J. Optimizing tabu list size for the traveling salesman problem. *Computers & operations research 25*, 2 (1998), 91–97.

[133] Ullrich, C. A. Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research 227*, 1 (2013), 152–165.

[134] Van Buer, M., Woodruff, D., and Olson, R. Solving the medium newspaper production/distribution problem. *European Journal of Operational Research 115*, 2 (1999), 237–253.

[135] Wang, D. *Integrated Scheduling of Production and Transportation Operations with Stage-Dependent Inventory Costs and Due Dates Considerations*. PhD thesis, Universitè de Technologie de Belfort-Montbéliard (UTBM), 2012.

[136] Wang, X., Xu, C., and Hu, X. Genetic algorithm for vehicle routing problem with time windows and a limited number of vehicles. In *Management Science and Engineering, 2008. ICMSE 2008. 15th Annual Conference Proceedings., International Conference on* (2008), IEEE, pp. 128–133.

[137] Wark, P., and Holt, J. A repeated matching heuristic for the vehicle routeing problem. *Journal of the Operational Research Society* (1994), 1156–1167.

[138] Wy, J., Kim, B.-I., and Kim, S. The rollon–rolloff waste collection vehicle routing problem with time windows. *European Journal of Operational Research 224*, 3 (2013), 466–476.

[139] Xu, J., Yan, F., and Li, S. Vehicle routing optimization with soft time windows in a fuzzy random environment. *Transportation Research Part E: Logistics and Transportation Review 47*, 6 (2011), 1075–1091.

[140] Yu, B., Yang, Z., and Yao, B. A hybrid algorithm for vehicle routing problem with time windows. *Expert Systems with Applications 38*, 1 (2011), 435–441.

Appendix A

Experiments on VNS Shaking Phase

A.1 Big-Fleet Size, a=0.3

Table A.1: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	1103.61	110.43	851.62	802.46	412.13	433.60	618.98
2	1129.01	110.19	848.39	805.21	413.24	437.51	623.92
3	1144.08	109.98	854.68	817.72	412.61	437.82	629.48
4	1125.02	110.36	843.59	811.14	415.67	428.47	622.37
5	1134.38	110.59	851.25	803.28	407.26	438.94	624.28
6	1137.59	110.47	859.09	817.33	412.79	435.61	628.81

Table A.2: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	1987.18	578.09	1287.03	1396.30	1327.49	1606.23	1363.72
2	1980.55	573.51	1285.42	1399.41	1326.19	1593.41	1359.75
3	1986.59	573.24	1287.27	1401.36	1330.77	1605.42	1364.11
4	1984.03	574.32	1284.57	1406.34	1326.50	1591.94	1361.28
5	1986.54	587.73	1287.28	1400.31	1328.35	1608.34	1366.43
6	1994.12	582.14	1286.60	1402.42	1327.94	1611.92	1367.52

TABLE A.3: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	2195.95	958.39	1553.07	1888.39	1710.63	2137.81	1740.71
2	2193.79	958.85	1554.78	1897.23	1720.25	2144.23	1744.85
3	2201.01	961.32	1548.90	1894.00	1716.54	2140.98	1743.79
4	2197.85	962.53	1552.16	1893.87	1713.24	2141.84	1743.58
5	2196.16	962.30	1549.72	1898.59	1721.87	2141.37	1745.00
6	2201.01	960.91	1556.23	1892.24	1725.40	2142.77	1746.43

A.2 Medium-Fleet Size, a=0.5

TABLE A.4: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	4050.80	109.50	2436.92	2726.46	1321.70	2288.60	2155.66
2	4040.52	109.07	2407.58	2799.63	1319.00	2277.37	2158.86
3	4064.92	110.46	2442.73	2802.33	1321.15	2216.38	2159.66
4	4077.81	109.70	2435.91	2804.37	1318.93	2320.25	2177.83
5	4076.56	110.45	2432.07	2785.94	1319.26	2322.77	2174.51
6	4095.14	110.31	2446.65	2805.23	1319.00	2352.62	2188.16

TABLE A.5: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	3400.00	582.05	1958.89	2416.10	1623.84	2208.52	2031.57
2	3395.75	573.07	1944.94	2418.06	1624.83	2205.03	2026.95
3	3446.30	577.94	1962.36	2439.69	1630.86	2204.57	2043.62
4	3448.40	579.32	1962.73	2457.27	1631.22	2199.37	2046.39
5	3433.14	580.07	1966.45	2435.30	1628.72	2220.09	2043.96
6	3414.15	583.97	1954.95	2427.48	1627.26	2219.40	2037.87

TABLE A.6: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	2333.99	958.55	1559.94	1859.40	1705.86	2143.52	1760.21
2	2327.55	961.51	1561.77	1842.23	1723.17	2144.59	1760.14
3	2349.61	962.28	1563.42	1850.61	1721.28	2146.19	1765.57
4	2343.91	960.74	1562.82	1858.46	1718.32	2137.73	1763.67
5	2341.04	962.71	1560.28	1857.67	1724.71	2142.72	1764.85
6	2338.30	959.75	1562.54	1833.55	1710.71	2148.68	1758.92

A.3 Small-Fleet Size, a=0.7

Table A.7: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	8726.81	110.24	4133.15	5285.74	3562.40	4095.94	4319.05
2	8691.35	108.46	4156.15	5210.03	3634.27	4066.50	4311.13
3	8736.10	109.43	4152.59	5302.15	3610.51	4026.88	4322.94
4	8758.09	109.44	4157.43	5241.93	3599.10	4097.34	4327.22
5	8723.93	109.70	4183.89	5320.09	3594.34	3979.36	4318.55
6	8761.63	109.49	4162.30	5271.93	3566.26	4036.63	4318.04

Table A.8: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	5757.14	581.59	2951.60	3796.23	2169.39	3182.34	3073.05
2	5748.61	579.27	2949.28	3793.96	2174.36	3182.96	3071.41
3	5726.53	582.59	2959.86	3807.39	2171.26	3178.50	3071.02
4	5742.22	581.90	2957.56	3806.51	2173.56	3187.46	3074.87
5	5772.62	583.77	2953.08	3790.17	2169.64	3187.46	3076.12
6	5748.02	586.36	2953.09	3810.58	2171.67	3183.51	3075.54

Table A.9: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

k_{max}	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
1	2577.06	959.30	1633.02	2013.00	1720.64	2121.30	1837.38
2	2574.97	959.45	1629.28	2013.20	1729.53	2121.85	1838.05
3	2597.11	962.75	1637.33	2014.05	1736.55	2121.18	1844.83
4	2600.38	960.90	1636.64	2011.99	1721.87	2123.85	1842.61
5	2595.92	962.28	1633.64	1999.91	1733.84	2123.94	1841.59
6	2589.18	961.00	1631.54	2010.31	1731.33	2123.94	1841.22

Appendix B

VND Local Search: Setting 1

B.1 Big-Fleet Size, a=0.3

Table B.1: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	1173.07	110.46	770.90	798.38	415.67	445.02	618.91
1 2 4 3	1048.12	108.96	784.85	759.58	411.64	386.44	583.26
1 3 2 4	1043.67	109.77	769.62	740.49	415.67	445.02	587.37
1 3 4 2	1078.35	110.31	859.09	818.07	415.67	402.83	614.05
1 4 2 3	1117.10	110.46	850.25	818.07	415.67	445.02	626.09
1 4 3 2	1157.67	110.46	819.79	803.77	415.67	445.02	625.40
2 1 3 4	1065.30	108.30	823.77	814.06	415.46	443.10	611.66
2 1 4 3	1066.30	108.10	829.71	818.07	415.67	394.54	605.40
2 3 1 4	990.21	110.18	784.76	818.07	415.67	445.02	593.99
2 3 4 1	1139.53	110.46	853.16	818.07	415.67	445.02	630.32
2 4 1 3	1068.10	110.46	825.59	818.07	415.67	445.02	613.82
2 4 3 1	1046.89	110.46	859.09	818.07	415.67	445.02	615.87
3 1 2 4	1041.68	110.31	817.53	818.07	415.67	445.02	608.05
3 1 4 2	1088.50	108.81	859.09	818.07	415.67	445.02	622.53
3 2 1 4	1040.58	110.78	816.23	818.07	415.67	397.57	599.82
3 2 4 1	1139.65	110.46	846.92	793.97	415.67	435.51	623.70
3 4 1 2	1108.58	110.46	802.79	739.14	415.67	445.02	603.61
3 4 2 1	1064.59	110.46	853.39	807.33	415.67	445.02	616.08
4 1 2 3	1167.99	110.46	806.80	802.27	415.67	445.02	624.70
4 1 3 2	1092.80	110.27	859.09	818.07	401.29	430.75	618.71
4 2 1 3	1059.74	110.46	826.80	805.49	415.67	435.36	608.92
4 2 3 1	1173.07	110.46	859.09	818.07	415.67	435.51	635.31
4 3 1 2	1092.80	108.81	859.09	815.45	415.67	445.02	622.81
4 3 2 1	1129.3	110.362	859.091	793.965	400.352	437.405	621.75

Table B.2: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	1994.12	589.37	1273.69	1400.67	1291.06	1618.61	1361.25
1 2 4 3	1994.12	564.01	1292.53	1384.47	1318.10	1594.16	1357.90
1 3 2 4	1967.23	572.75	1259.05	1406.54	1292.86	1601.26	1349.95
1 3 4 2	1973.17	581.91	1253.15	1391.93	1333.71	1618.96	1358.81
1 4 2 3	1958.59	584.71	1274.42	1406.54	1333.71	1582.43	1356.73
1 4 3 2	1994.12	583.73	1292.94	1379.30	1331.42	1566.49	1358.00
2 1 3 4	1994.12	585.45	1240.33	1406.54	1323.29	1587.55	1356.21
2 1 4 3	1953.50	580.47	1270.13	1400.67	1333.71	1557.37	1349.31
2 3 1 4	1994.12	589.44	1271.81	1392.60	1319.91	1604.86	1362.12
2 3 4 1	1991.43	589.44	1282.38	1406.54	1332.55	1619.23	1370.26
2 4 1 3	1958.44	575.59	1293.24	1401.51	1326.21	1619.23	1362.37
2 4 3 1	1979.27	575.64	1271.44	1393.62	1323.34	1617.71	1360.17
3 1 2 4	1990.42	562.42	1260.94	1405.77	1333.71	1604.86	1359.69
3 1 4 2	1980.43	588.23	1293.24	1405.78	1313.05	1616.69	1366.24
3 2 1 4	1950.48	585.89	1287.49	1363.40	1332.30	1599.20	1353.13
3 2 4 1	1991.13	589.44	1293.24	1406.54	1328.58	1611.72	1370.11
3 4 1 2	1993.01	589.44	1282.22	1392.52	1320.59	1565.14	1357.15
3 4 2 1	1918.86	555.40	1278.98	1402.52	1333.71	1615.82	1350.88
4 1 2 3	1992.11	579.53	1293.24	1400.89	1333.71	1601.71	1366.86
4 1 3 2	1994.12	575.64	1282.22	1406.54	1333.71	1619.23	1368.58
4 2 1 3	1991.43	576.84	1293.24	1400.39	1313.05	1612.03	1364.50
4 2 3 1	1994.12	589.44	1278.78	1389.39	1333.71	1566.34	1358.63
4 3 1 2	1994.12	587.81	1265.00	1406.54	1333.71	1619.23	1367.74
4 3 2 1	1985.81	589.444	1280.16	1406.54	1330.57	1619.23	1368.63

Table B.3: Focusing on Minimising the Travelling Cost, $\alpha = 0.9$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	2195.55	959.14	1557.38	1891.49	1717.56	2146.26	1744.56
1 2 4 3	2201.58	957.61	1525.16	1888.54	1694.26	2146.26	1735.57
1 3 2 4	2201.58	963.25	1548.35	1885.24	1749.89	2137.54	1747.64
1 3 4 2	2201.58	959.98	1557.13	1871.98	1697.38	2146.26	1739.05
1 4 2 3	2185.99	960.78	1554.26	1888.00	1724.45	2146.26	1743.29
1 4 3 2	2201.58	959.05	1555.97	1873.93	1738.53	2146.26	1745.89
2 1 3 4	2188.52	952.87	1547.16	1891.49	1673.60	2146.26	1733.32
2 1 4 3	2194.43	960.61	1557.38	1898.59	1730.02	2146.26	1747.88
2 3 1 4	2201.58	952.24	1555.28	1876.00	1713.57	2141.15	1739.97
2 3 4 1	2201.58	954.70	1552.25	1898.59	1747.57	2136.10	1748.47
2 4 1 3	2192.18	950.75	1546.15	1847.82	1673.76	2121.59	1722.04
2 4 3 1	2201.58	963.25	1557.38	1870.49	1707.02	2120.52	1736.71
3 1 2 4	2186.65	955.49	1546.99	1892.27	1732.58	2146.26	1743.37
3 1 4 2	2198.73	959.05	1543.09	1880.62	1716.82	2146.26	1740.76
3 2 1 4	2195.35	955.49	1537.61	1881.13	1693.75	2110.76	1729.02
3 2 4 1	2201.58	963.25	1538.46	1890.03	1733.00	2146.26	1745.43
3 4 1 2	2186.69	955.28	1557.38	1888.00	1693.47	2146.26	1737.85
3 4 2 1	2192.79	957.84	1557.38	1898.59	1676.48	2134.32	1736.23
4 1 2 3	2199.34	957.84	1557.38	1877.67	1740.82	2121.82	1742.48
4 1 3 2	2201.58	963.25	1554.99	1877.85	1702.89	2146.26	1741.14
4 2 1 3	2201.58	957.84	1555.41	1871.96	1676.37	2126.03	1731.53
4 2 3 1	2178.52	960.11	1543.59	1866.19	1671.47	2138.59	1726.41
4 3 1 2	2201.58	963.25	1557.38	1881.14	1727.94	2134.32	1744.27
4 3 2 1	2198.73	963.249	1531.79	1898.59	1747.57	2146.26	1747.70

B.2 Medium-Fleet Size, a=0.5

TABLE B.4: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	4103.72	110.57	2455.20	2793.94	1316.96	2190.04	2161.74
1 2 4 3	4009.98	110.00	2453.25	2744.31	1305.65	2383.77	2167.83
1 3 2 4	3914.70	108.15	2411.57	2816.57	1322.25	2179.02	2125.38
1 3 4 2	4103.72	110.78	2417.26	2677.60	1310.63	2176.45	2132.74
1 4 2 3	4076.21	110.78	2419.66	2805.02	1322.25	2334.81	2178.12
1 4 3 2	4072.17	109.77	2455.20	2763.95	1307.28	2323.08	2171.91
2 1 3 4	4038.61	110.57	2455.20	2718.98	1315.38	2160.94	2133.28
2 1 4 3	4035.22	108.38	2455.20	2814.29	1300.66	2367.50	2180.21
2 3 1 4	4035.22	109.82	2453.25	2581.87	1320.54	2121.17	2103.64
2 3 4 1	4103.72	110.20	2435.00	2762.05	1322.25	2383.77	2186.17
2 4 1 3	4103.72	108.35	2389.59	2809.75	1322.25	2254.65	2164.72
2 4 3 1	4103.72	110.27	2337.52	2816.57	1322.25	2196.11	2147.74
3 1 2 4	3942.68	110.78	2365.76	2653.98	1316.77	2354.64	2124.10
3 1 4 2	4079.25	110.46	2436.96	2816.57	1322.25	2383.77	2191.54
3 2 1 4	3973.91	110.37	2403.60	2655.13	1288.21	2190.04	2103.54
3 2 4 1	4082.90	110.20	2455.20	2816.57	1322.25	2236.56	2170.61
3 4 1 2	4103.72	108.81	2399.87	2765.10	1311.00	2336.67	2170.86
3 4 2 1	4097.30	110.27	2455.20	2816.57	1322.25	2383.77	2197.56
4 1 2 3	4004.82	110.28	2422.40	2791.78	1282.92	2190.12	2133.72
4 1 3 2	4070.84	109.86	2436.96	2658.58	1322.25	2383.77	2163.71
4 2 1 3	4103.72	110.46	2444.62	2636.75	1322.25	2281.13	2149.82
4 2 3 1	4103.72	108.63	2455.20	2816.57	1322.25	2281.13	2181.25
4 3 1 2	4006.55	110.27	2423.46	2813.07	1322.25	2281.13	2159.45
4 3 2 1	4103.72	110.457	2455.2	2816.57	1322.25	2241.2	2174.90

TABLE B.5: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	3364.56	586.85	1960.36	2399.38	1606.40	2180.02	2016.26
1 2 4 3	3433.91	568.26	1883.45	2463.41	1626.04	2187.77	2027.14
1 3 2 4	3387.12	589.44	1968.03	2383.03	1583.87	2179.69	2015.20
1 3 4 2	3457.71	574.99	1960.25	2344.10	1600.51	2160.76	2016.39
1 4 2 3	3314.64	577.18	1939.54	2352.89	1615.38	2223.21	2003.81
1 4 3 2	3456.98	576.24	1963.59	2444.51	1615.38	2220.84	2046.26
2 1 3 4	3412.23	582.99	1968.03	2362.39	1587.16	2223.21	2022.67
2 1 4 3	3448.31	584.82	1956.47	2416.46	1599.09	2223.21	2038.06
2 3 1 4	3383.95	576.40	1962.98	2463.41	1632.39	2223.21	2040.39
2 3 4 1	3426.70	587.92	1943.39	2463.41	1632.39	2189.59	2040.57
2 4 1 3	3334.90	570.06	1920.48	2384.63	1610.51	2195.17	2002.63
2 4 3 1	3457.71	579.26	1968.03	2461.94	1632.39	2192.65	2048.66
3 1 2 4	3431.17	579.65	1968.03	2361.68	1632.39	2198.24	2028.53
3 1 4 2	3376.08	557.88	1968.03	2433.49	1629.34	2223.21	2031.34
3 2 1 4	3334.90	572.75	1929.95	2354.54	1632.39	2223.21	2007.96
3 2 4 1	3431.52	575.64	1968.03	2463.41	1588.96	2174.11	2033.61
3 4 1 2	3362.73	573.89	1934.94	2463.41	1632.39	2222.25	2031.60
3 4 2 1	3405.37	570.26	1968.03	2411.22	1632.39	2199.30	2031.10
4 1 2 3	3457.71	589.44	1935.20	2426.85	1632.39	2219.05	2043.44
4 1 3 2	3362.73	589.44	1968.03	2462.52	1632.39	2223.21	2039.72
4 2 1 3	3426.70	589.44	1955.00	2278.81	1632.39	2223.21	2017.59
4 2 3 1	3357.26	573.50	1939.68	2341.43	1632.39	2223.21	2011.24
4 3 1 2	3423.85	584.85	1968.03	2443.30	1614.04	2189.59	2037.28
4 3 2 1	3340.01	588.473	1941.31	2423.92	1632.39	2219.05	2024.19

Table B.6: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	2302.47	963.25	1563.63	1871.36	1731.59	2152.36	1764.11
1 2 4 3	2351.20	953.36	1562.48	1874.88	1727.47	2152.36	1770.29
1 3 2 4	2351.20	958.28	1556.46	1820.52	1703.51	2152.36	1757.05
1 3 4 2	2351.20	963.25	1555.08	1824.03	1698.81	2152.36	1757.45
1 4 2 3	2351.20	963.25	1538.92	1846.52	1681.24	2132.94	1752.34
1 4 3 2	2351.20	959.05	1559.89	1874.88	1696.85	2152.36	1765.71
2 1 3 4	2350.05	954.13	1554.94	1870.10	1715.40	2137.91	1763.76
2 1 4 3	2314.51	957.84	1534.31	1866.01	1724.01	2152.36	1758.17
2 3 1 4	2351.20	959.45	1558.76	1866.52	1668.02	2121.12	1754.18
2 3 4 1	2351.05	960.56	1563.63	1817.59	1714.33	2130.36	1756.25
2 4 1 3	2351.20	963.25	1561.84	1865.88	1688.18	2138.85	1761.53
2 4 3 1	2323.82	963.25	1563.63	1874.88	1685.86	2152.36	1760.63
3 1 2 4	2343.49	947.14	1539.80	1851.77	1676.42	2152.36	1751.83
3 1 4 2	2326.93	963.25	1559.29	1874.88	1736.10	2152.36	1768.80
3 2 1 4	2342.18	955.49	1563.63	1825.34	1707.57	2152.35	1757.76
3 2 4 1	2351.20	963.25	1561.41	1874.88	1687.32	2152.36	1765.07
3 4 1 2	2305.42	963.25	1555.52	1874.88	1747.84	2130.36	1762.88
3 4 2 1	2332.64	960.56	1563.63	1863.17	1715.52	2141.38	1762.82
4 1 2 3	2351.20	952.87	1563.63	1862.49	1749.89	2152.36	1772.07
4 1 3 2	2350.05	957.84	1563.63	1854.41	1711.12	2144.83	1763.65
4 2 1 3	2351.05	954.70	1561.24	1874.88	1749.89	2152.14	1773.98
4 2 3 1	2328.65	958.42	1563.63	1874.88	1696.95	2152.36	1762.48
4 3 1 2	2342.38	961.53	1535.91	1852.33	1713.57	2133.90	1756.60
4 3 2 1	2351.20	963.25	1563.63	1825.82	1724.03	2151.27	1763.20

B.3 Small-Fleet Size, a=0.7

TABLE B.7: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	8687.99	110.30	4031.14	5091.96	3545.26	4162.46	4271.52
1 2 4 3	8685.30	108.92	4156.15	5232.32	3377.49	4162.46	4287.11
1 3 2 4	8614.02	110.78	4195.44	5252.65	3454.40	3883.37	4251.78
1 3 4 2	8766.19	108.81	4063.78	5320.09	3565.53	3958.80	4297.20
1 4 2 3	8625.08	110.60	4031.79	5320.09	3499.29	4162.46	4291.55
1 4 3 2	8675.46	110.78	4194.56	5200.61	3481.51	4117.52	4296.74
2 1 3 4	8284.98	110.27	4157.78	5284.05	3588.15	4098.14	4253.89
2 1 4 3	8635.32	109.34	4088.62	5256.58	3546.72	3946.00	4263.76
2 3 1 4	8704.07	110.46	4043.59	5163.14	3504.96	4031.77	4259.66
2 3 4 1	8706.81	108.62	4195.29	5320.09	3534.94	4074.94	4323.45
2 4 1 3	8565.89	110.21	4140.92	5252.95	3518.44	4162.46	4291.81
2 4 3 1	8711.50	109.05	4187.82	5320.09	3652.19	4059.82	4340.08
3 1 2 4	8659.50	110.60	4079.31	5320.09	3506.52	4159.79	4305.97
3 1 4 2	8534.26	110.46	4140.38	5320.09	3546.17	4161.96	4302.22
3 2 1 4	8618.38	110.78	4089.93	5177.82	3550.86	4162.46	4285.04
3 2 4 1	8728.95	110.28	4200.96	5312.18	3608.93	4081.85	4340.53
3 4 1 2	8766.19	108.81	4049.96	5296.06	3594.08	3963.26	4296.39
3 4 2 1	8711.50	108.81	4208.43	5320.09	3594.08	3832.65	4295.93
4 1 2 3	8766.19	110.46	4152.61	5214.96	3588.63	4162.46	4332.55
4 1 3 2	8711.50	110.46	4031.58	5320.09	3562.26	3940.66	4279.42
4 2 1 3	8660.81	110.46	4096.32	5320.09	3679.80	3892.25	4293.29
4 2 3 1	8707.11	110.27	4185.03	5253.80	3594.08	4065.89	4319.36
4 3 1 2	8766.19	110.46	4208.43	5320.09	3645.94	4014.72	4344.30
4 3 2 1	8766.04	110.457	4111.09	5283.96	3608.42	4059.82	4323.30

Table B.8: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	5759.21	586.85	2967.05	3814.62	2182.64	3168.34	3079.79
1 2 4 3	5649.39	588.90	2967.05	3684.17	2163.64	3187.46	3040.10
1 3 2 4	5710.78	583.76	2967.05	3814.62	2135.10	3187.46	3066.46
1 3 4 2	5669.94	578.38	2962.09	3814.62	2181.72	3187.46	3065.70
1 4 2 3	5730.46	584.61	2871.62	3550.13	2182.64	3179.63	3016.51
1 4 3 2	5795.36	571.54	2931.02	3802.13	2160.91	3155.96	3069.49
2 1 3 4	5766.08	588.90	2943.11	3720.25	2163.64	3076.87	3043.14
2 1 4 3	5795.36	583.31	2951.60	3774.45	2181.72	3177.58	3077.34
2 3 1 4	5795.36	587.10	2933.58	3683.97	2161.53	3177.07	3056.44
2 3 4 1	5748.82	589.44	2967.05	3781.73	2161.97	3187.46	3072.75
2 4 1 3	5584.70	589.20	2861.15	3812.07	2131.66	3187.46	3027.71
2 4 3 1	5795.36	589.44	2967.05	3755.59	2182.64	3177.58	3077.94
3 1 2 4	5760.12	569.65	2903.74	3810.37	2163.64	3177.58	3064.18
3 1 4 2	5682.00	578.56	2934.22	3727.54	2182.64	3155.98	3043.49
3 2 1 4	5671.39	574.08	2904.52	3778.84	2110.04	3137.38	3029.37
3 2 4 1	5795.36	577.41	2967.05	3784.65	2161.97	3187.46	3078.98
3 4 1 2	5754.24	581.23	2967.05	3814.62	2105.10	3177.58	3066.64
3 4 2 1	5795.36	588.49	2967.05	3806.57	2182.64	3187.46	3087.93
4 1 2 3	5795.36	579.53	2967.05	3802.79	2161.60	3177.58	3080.65
4 1 3 2	5795.36	586.86	2951.60	3814.62	2164.28	3187.46	3083.36
4 2 1 3	5691.07	557.88	2956.74	3814.62	2182.64	3164.20	3061.19
4 2 3 1	5771.55	585.78	2934.38	3774.45	2182.64	3187.46	3072.71
4 3 1 2	5710.63	575.64	2918.93	3814.62	2182.64	3155.98	3059.74
4 3 2 1	5795.36	575.64	2934.38	3786.37	2182.64	3187.46	3076.98

TABLE B.9: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol.**
1 2 3 4	2598.73	959.97	1631.67	2015.72	1732.51	2123.94	1843.76
1 2 4 3	2584.53	963.25	1637.33	2015.72	1708.47	2103.42	1835.45
1 3 2 4	2614.45	963.25	1631.76	2015.72	1697.65	2123.94	1841.13
1 3 4 2	2602.37	960.56	1626.18	1971.32	1723.49	2123.94	1834.64
1 4 2 3	2601.58	959.84	1637.33	2004.65	1726.42	2123.94	1842.29
1 4 3 2	2606.98	960.56	1626.08	2015.72	1737.13	2123.51	1845.00
2 1 3 4	2584.61	953.36	1628.87	2011.54	1726.43	2123.94	1838.12
2 1 4 3	2577.38	963.25	1637.33	2008.09	1661.98	2123.94	1828.67
2 3 1 4	2595.80	963.25	1637.33	2008.09	1663.81	2116.15	1830.74
2 3 4 1	2615.97	963.25	1637.33	2015.72	1728.74	2123.94	1847.49
2 4 1 3	2613.17	957.61	1637.33	2006.46	1732.51	2113.68	1843.46
2 4 3 1	2609.86	957.84	1625.87	2010.68	1737.13	2113.02	1842.40
3 1 2 4	2578.45	949.63	1636.71	1993.70	1723.01	2123.94	1834.24
3 1 4 2	2609.86	960.11	1637.33	2015.72	1737.13	2118.58	1846.46
3 2 1 4	2615.97	963.25	1636.22	2008.09	1695.89	2118.58	1839.67
3 2 4 1	2599.96	960.56	1601.16	2015.72	1729.78	2123.94	1838.52
3 4 1 2	2580.47	957.84	1633.88	2009.57	1723.12	2123.94	1838.14
3 4 2 1	2572.96	963.25	1619.52	1978.06	1735.88	2123.94	1832.27
4 1 2 3	2615.97	958.42	1631.76	2011.54	1737.13	2113.20	1844.67
4 1 3 2	2604.21	963.25	1625.87	2005.13	1684.04	2123.94	1834.41
4 2 1 3	2581.01	957.84	1626.18	2015.72	1729.78	2123.94	1839.07
4 2 3 1	2615.97	963.25	1637.33	2007.69	1717.03	2123.94	1844.20
4 3 1 2	2564.76	958.34	1637.33	2006.46	1732.59	2111.92	1835.23
4 3 2 1	2602.37	963.249	1637.33	2015.72	1707.81	2113.2	1839.95

Appendix C

VND Local Search: Setting 2

C.1 Best Order in Setting 1 basis

C.1.1 Big-Fleet Size, a=0.3

Table C.1: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	1094.60	109.31	855.58	796.20	415.67	403.39	612.46
3 2 1 5 4	1018.58	110.46	804.42	763.19	412.59	445.02	592.37

Table C.2: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	1984.76	584.82	1280.85	1395.33	1333.71	1591.98	1361.91
3 2 1 5 4	1986.40	588.97	1259.05	1383.61	1327.68	1609.98	1359.28

Table C.3: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	2194.50	948.33	1556.79	1898.59	1709.09	2146.26	1742.26
3 2 1 5 4	2169.44	954.22	1554.46	1874.51	1739.64	2146.26	1739.76

C.1.2 Medium-Fleet Size, a=0.5

Table C.4: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	3914.50	110.08	2440.93	2751.71	1322.25	2383.77	2153.87
3 2 1 5 4	4103.72	109.98	2277.20	2728.70	1282.92	2331.61	2139.02

Table C.5: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	3382.66	577.22	1967.44	2297.12	1618.00	2223.21	2010.94
3 2 1 5 4	3423.53	589.44	1940.66	2389.40	1604.88	2223.21	2028.52

Table C.6: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	2257.46	956.42	1555.97	1865.42	1711.88	2111.28	1743.07
3 2 1 5 4	2342.53	963.25	1552.70	1861.08	1719.24	2152.36	1765.19

C.1.3 Small-Fleet Size, a=0.7

Table C.7: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	8600.33	110.66	4050.03	5193.57	3441.91	4085.39	4246.98
3 2 1 5 4	8616.30	109.37	4043.36	5307.55	3564.17	4078.54	4286.55

Table C.8: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	5747.30	589.44	2910.55	3814.62	2044.83	3187.46	3049.03
3 2 1 5 4	5747.52	587.69	2804.16	3755.59	2093.51	3187.46	3029.32

TABLE C.9: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
3 2 1 4 5	2541.89	957.61	1635.40	2011.54	1737.13	2115.71	1833.21
3 2 1 5 4	2577.68	959.77	1629.28	1960.71	1708.89	2123.94	1826.71

C.2 First Best basis: Identify the neighborhood's value

C.2.1 Big-Fleet Size, a=0.3

TABLE C.10: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	1163.05	110.68	858.76	812.11	415.67	443.20	633.91
2opt	1170.32	110.78	859.09	815.30	415.67	445.02	636.03
3opt	1173.07	109.85	860.48	818.07	416.30	444.32	637.02
RelRD	1168.30	107.08	858.90	818.07	415.67	445.02	635.51
RelCost	1151.95	108.99	839.68	798.50	400.76	450.00	624.98

TABLE C.11: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	1992.30	589.44	1392.56	1406.54	1334.00	1618.33	1388.86
2opt	1994.12	587.00	1293.24	1406.54	1330.42	1619.23	1371.76
3opt	1995.23	589.44	1294.30	1406.54	1333.71	1619.23	1373.08
RelRD	1993.43	583.60	1290.03	1400.89	1330.40	1612.46	1368.47
RelCost	1982.36	576.30	1283.40	1403.10	1330.78	1619.23	1365.86

TABLE C.12: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	2201.83	963.25	1557.38	1889.60	1745.32	2147.55	1750.82
2opt	2200.54	962.88	1556.90	1890.45	1749.89	2142.56	1750.54
3opt	2201.58	963.25	1557.38	1899.36	1750.32	2146.26	1753.02
RelRD	2198.30	957.84	1547.32	1898.59	1749.89	2145.89	1749.64
RelCost	2195.35	963.25	1556.98	1880.63	1724.21	2106.45	1737.81

C.2.2 Medium-Fleet Size, a=0.5

Table C.13: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	4103.72	110.78	2450.55	2816.57	1322.25	2383.77	2197.94
2opt	4102.50	111.50	2455.20	2815.00	1320.45	2388.53	2198.86
3opt	4103.72	110.78	2455.56	2816.57	1322.75	2383.10	2198.75
RelRD	4006.20	110.27	2442.92	2707.22	1315.27	2359.47	2156.89
RelCost	3975.77	110.78	2446.24	2816.57	1322.25	2385.78	2176.23

Table C.14: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	3457.71	589.44	1967.30	2460.05	1634.85	2228.30	2056.28
2opt	3458.52	589.44	1970.35	2463.41	1632.39	2224.87	2056.50
3opt	3457.71	590.20	1968.03	2460.53	1633.48	2223.21	2055.53
RelRD	3458.45	584.02	1963.87	2462.65	1630.45	2224.20	2053.94
RelCost	3382.11	589.44	1943.39	2453.02	1632.39	2219.05	2036.57

Table C.15: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	2350.98	961.14	1563.63	1872.06	1750.33	2153.78	1775.32
2opt	2351.20	963.25	1560.23	1874.88	1745.63	2153.56	1774.79
3opt	2350.45	960.45	1563.63	1875.36	1749.89	2152.88	1775.44
RelRD	2335.69	956.65	1563.63	1843.14	1743.60	2122.24	1760.83
RelCost	2351.20	963.25	1554.74	1848.49	1724.11	2152.36	1765.69

C.2.3 Small-Fleet Size, a=0.7

Table C.16: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	8766.19	110.45	4204.90	5252.60	3668.61	4055.51	4343.04
2opt	8723.91	111.50	4168.30	5328.46	3685.01	4162.46	4363.27
3opt	8777.56	110.78	4200.08	5321.56	3660.03	4162.46	4372.08
RelRD	8716.85	109.85	4208.43	5258.69	3629.96	4050.09	4328.98
RelCost	8642.03	110.78	4201.85	5320.09	3656.08	4163.52	4349.06

Table C.17: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	5734.35	587.98	2967.05	3712.47	2183.45	3188.98	3062.38
2opt	5795.36	588.97	2966.45	3812.45	2182.64	3187.46	3088.89
3opt	5780.23	589.44	2967.05	3814.62	2181.56	3180.90	3085.63
RelRD	5795.36	589.25	2914.48	3777.38	2180.48	3158.50	3069.24
RelCost	5784.63	588.92	2967.12	3788.34	2182.78	3187.46	3083.21

Table C.18: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

VND	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**	**Avg.Sol**
swap	2618.78	959.97	1635.24	2008.22	1740.89	2123.85	1847.83
2opt	2615.97	964.56	1637.33	2014.99	1737.13	2125.66	1849.27
3opt	2614.32	963.25	1636.88	2015.72	1738.21	2123.94	1848.72
RelRD	2602.55	960.48	1637.16	2016.78	1736.32	2124.51	1846.30
RelCost	2609.86	963.25	1635.89	1998.80	1722.79	2106.76	1839.56

C.3 Random basis

C.3.1 Big-Fleet Size, a=0.3

Table C.19: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
Random1	1173.07	110.46	820.42	816.28	404.87	445.02
Random2	1151.03	110.00	823.38	736.12	415.67	442.98
Random3	1173.07	110.33	850.00	773.39	415.67	436.35
Random4	1160.49	110.17	850.40	818.05	415.67	444.17
Random5	1047.98	109.68	859.09	818.07	415.67	419.19
Random6	1040.59	109.64	859.09	807.28	415.67	398.33
Random7	1040.59	109.64	859.09	807.28	415.67	398.33
Random8	1173.07	110.46	808.37	818.07	415.67	445.02
Random9	1128.41	110.46	832.68	726.23	415.67	428.98
Random10	1130.79	110.46	774.47	804.60	415.67	445.02
Average	1121.91	110.13	833.70	792.54	414.59	430.34

Table C.20: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
Random1	1970.65	576.24	1284.76	1391.97	1333.71	1596.35
Random2	1965.83	589.44	1276.12	1383.89	1332.36	1612.03
Random3	1983.82	584.57	1279.15	1406.54	1333.71	1619.23
Random4	1994.12	589.44	1271.44	1406.54	1330.92	1589.04
Random5	1989.16	568.38	1258.48	1385.57	1328.26	1619.23
Random6	1994.12	574.01	1257.83	1379.79	1330.57	1612.78
Random7	1994.12	574.01	1257.83	1379.79	1330.57	1612.78
Random8	1993.01	577.82	1270.13	1396.30	1320.78	1619.23
Random9	1989.16	578.65	1291.28	1397.44	1322.04	1609.53
Random10	1958.33	586.44	1278.98	1400.89	1333.71	1619.23
Average	1983.23	579.90	1272.60	1392.87	1329.66	1610.94

Table C.21: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
Random1	2201.58	963.25	1552.57	1886.77	1713.57	2146.26
Random2	2201.58	963.25	1543.12	1890.92	1749.89	2113.68
Random3	2199.34	955.71	1556.71	1898.46	1729.50	2146.26
Random4	2187.15	948.53	1556.21	1898.59	1741.17	2146.26
Random5	2189.79	953.36	1549.37	1887.86	1713.57	2121.60
Random6	2201.58	958.58	1533.93	1889.99	1711.41	2146.26
Random7	2201.58	958.58	1533.93	1889.99	1711.41	2146.26
Random8	2201.58	963.25	1549.45	1863.82	1749.89	2146.26
Random9	2189.93	950.22	1532.24	1867.31	1698.02	2138.59
Random10	2199.34	962.10	1550.12	1894.07	1728.90	2137.07
Average	2197.35	957.68	1545.77	1886.78	1724.73	2138.85

C.3.2 Medium-Fleet Size, a=0.5

Table C.22: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
Random1	3959.23	110.26	2455.20	2527.29	1322.25	2320.21
Random2	3939.59	110.26	2448.99	2811.45	1307.28	2227.99
Random3	4059.59	109.34	2425.60	2755.57	1322.25	2383.77
Random4	4000.56	110.46	2370.11	2800.11	1319.14	2276.20
Random5	4052.49	110.46	2406.48	2816.57	1322.25	2337.29
Random6	4085.36	106.88	2450.51	2763.98	1321.32	2286.97
Random7	4085.36	106.88	2450.51	2763.98	1321.32	2286.97
Random8	4103.72	110.20	2256.06	2672.79	1322.25	2279.20
Random9	4073.82	110.46	2439.94	2678.33	1316.96	2188.15
Random10	4035.22	109.53	2443.02	2805.91	1322.25	2244.59
Average	4039.49	109.47	2414.64	2739.60	1319.73	2283.13

Table C.23: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
Random1	3380.79	585.80	1948.20	2463.41	1632.39	2192.03
Random2	3444.93	581.63	1963.59	2446.70	1617.92	2203.80
Random3	3457.71	564.25	1962.34	2426.85	1626.55	2223.21
Random4	3362.73	588.49	1968.03	2419.73	1546.53	2219.20
Random5	3362.73	589.44	1957.42	2422.88	1602.95	2223.21
Random6	3404.44	588.29	1968.03	2453.85	1632.39	2223.21
Random7	3404.44	588.29	1968.03	2453.85	1632.39	2223.21
Random8	3457.71	569.36	1966.88	2425.05	1601.16	2223.19
Random9	3335.20	578.17	1929.95	2334.77	1600.92	2223.21
Random10	3457.71	589.44	1968.03	2334.77	1632.39	2223.21
Average	3406.84	582.32	1960.05	2418.19	1612.56	2217.75

Table C.24: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

Order	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
Random1	2351.20	952.17	1543.43	1870.47	1718.76	2152.36
Random2	2327.24	952.17	1563.63	1872.40	1673.95	2152.36
Random3	2351.20	957.50	1563.23	1858.74	1728.71	2152.36
Random4	2351.20	950.22	1563.63	1821.08	1714.69	2143.16
Random5	2327.37	961.53	1548.47	1874.22	1738.71	2145.68
Random6	2351.20	960.34	1540.14	1869.15	1728.36	2121.12
Random7	2351.20	960.34	1540.14	1869.15	1728.36	2121.12
Random8	2341.77	957.84	1563.63	1843.38	1687.56	2110.87
Random9	2347.39	960.11	1562.86	1820.47	1732.26	2152.36
Random10	2351.20	960.56	1547.88	1818.75	1711.61	2139.47
Average	2345.10	957.28	1553.70	1851.78	1716.30	2139.09

C.3.3 Small-Fleet Size, a=0.7

Table C.25: Focusing on Minimising the Tardiness Cost,$\alpha = 0.1$

Order	mod_C101	mod_C201	mod_R101	mod_R201	mod_RC101	mod_RC201
Random1	8669.92	109.32	4153.82	5253.80	3613.11	4162.46
Random2	8568.61	110.24	4095.42	5320.09	3664.32	3922.50
Random3	8766.19	110.46	4140.70	5087.22	3685.01	4074.94
Random4	8671.04	110.46	4067.64	5319.12	3591.04	4102.95
Random5	8591.59	110.30	4076.39	5254.07	3470.65	3977.20
Random6	8766.19	110.46	4090.89	5311.20	3685.01	3915.03
Random7	8766.19	110.46	4090.89	5311.20	3685.01	3915.03
Random8	8721.51	108.77	4088.16	5134.92	3542.05	4162.46
Random9	8708.36	109.26	4208.43	5091.96	3505.73	4032.05
Random10	8676.24	108.27	4136.12	4976.98	3478.01	4162.46
Average	8690.58	109.80	4114.85	5206.06	3591.99	4042.71

Table C.26: Focusing on Minimising the Solution Cost,$\alpha = 0.5$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
Random1	5795.36	587.64	2837.20	3802.78	2132.37	3067.56
Random2	5795.36	578.83	2884.45	3806.57	2181.72	3187.46
Random3	5660.35	569.85	2930.52	3786.82	2182.64	3187.46
Random4	5775.81	577.82	2967.05	3688.03	2134.69	3187.46
Random5	5730.44	589.16	2956.74	3710.13	2182.64	3065.85
Random6	5789.56	589.34	2934.22	3814.62	2134.52	3087.26
Random7	5789.56	589.34	2934.22	3814.62	2134.52	3087.26
Random8	5768.80	587.52	2929.35	3801.61	2182.64	3181.46
Random9	5669.94	589.44	2922.44	3814.62	2182.64	3187.46
Random10	5711.31	573.19	2899.22	3814.62	2163.64	3137.73
Average	5748.65	583.21	2919.54	3785.44	2161.20	3137.70

Table C.27: Focusing on Minimising the Travelling Cost,$\alpha = 0.9$

Order	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
Random1	2587.13	962.41	1630.54	2015.72	1723.00	2105.03
Random2	2534.62	961.50	1631.45	1993.45	1737.13	2113.20
Random3	2598.73	960.61	1635.85	2015.72	1700.04	2111.92
Random4	2599.54	963.25	1608.40	1998.76	1702.15	2123.94
Random5	2615.97	957.84	1631.31	2007.65	1715.23	2123.94
Random6	2611.75	957.61	1634.00	2015.72	1737.13	2123.94
Random7	2611.75	957.61	1634.00	2015.72	1737.13	2123.94
Random8	2615.97	957.84	1636.88	2008.21	1718.81	2123.94
Random9	2613.17	963.25	1628.95	2011.54	1733.82	2104.63
Random10	2582.43	963.25	1626.18	2015.72	1729.60	2123.94
Average	2597.11	960.52	1629.76	2009.82	1723.40	2117.84

Appendix D

Experiments on LNS

Table D.1: Big Fleet Size, $a = 0.3$

Focusing on Tardiness Cost, $\alpha = 0.1$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	1046.98	110.46	834.49	742.83	415.23	438.18
10	1151.66	110.24	838.95	818.07	415.67	445.02
15	1133.79	110.46	842.24	698.34	415.67	444.79
20	1099.07	109.34	822.91	818.07	415.67	445.02
25	1161.55	110.46	817.75	818.07	396.78	445.02
30	1037.24	110.46	859.09	802.13	415.67	445.02
35	981.70	110.46	836.85	818.07	415.67	445.02
40	1083.76	110.46	802.14	818.07	415.67	393.17
45	997.24	110.46	833.77	727.02	415.67	435.36
50	1071.10	110.30	834.99	626.65	415.67	390.06
55	1049.39	110.46	859.09	649.87	408.25	386.24
60	1173.07	110.46	859.09	810.50	415.67	434.44
65	1173.07	109.39	824.62	616.15	414.85	445.02
70	1158.56	110.46	765.16	738.03	415.67	398.30
75	1154.00	110.46	802.86	818.07	415.67	411.00
80	1173.07	108.81	805.95	764.56	408.82	445.02
85	1012.40	110.46	859.09	697.25	415.67	431.44
Focusing on Solution Cost, $\alpha = 0.5$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	1994.12	588.48	1293.24	1383.61	1333.71	1551.38
10	1943.93	564.22	1293.24	1406.54	1330.59	1604.54
15	1978.28	584.68	1254.75	1406.54	1333.71	1615.05
20	1994.12	589.44	1293.24	1406.54	1316.88	1596.15
25	1950.75	586.70	1290.91	1368.37	1333.71	1582.18
30	1982.78	588.00	1262.54	1376.35	1333.71	1598.11
35	1994.12	578.20	1286.14	1406.54	1333.71	1619.23
40	1953.73	573.37	1293.24	1401.32	1318.56	1563.88
45	1994.12	581.27	1282.49	1406.54	1331.13	1547.97
50	1994.12	589.44	1252.88	1373.30	1319.91	1619.23
55	1970.65	575.67	1284.95	1406.54	1333.71	1604.46
60	1994.12	568.67	1278.41	1384.23	1333.71	1619.23
65	1950.48	581.57	1261.37	1406.54	1333.71	1602.47
70	1956.21	575.76	1293.24	1383.60	1333.71	1618.11
75	1934.28	572.14	1274.10	1334.82	1323.08	1619.23
80	1964.26	584.37	1252.00	1351.21	1316.09	1609.96
85	1955.73	588.47	1285.22	1344.40	1324.20	1551.38
Focusing on Traveling Cost, $\alpha = 0.9$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	2194.34	963.25	1551.36	1862.31	1746.29	2138.62
10	2200.44	963.25	1553.92	1881.14	1694.52	2146.26
15	2192.03	961.95	1554.02	1898.59	1728.96	2132.00
20	2201.58	963.25	1557.38	1881.08	1729.64	2139.91
25	2201.58	953.36	1557.38	1874.61	1683.96	2138.40
30	2201.58	959.74	1555.36	1897.00	1702.64	2146.26
35	2196.02	963.25	1555.18	1898.59	1736.09	2140.51
40	2195.45	963.25	1546.72	1889.85	1749.89	2146.26
45	2189.81	963.25	1554.26	1866.11	1737.18	2137.25
50	2201.58	963.25	1544.71	1855.60	1733.02	2141.65
55	2200.76	962.06	1557.38	1898.59	1739.07	2145.50
60	2201.58	963.25	1557.38	1841.36	1681.82	2146.26
65	2175.53	954.93	1557.38	1874.06	1710.54	2144.30
70	2190.35	957.61	1557.38	1898.59	1741.20	2146.26
75	2201.58	950.22	1540.78	1886.40	1689.24	2146.26
80	2192.79	958.51	1557.38	1888.62	1663.62	2146.26
85	2201.58	954.48	1551.34	1896.59	1719.88	2146.26

TABLE D.2: Medium Fleet Size, $a = 0.5$

Focusing on Tardiness Cost, $\alpha = 0.1$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	4103.72	109.68	2418.45	2816.57	1313.22	2110.78
10	4038.49	110.31	2392.82	2815.33	1270.09	2372.48
15	4103.72	110.46	2437.51	2653.10	1322.25	2383.77
20	4046.46	110.46	2441.14	2816.57	1322.25	2300.41
25	4103.72	109.34	2455.20	2586.13	1303.67	2302.12
30	4103.72	110.33	2437.94	2800.11	1322.25	2285.81
35	4103.72	110.46	2455.20	2816.57	1322.25	2234.07
40	4077.87	110.46	2274.15	2789.93	1296.54	2158.50
45	4031.69	110.46	2455.20	2816.57	1306.02	2299.57
50	3988.71	108.83	2289.93	2595.62	1322.25	2372.40
55	3916.78	110.03	2381.96	2799.21	1322.25	2357.09
60	4103.72	110.46	2367.86	2706.57	1288.71	2383.77
65	3880.75	110.18	2383.99	2742.73	1322.25	2291.63
70	3974.91	110.46	2455.20	2816.57	1322.25	2356.63
75	3996.55	110.46	2443.55	2601.13	1322.25	2243.33
80	4103.72	110.40	2376.25	2816.57	1300.95	2187.46
85	4080.14	110.46	2455.20	2811.80	1322.25	2335.06
Focusing on Solution Cost, $\alpha = 0.5$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	3442.39	589.44	1965.60	2415.19	1632.39	2223.21
10	3366.28	589.44	1924.84	2330.79	1632.39	2223.21
15	3457.71	565.70	1968.03	2410.53	1632.39	2223.21
20	3438.38	589.44	1909.13	2450.94	1632.39	2223.21
25	3452.04	589.18	1941.58	2463.41	1632.39	2192.27
30	3457.71	578.87	1934.86	2414.30	1632.39	2223.21
35	3376.20	569.65	1885.08	2424.18	1632.39	2223.09
40	3281.22	579.24	1939.66	2380.14	1632.39	2183.34
45	3383.30	572.19	1945.32	2380.87	1602.80	2223.21
50	3426.99	573.44	1958.01	2295.09	1579.17	2182.07
55	3347.42	582.13	1945.38	2333.84	1632.39	2166.20
60	3445.06	579.62	1930.63	2445.46	1632.39	2223.21
65	3457.71	583.45	1961.68	2400.67	1632.39	2159.39
70	3453.79	589.20	1937.83	2307.21	1602.51	2223.21
75	3457.71	577.10	1932.19	2326.23	1632.39	2223.21
80	3457.71	574.90	1937.54	2409.94	1571.13	2223.21
85	3457.71	567.56	1968.03	2358.40	1618.54	2129.56
Focusing on Traveling Cost, $\alpha = 0.9$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	2351.20	963.25	1563.63	1867.49	1726.60	2152.36
10	2309.00	963.25	1561.90	1869.15	1728.90	2152.36
15	2351.20	963.25	1563.63	1870.50	1713.67	2142.58
20	2351.20	959.30	1563.63	1804.01	1680.40	2115.71
25	2351.20	963.25	1535.62	1871.51	1659.32	2151.05
30	2302.07	956.77	1561.80	1874.88	1690.37	2148.52
35	2282.52	963.25	1554.94	1873.78	1732.26	2145.09
40	2351.20	963.25	1548.13	1843.40	1696.09	2146.07
45	2349.89	960.56	1563.63	1873.31	1712.58	2143.38
50	2298.53	963.25	1551.71	1854.09	1717.12	2124.36
55	2314.07	960.11	1558.27	1874.88	1708.97	2147.07
60	2313.52	960.56	1540.36	1851.90	1749.89	2141.91
65	2306.40	963.25	1563.63	1811.81	1707.54	2152.36
70	2348.46	963.25	1563.31	1827.10	1719.55	2114.80
75	2302.47	960.56	1554.94	1868.14	1661.83	2152.36
80	2351.20	954.70	1548.99	1869.15	1706.54	2140.55
85	2332.32	963.25	1548.62	1874.88	1695.47	2152.36

Table D.3: Small Fleet Size, $a = 0.5$

Focusing on Tardiness Cost, $\alpha = 0.1$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	8575.83	110.46	4023.06	5194.47	3598.21	4063.52
10	8766.19	110.46	4110.71	5320.09	3578.12	4131.03
15	8616.69	109.01	3986.73	5320.09	3311.05	4147.08
20	8744.38	109.34	4079.07	5212.04	3537.75	3766.44
25	8443.68	110.34	4089.28	5134.22	3413.62	3906.99
30	8669.48	110.46	4004.63	5227.25	3621.02	4142.35
35	8587.21	110.46	4056.54	5320.09	3665.28	3939.82
40	8766.19	110.46	4011.50	5257.53	3593.03	4132.44
45	8614.13	109.38	3990.26	5018.95	3449.67	4020.19
50	8739.58	110.46	4043.15	5200.64	3229.06	4009.53
55	8707.11	108.63	4093.71	5312.72	3427.32	4162.46
60	8635.97	110.36	4132.85	5320.09	3447.53	4151.50
65	8403.28	109.53	3998.10	5115.79	3457.19	4040.44
70	8618.40	109.68	4137.93	5253.80	3514.13	3938.85
75	8459.29	109.64	4120.52	5268.42	3401.09	4111.85
80	8676.61	110.46	4093.91	5081.14	3454.57	4162.46
85	8752.53	110.46	4046.81	5319.12	3459.62	3878.00
Focusing on Solution Cost, $\alpha = 0.5$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	5794.75	576.99	2870.70	3771.91	2161.26	3187.46
10	5795.36	589.44	2964.62	3713.77	2182.64	3121.03
15	5795.36	589.44	2959.20	3761.35	2163.64	3187.46
20	5794.53	573.01	2914.03	3814.62	2182.64	3162.69
25	5696.74	589.44	2867.32	3556.26	2182.64	3179.63
30	5795.36	580.50	2896.27	3555.14	2153.39	3187.46
35	5783.57	574.17	2878.49	3697.12	2102.57	3071.02
40	5784.51	582.79	2932.67	3713.98	2176.25	3173.53
45	5784.35	580.36	2961.76	3654.17	2118.98	3187.46
50	5795.36	579.05	2940.40	3812.56	2182.64	3187.46
55	5795.36	589.44	2907.52	3814.62	2169.39	3187.46
60	5744.38	580.86	2952.31	3760.30	2164.68	3187.46
65	5653.49	589.44	2844.93	3699.23	2060.47	3152.81
70	5765.93	584.33	2863.74	3564.48	2062.11	3187.46
75	5756.76	583.10	2920.06	3814.62	2159.14	3187.46
80	5739.40	568.53	2951.74	3814.62	2121.80	3187.46
85	5785.12	589.44	2967.05	3782.98	2163.86	3187.46
Focusing on Traveling Cost, $\alpha = 0.9$						
K	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	2602.54	963.25	1629.60	2015.72	1703.33	2092.79
10	2609.49	963.25	1625.87	2015.72	1691.55	2115.22
15	2615.97	952.63	1636.16	1998.76	1732.41	2123.94
20	2601.39	963.25	1637.33	2015.72	1737.13	2123.94
25	2589.98	963.25	1636.39	1944.66	1737.13	2113.87
30	2563.97	963.25	1637.33	2015.72	1683.52	2121.66
35	2558.59	960.56	1628.86	2013.60	1693.59	2121.54
40	2564.76	953.36	1634.75	2011.64	1708.80	2123.94
45	2581.47	958.32	1618.14	2011.86	1721.26	2123.94
50	2583.04	963.25	1617.43	2015.72	1690.70	2113.87
55	2537.56	953.39	1630.42	2015.72	1701.93	2123.94
60	2561.97	963.25	1630.56	2011.86	1720.69	2103.23
65	2590.34	949.88	1637.33	1956.45	1696.25	2116.15
70	2584.61	956.56	1635.66	1998.37	1701.35	2123.94
75	2588.42	963.25	1637.33	2011.86	1702.67	2123.94
80	2491.69	954.72	1637.33	2015.72	1737.13	2123.94
85	2612.98	933.66	1622.71	2012.38	1737.13	2123.94

Appendix E

Experiment of Tabu List Size

TABLE E.1: Big-Fleet Size, $a = 0.3$

Focusing on Tardiness Cost, $\alpha = 0.1$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	1034.50	104.24	811.24	787.74	471.26	485.47
10	1021.46	99.36	897.29	772.52	454.99	485.96
15	1065.00	99.02	796.19	740.39	450.99	485.96
20	1049.72	98.01	788.08	732.74	447.21	483.94
25	1059.09	99.42	818.16	736.32	453.01	485.96
30	1071.26	100.63	797.89	734.01	450.95	483.93
35	1055.54	100.63	801.56	740.10	455.43	485.96
40	980.17	100.63	814.55	738.08	455.43	485.96
Focusing on Solution Cost, $\alpha = 0.5$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	1931.45	649.51	1296.63	1360.51	1344.28	1612.17
10	1956.35	658.47	1299.25	1364.57	1344.95	1587.78
15	1941.62	659.23	1297.87	1389.02	1344.28	1593.15
20	1954.06	604.90	1292.42	1363.56	1344.95	1590.92
25	1953.65	603.97	1282.67	1376.05	1344.95	1593.15
30	1952.52	609.58	1300.72	1388.60	1344.95	1600.01
35	1950.78	626.53	1269.14	1383.60	1344.95	1604.04
40	1968.57	611.64	1282.83	1395.74	1344.95	1605.93
Focusing on Traveling Cost, $\alpha = 0.9$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	2131.72	2231.72	1596.37	1880.05	1804.93	2181.42
10	2132.85	2232.85	1594.77	1880.59	1745.49	2140.78
15	2132.32	2232.32	1577.22	1872.04	1726.54	2125.02
20	2139.60	2239.60	1587.04	1854.06	1745.24	2097.74
25	2132.96	2232.96	1580.57	1846.07	1730.74	2110.20
30	2143.86	2243.86	1599.58	1866.96	1727.78	2117.41
35	2136.87	2236.87	1595.39	1859.00	1752.81	2120.54
40	2149.88	2249.88	1605.15	1863.81	1728.17	2115.68

TABLE E.2: Medium-Fleet Size, $a = 0.3$

Focusing on Tardiness Cost, $\alpha = 0.1$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	4435.09	104.24	2475.39	2742.35	1360.99	2431.77
10	4313.15	99.36	2419.00	2684.70	1361.46	2248.19
15	4395.04	99.02	2432.45	2730.55	1347.72	2191.07
20	4327.52	98.01	2384.56	2710.83	1376.87	2236.86
25	4466.88	99.42	2471.11	2732.81	1380.23	2368.48
30	4408.95	100.63	2466.64	2730.63	1426.44	2299.43
35	4428.38	100.63	2484.97	2751.19	1336.57	2511.04
40	4459.07	100.63	2513.40	2751.19	1369.56	2513.54
Focusing on Solution Cost, $\alpha = 0.5$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	3493.04	599.51	1912.55	2427.78	1613.72	2211.90
10	3468.68	608.47	1907.31	2427.78	1611.92	2204.53
15	3442.89	609.23	1904.57	2416.79	1613.72	2171.03
20	3443.31	563.09	1919.80	2392.40	1598.94	2187.36
25	3478.47	592.36	1915.72	2428.64	1596.06	2211.56
30	3483.73	560.25	1889.77	2385.68	1613.72	2227.17
35	3488.49	577.57	1877.79	2428.64	1613.72	2227.17
40	3542.48	561.64	1906.02	2428.64	1613.72	2227.17
Focusing on Traveling Cost, $\alpha = 0.9$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	2331.70	1010.71	1577.71	1854.61	1804.93	2181.03
10	2330.95	998.97	1576.42	1847.99	1735.35	2176.23
15	2336.84	1002.57	1577.52	1853.93	1726.54	2128.19
20	2332.37	982.32	1574.05	1849.10	1739.55	2133.94
25	2292.52	989.09	1577.72	1856.22	1729.90	2131.40
30	2297.14	986.57	1577.72	1856.22	1727.78	2123.21
35	2305.46	998.07	1579.29	1851.01	1752.81	2145.92
40	2299.41	998.01	1579.29	1856.22	1728.17	2132.17

TABLE E.3: Small-Fleet Size, $a = 0.3$

Focusing on Tardiness Cost, $\alpha = 0.1$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	8742.08	104.24	4293.84	5273.39	3387.54	4396.70
10	8458.36	99.36	4293.66	5142.10	3397.86	4372.36
15	8526.46	100.42	4269.30	5361.69	3405.11	4423.69
20	8470.71	98.01	4081.24	5257.38	3423.77	4379.31
25	8400.15	99.42	4110.21	5264.95	3391.01	4538.20
30	8337.72	100.63	4164.29	5309.92	3434.79	4538.20
35	8588.25	100.63	4026.05	5306.15	3527.90	4538.20
40	8634.27	100.63	4127.24	5367.86	3547.51	4538.20
Focusing on Solution Cost, $\alpha = 0.5$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	5950.08	549.51	3114.45	3766.95	2109.46	3136.57
10	5773.66	558.47	2987.45	3737.12	2109.46	3058.58
15	5674.84	559.23	3089.92	3663.87	2123.15	2959.55
20	5747.72	512.66	3053.57	3662.04	2082.01	3050.54
25	5765.64	542.36	3029.48	3711.20	2116.56	3074.75
30	5755.21	510.25	3034.20	3705.24	2123.15	2996.96
35	5857.38	527.57	3151.66	3620.90	2116.82	2967.19
40	5876.61	511.64	3133.01	3762.83	2123.15	3097.89
Focusing on Traveling Cost, $\alpha = 0.9$						
$TabuListMax$	**mod_C101**	**mod_C201**	**mod_R101**	**mod_R201**	**mod_RC101**	**mod_RC201**
5	2446.40	960.71	1653.89	2060.77	1811.83	2092.82
10	2444.10	948.97	1634.05	2026.53	1740.55	2064.67
15	2447.63	949.25	1635.31	1994.34	1744.37	2069.47
20	2454.08	932.32	1625.17	1983.45	1735.13	2071.71
25	2466.69	939.09	1624.05	1980.47	1750.31	2041.49
30	2459.62	936.57	1598.82	2022.26	1732.80	2067.06
35	2469.71	958.71	1634.20	2049.51	1745.39	2072.51
40	2469.71	954.22	1592.02	2050.93	1740.95	2080.13

Appendix F

Computational Results of 56 instances of VRPRDD

F.1 Big-Fleet Size, a=0.3

F.2 Medium-Fleet Size, a=0.5

F.3 Small-Fleet Size, a=0.7

TABLE F.1: $a = 0.3$: Focusing on Minimize the Tardiness Cost,$\alpha = 0.1$

Instances	**VNS**	**LNS**	**TS**
mod_C101	1064.93	1098.29	1049.72
mod_C102	261.20	262.70	274.19
mod_C103	165.95	169.62	178.93
mod_C104	125.04	124.91	130.87
mod_C105	1277.61	1267.27	1300.45
mod_C106	1017.88	1034.72	954.63
mod_C107	698.95	709.61	650.78
mod_C108	388.54	388.54	378.62
mod_C109	229.73	229.73	237.68
mod_C201	110.78	110.46	108.01
mod_C202	106.75	109.42	116.44
mod_C203	114.83	117.44	114.58
mod_C204	112.13	111.16	113.51
mod_C205	103.72	103.22	106.95
mod_C206	104.86	106.32	107.75
mod_C207	104.86	104.26	106.32
mod_C208	103.11	103.23	106.25
mod_R101	833.49	859.09	788.08
mod_R102	533.75	489.59	489.32
mod_R103	364.60	347.65	347.76
mod_R104	256.32	237.87	250.22
mod_R105	802.85	725.78	735.41
mod_R106	433.28	436.58	409.82
mod_R107	339.21	331.33	364.16
mod_R108	234.42	248.68	241.10
mod_R109	686.09	662.22	634.30
mod_R110	351.07	402.20	343.00
mod_R111	363.79	368.52	394.53
mod_R112	297.28	307.87	298.89
mod_R201	805.07	808.07	832.74
mod_R202	178.23	180.87	183.75
mod_R203	144.21	142.33	164.58
mod_R204	122.87	121.30	138.39
mod_R205	478.29	463.81	433.03
mod_R206	149.67	150.29	161.08
mod_R207	136.51	137.54	150.55
mod_R208	126.86	124.28	129.06
mod_R209	202.40	195.53	211.79
mod_R210	172.10	171.74	184.66
mod_R211	171.24	170.09	182.40
mod_RC101	415.67	415.67	447.21
mod_RC102	582.25	602.12	557.05
mod_RC103	483.82	500.01	496.39
mod_RC104	303.34	303.34	303.34
mod_RC105	485.67	487.20	473.97
mod_RC106	587.69	588.04	587.41
mod_RC107	324.79	324.79	351.56
mod_RC108	359.79	359.79	356.89
mod_RC201	444.79	431.70	473.94
mod_RC202	362.63	358.94	362.25
mod_RC203	199.05	201.61	213.90
mod_RC204	159.42	158.64	165.59
mod_RC205	341.31	341.91	332.55
mod_RC206	881.41	903.48	890.04
mod_RC207	327.71	327.71	317.50
mod_RC208	231.63	231.63	251.79

Table F.2: $a = 0.3$: Focusing on Minimize the Solution Cost,$\alpha = 0.5$

Instances	**VNS**	**LNS**	**TS**
mod_C101	1955.45	1950.94	1954.06
mod_C102	1294.01	1281.94	1285.18
mod_C103	828.81	835.79	852.79
mod_C104	701.33	711.21	647.57
mod_C105	2117.32	2071.05	1963.54
mod_C106	2003.38	1960.46	1827.47
mod_C107	1856.70	1915.25	1797.42
mod_C108	1551.77	1495.14	1462.89
mod_C109	1128.66	1139.92	1133.43
mod_C201	588.90	582.03	604.90
mod_C202	536.05	543.47	537.18
mod_C203	564.50	537.81	534.57
mod_C204	543.78	561.22	530.66
mod_C205	517.37	521.02	495.72
mod_C206	525.02	529.79	501.08
mod_C207	521.46	522.09	491.60
mod_C208	518.13	519.98	492.50
mod_R101	1277.73	1289.13	1292.32
mod_R102	1133.99	1101.63	1132.58
mod_R103	942.12	942.33	975.42
mod_R104	834.18	873.66	913.46
mod_R105	1248.89	1255.03	1276.45
mod_R106	1054.98	1028.29	1048.47
mod_R107	919.59	918.17	977.66
mod_R108	845.31	843.04	888.65
mod_R109	1252.61	1265.40	1227.87
mod_R110	1009.55	1015.15	1053.47
mod_R111	969.93	961.73	1004.97
mod_R112	880.28	896.57	939.15
mod_R201	1369.64	1375.08	1363.56
mod_R202	839.82	859.37	852.22
mod_R203	722.18	725.03	759.45
mod_R204	637.80	619.55	653.21
mod_R205	1228.16	1194.30	1178.23
mod_R206	755.21	754.95	823.52
mod_R207	671.86	648.03	653.03
mod_R208	572.72	578.43	598.12
mod_R209	943.09	943.88	975.01
mod_R210	889.04	889.04	925.26
mod_R211	839.91	839.91	870.70
mod_RC101	1315.88	1333.71	1344.95
mod_RC102	1398.81	1399.19	1355.13
mod_RC103	1208.68	1184.36	1226.20
mod_RC104	997.20	997.20	1055.98
mod_RC105	1320.18	1335.91	1345.26
mod_RC106	1366.70	1370.55	1403.39
mod_RC107	1224.99	1236.14	1240.20
mod_RC108	1160.28	1168.99	1209.93
mod_RC201	1590.70	1605.94	1590.92
mod_RC202	1260.06	1245.32	1282.88
mod_RC203	994.36	994.36	1035.62
mod_RC204	800.86	810.97	799.01
mod_RC205	1386.52	1395.47	1454.38
mod_RC206	1513.87	1553.67	1599.78
mod_RC207	1336.68	1317.87	1317.60
mod_RC208	1176.27	1166.63	1200.16

TABLE F.3: $a = 0.3$: Focusing on Minimize the Traveling Cost,$\alpha = 0.9$

Instances	**VNS**	**LNS**	**TS**
mod_C101	2198.94	2201.58	2239.60
mod_C102	1790.81	1820.40	1866.73
mod_C103	1357.81	1332.30	1382.18
mod_C104	1073.83	1048.49	1136.23
mod_C105	2264.41	2257.50	2280.89
mod_C106	2148.01	2152.62	2223.82
mod_C107	2140.15	2141.96	2213.24
mod_C108	1968.49	1979.74	2017.38
mod_C109	1673.59	1683.01	1719.69
mod_C201	962.06	958.72	982.32
mod_C202	963.19	982.75	978.39
mod_C203	1000.67	1024.32	983.84
mod_C204	985.14	990.52	971.71
mod_C205	943.21	944.01	944.12
mod_C206	939.94	938.73	935.72
mod_C207	932.33	929.10	915.74
mod_C208	928.94	923.06	918.53
mod_R101	1546.11	1548.74	1587.04
mod_R102	1450.95	1437.61	1467.46
mod_R103	1382.87	1367.79	1401.43
mod_R104	1261.22	1263.70	1252.87
mod_R105	1560.80	1560.80	1589.85
mod_R106	1421.72	1429.77	1456.75
mod_R107	1362.67	1341.02	1360.56
mod_R108	1275.99	1280.71	1269.42
mod_R109	1566.27	1562.41	1497.59
mod_R110	1382.57	1392.92	1427.03
mod_R111	1357.31	1343.95	1359.16
mod_R112	1397.89	1316.06	1345.31
mod_R201	1877.73	1879.12	1854.06
mod_R202	1312.83	1260.86	1333.59
mod_R203	1228.53	1215.59	1193.13
mod_R204	1053.42	1055.25	1048.66
mod_R205	1723.29	1725.49	1718.66
mod_R206	1248.96	1245.09	1276.11
mod_R207	1107.99	1094.84	1108.00
mod_R208	991.26	1005.77	1022.97
mod_R209	1456.05	1464.60	1520.43
mod_R210	1356.25	1357.38	1380.43
mod_R211	1289.52	1292.69	1349.79
mod_RC101	1711.38	1724.56	1745.24
mod_RC102	1678.14	1679.91	1691.76
mod_RC103	1540.52	1542.51	1593.52
mod_RC104	1368.71	1373.22	1457.49
mod_RC105	1686.35	1720.00	1711.50
mod_RC106	1733.38	1749.67	1782.29
mod_RC107	1531.00	1564.15	1577.56
mod_RC108	1533.97	1499.64	1527.91
mod_RC201	2146.26	2146.26	2097.74
mod_RC202	1930.43	1938.65	1960.83
mod_RC203	1476.54	1489.62	1602.66
mod_RC204	1309.71	1318.03	1413.46
mod_RC205	2022.29	2045.29	2152.44
mod_RC206	2134.98	2160.43	2236.95
mod_RC207	1927.90	1948.89	2038.20
mod_RC208	1716.45	1731.49	1761.39

Table F.4: $a = 0.5$: Focusing on Minimize the Tardiness Cost,$\alpha = 0.1$

Instances	**VNS**	**LNS**	**TS**
mod_C101	3958.44	3871.20	4327.85
mod_C102	947.25	1010.19	1011.56
mod_C103	163.20	169.62	170.59
mod_C104	123.99	124.83	130.56
mod_C105	4008.38	4238.18	4217.07
mod_C106	4226.74	4257.34	4362.79
mod_C107	3818.44	3690.29	3712.49
mod_C108	1912.21	1935.37	1931.94
mod_C109	506.35	513.48	520.76
mod_C201	109.58	110.46	108.01
mod_C202	111.00	109.30	110.54
mod_C203	115.95	116.74	116.29
mod_C204	112.92	113.15	114.01
mod_C205	103.30	103.91	104.65
mod_C206	105.75	106.32	105.60
mod_C207	104.39	104.92	105.31
mod_C208	103.20	103.03	96.25
mod_R101	2368.11	2404.20	2384.56
mod_R102	2427.39	2450.18	2570.67
mod_R103	1760.78	1658.80	1645.38
mod_R104	873.99	945.53	937.91
mod_R105	2298.06	2270.73	2280.26
mod_R106	2014.06	1889.25	1843.12
mod_R107	1469.36	1390.26	1326.98
mod_R108	839.77	872.47	870.24
mod_R109	2154.54	2207.13	2239.10
mod_R110	1205.68	1259.94	1242.50
mod_R111	1428.28	1507.46	1506.70
mod_R112	1087.35	1132.66	1071.21
mod_R201	2753.41	2696.73	2710.83
mod_R202	202.41	202.88	196.10
mod_R203	145.88	146.89	146.00
mod_R204	124.78	126.30	120.50
mod_R205	1559.94	1579.83	1524.43
mod_R206	153.67	157.51	151.37
mod_R207	132.71	134.12	134.59
mod_R208	120.44	119.54	118.65
mod_R209	630.74	659.11	674.33
mod_R210	255.51	242.07	259.71
mod_R211	216.75	198.35	216.80
mod_RC101	1285.00	1322.25	1376.87
mod_RC102	1688.33	1639.63	1644.19
mod_RC103	1588.39	1522.25	1505.70
mod_RC104	1163.35	1222.42	1236.29
mod_RC105	1335.32	1335.32	1322.40
mod_RC106	1363.63	1367.02	1390.09
mod_RC107	1181.83	1224.11	1182.29
mod_RC108	1328.51	1309.30	1339.96
mod_RC201	2066.36	2227.92	2236.86
mod_RC202	1213.62	1106.11	1160.29
mod_RC203	610.84	592.67	596.08
mod_RC204	161.81	162.02	162.72
mod_RC205	1001.33	1039.44	1048.37
mod_RC206	2001.49	2001.49	2036.57
mod_RC207	1056.24	1005.28	1060.85
mod_RC208	546.77	567.73	549.13

TABLE F.5: $a = 0.5$: Focusing on Minimize the Solution Cost,$\alpha = 0.5$

Instances	**VNS**	**LNS**	**TS**
mod_C101	3417.59	3408.55	3443.31
mod_C102	1667.49	1673.89	1646.21
mod_C103	815.98	828.51	824.62
mod_C104	677.43	714.42	682.50
mod_C105	3602.98	3607.44	3600.64
mod_C106	3578.74	3568.48	3540.09
mod_C107	3333.65	3363.12	3634.66
mod_C108	2256.96	2187.94	2195.86
mod_C109	1438.35	1446.19	1410.99
mod_C201	575.05	577.42	563.09
mod_C202	530.54	546.84	532.74
mod_C203	568.74	555.49	534.57
mod_C204	563.67	563.73	564.06
mod_C205	514.44	520.78	485.72
mod_C206	524.96	524.45	512.78
mod_C207	524.28	524.28	511.60
mod_C208	512.84	520.48	534.50
mod_R101	1960.49	1913.35	1919.80
mod_R102	1701.76	1726.02	1702.33
mod_R103	1461.13	1531.75	1243.22
mod_R104	1106.96	1129.28	1192.22
mod_R105	1900.46	1914.30	1941.59
mod_R106	1671.34	1710.73	1792.94
mod_R107	1285.46	1294.71	1265.06
mod_R108	1138.30	1151.05	1164.09
mod_R109	1790.91	1757.23	1708.64
mod_R110	1432.80	1448.78	1492.90
mod_R111	1299.80	1294.06	1184.47
mod_R112	1254.70	1225.24	1209.80
mod_R201	2410.19	2398.27	2392.40
mod_R202	820.19	825.26	887.97
mod_R203	689.67	708.75	702.23
mod_R204	619.30	618.33	635.48
mod_R205	1697.42	1695.05	1692.18
mod_R206	728.80	725.85	725.56
mod_R207	664.49	681.75	624.48
mod_R208	573.01	568.67	569.15
mod_R209	1115.59	1134.90	1154.37
mod_R210	932.30	986.61	989.43
mod_R211	836.84	822.07	872.55
mod_RC101	1602.99	1618.00	1598.94
mod_RC102	1682.28	1711.75	1747.55
mod_RC103	1461.17	1502.06	1554.84
mod_RC104	1171.82	1199.78	1155.44
mod_RC105	1621.60	1639.42	1646.18
mod_RC106	1733.65	1816.46	1816.13
mod_RC107	1417.83	1464.09	1477.75
mod_RC108	1376.57	1405.97	1402.82
mod_RC201	2178.44	2223.21	2187.36
mod_RC202	1619.03	1632.99	1697.96
mod_RC203	1079.62	1171.16	1154.38
mod_RC204	830.55	855.33	813.04
mod_RC205	1680.58	1680.58	1622.61
mod_RC206	2218.83	2200.81	2289.46
mod_RC207	1440.93	1509.10	1540.96
mod_RC208	1237.05	1307.16	1255.17

TABLE F.6: $a = 0.5$: Focusing on Minimize the Traveling Cost,$\alpha = 0.9$

Instances	**VNS**	**LNS**	**TS**
mod_C101	2351.05	2351.20	2332.37
mod_C102	1806.28	1809.41	1866.73
mod_C103	1330.32	1358.01	1382.18
mod_C104	1040.68	1071.81	1032.65
mod_C105	2326.48	2325.59	2287.98
mod_C106	2418.76	2428.66	2453.24
mod_C107	2199.29	2202.10	2223.30
mod_C108	2071.65	2077.96	2011.99
mod_C109	1663.60	1670.37	1679.69
mod_C201	957.61	958.51	982.32
mod_C202	939.46	959.77	951.52
mod_C203	1000.47	999.29	1033.84
mod_C204	978.41	979.43	921.71
mod_C205	942.68	948.01	994.12
mod_C206	929.53	951.57	974.92
mod_C207	932.04	933.29	965.74
mod_C208	927.89	927.00	968.53
mod_R101	1561.80	1545.86	1574.05
mod_R102	1440.44	1454.01	1467.46
mod_R103	1378.07	1385.58	1361.73
mod_R104	1263.70	1263.70	1252.87
mod_R105	1520.47	1556.55	1554.57
mod_R106	1421.52	1444.27	1456.75
mod_R107	1348.61	1367.43	1357.23
mod_R108	1266.21	1274.99	1258.42
mod_R109	1570.02	1597.22	1597.36
mod_R110	1378.48	1372.25	1327.03
mod_R111	1342.87	1362.77	1353.61
mod_R112	1410.16	1370.54	1421.70
mod_R201	1858.56	1874.88	1849.10
mod_R202	1304.85	1310.54	1333.59
mod_R203	1216.15	1225.89	1293.13
mod_R204	1033.96	1041.55	1048.66
mod_R205	1693.16	1707.51	1692.58
mod_R206	1226.94	1259.19	1276.11
mod_R207	1102.52	1100.71	1108.00
mod_R208	989.54	998.81	992.97
mod_R209	1493.31	1495.54	1412.36
mod_R210	1353.69	1370.97	1380.43
mod_R211	1277.82	1277.35	1234.57
mod_RC101	1679.36	1735.86	1739.55
mod_RC102	1659.59	1657.85	1691.76
mod_RC103	1519.30	1541.39	1588.34
mod_RC104	1395.77	1384.99	1331.88
mod_RC105	1714.90	1660.16	1710.90
mod_RC106	1694.32	1746.69	1770.26
mod_RC107	1516.16	1555.42	1568.18
mod_RC108	1495.66	1533.97	1525.96
mod_RC201	2108.37	2134.70	2133.94
mod_RC202	1941.59	1944.28	1939.12
mod_RC203	1489.62	1489.62	1402.66
mod_RC204	1314.02	1311.07	1313.46
mod_RC205	1998.53	1994.12	1995.11
mod_RC206	2196.37	2200.18	2215.15
mod_RC207	1933.50	1959.10	1940.28
mod_RC208	1716.02	1721.25	1761.39

TABLE F.7: $a = 0.7$: Focusing on Minimize the Tardiness Cost,$\alpha = 0.1$

Instances	**VNS**	**LNS**	**TS**
mod_C101	8670.49	8653.27	8470.71
mod_C102	4056.15	3878.17	3839.77
mod_C103	193.38	193.55	199.58
mod_C104	123.26	123.47	125.42
mod_C105	8927.20	8670.22	8682.34
mod_C106	9117.68	8758.05	9568.52
mod_C107	8265.26	8396.82	8946.15
mod_C108	5823.05	5917.67	5894.56
mod_C109	2949.21	3019.25	2979.01
mod_C201	98.10	100.46	98.01
mod_C202	110.89	109.06	116.44
mod_C203	117.30	118.28	124.25
mod_C204	111.61	111.26	113.51
mod_C205	104.20	103.06	106.95
mod_C206	105.26	105.57	108.15
mod_C207	103.43	103.61	106.32
mod_C208	103.23	102.70	106.25
mod_R101	4107.44	4088.19	4081.24
mod_R102	4396.93	4468.18	4370.66
mod_R103	2721.69	2697.98	2715.71
mod_R104	2040.65	2150.09	2167.50
mod_R105	4121.18	3950.16	4243.11
mod_R106	3465.00	3443.58	3551.01
mod_R107	2557.80	2630.97	2614.19
mod_R108	1854.01	1896.54	1839.31
mod_R109	3717.65	3725.78	3700.15
mod_R110	2838.97	2846.97	2895.84
mod_R111	2446.74	2238.92	2376.69
mod_R112	2361.72	2306.41	2344.03
mod_R201	4952.27	5276.84	5257.38
mod_R202	373.50	340.58	373.40
mod_R203	295.24	287.65	296.86
mod_R204	122.31	121.25	121.53
mod_R205	3367.24	3281.66	3334.46
mod_R206	281.04	264.47	274.62
mod_R207	278.08	257.79	278.92
mod_R208	129.59	130.84	138.31
mod_R209	1547.95	1552.91	1587.17
mod_R210	1016.51	1023.97	1065.89
mod_R211	461.21	454.61	490.56
mod_RC101	3609.48	3515.47	3423.77
mod_RC102	3279.33	3398.88	3448.10
mod_RC103	2658.70	2649.62	2666.44
mod_RC104	2001.14	2094.90	2047.67
mod_RC105	3300.81	3346.40	3539.21
mod_RC106	3698.00	3777.85	3576.66
mod_RC107	2491.16	2500.99	2661.17
mod_RC108	2427.78	2282.71	2480.06
mod_RC201	4038.85	4073.49	4379.31
mod_RC202	2150.61	2179.57	2156.10
mod_RC203	1057.28	1027.88	1042.04
mod_RC204	585.38	585.92	626.54
mod_RC205	2598.00	2789.72	2723.39
mod_RC206	4048.18	4094.05	4050.96
mod_RC207	2389.00	2318.15	2480.62
mod_RC208	1206.88	1325.54	1233.14

TABLE F.8: $a = 0.7$: Focusing on Minimize the Solution Cost,$\alpha = 0.5$

Instances	**VNS**	**LNS**	**TS**
mod_C101	5706.52	5730.37	5747.72
mod_C102	3298.10	3173.64	3329.21
mod_C103	858.81	872.67	804.92
mod_C104	686.28	676.99	695.33
mod_C105	6046.79	6147.11	6243.76
mod_C106	6192.78	6488.27	6407.28
mod_C107	6137.16	6161.72	6137.13
mod_C108	4499.99	4250.27	4653.97
mod_C109	2356.71	2373.55	2311.50
mod_C201	589.37	585.02	592.66
mod_C202	526.21	530.92	532.74
mod_C203	556.45	568.01	564.57
mod_C204	550.10	562.58	570.66
mod_C205	520.48	515.81	555.72
mod_C206	529.79	514.19	551.08
mod_C207	524.60	524.60	581.60
mod_C208	517.03	513.81	522.50
mod_R101	2891.06	2921.04	3053.57
mod_R102	2720.27	2803.71	2812.69
mod_R103	2142.56	2153.38	2176.24
mod_R104	1769.87	1835.39	1855.06
mod_R105	3206.70	3131.06	3490.01
mod_R106	2416.50	2331.96	2452.13
mod_R107	1990.67	2065.72	2052.19
mod_R108	1693.14	1701.85	1730.79
mod_R109	2900.32	3011.81	3035.07
mod_R110	2359.77	2364.42	2458.72
mod_R111	1823.96	1924.23	1968.41
mod_R112	1852.28	1882.65	1884.58
mod_R201	3699.23	3733.27	3662.04
mod_R202	886.60	895.30	842.36
mod_R203	790.99	777.22	747.81
mod_R204	646.22	632.34	670.29
mod_R205	2905.50	2909.35	2912.92
mod_R206	783.82	781.69	783.94
mod_R207	621.85	628.31	622.50
mod_R208	585.27	594.88	631.54
mod_R209	2101.12	1919.42	1963.65
mod_R210	1042.26	1049.98	1086.84
mod_R211	992.40	1029.17	1028.18
mod_RC101	2130.45	2182.64	2082.01
mod_RC102	2454.12	2415.12	2408.05
mod_RC103	2065.81	2023.03	2045.53
mod_RC104	1808.13	1912.51	1939.91
mod_RC105	2058.92	2051.64	2081.22
mod_RC106	2264.79	2343.31	2335.30
mod_RC107	2158.05	2156.52	2167.53
mod_RC108	2079.75	2089.95	2102.89
mod_RC201	3133.36	3187.46	3050.54
mod_RC202	2453.93	2267.50	2387.25
mod_RC203	1630.81	1512.79	1551.50
mod_RC204	1318.22	1347.56	1337.15
mod_RC205	2374.58	2253.71	2321.48
mod_RC206	3641.71	3583.41	3654.94
mod_RC207	2008.50	2075.15	2013.33
mod_RC208	1611.66	1620.87	1668.36

TABLE F.9: $a = 0.7$: Focusing on Minimize the Traveling Cost,$\alpha = 0.9$

Instances	**VNS**	**LNS**	**TS**
mod_C101	2579.62	2564.76	2454.08
mod_C102	2082.11	2150.79	2180.62
mod_C103	1349.28	1353.46	1382.18
mod_C104	1075.77	1072.75	1086.23
mod_C105	2669.67	2690.50	2649.56
mod_C106	2652.66	2606.35	2635.39
mod_C107	2586.98	2592.89	2559.11
mod_C108	2348.63	2398.70	2388.04
mod_C109	1941.46	1842.25	1982.60
mod_C201	958.62	958.51	932.32
mod_C202	930.55	959.34	938.28
mod_C203	1000.90	1002.72	1003.84
mod_C204	989.61	987.59	991.71
mod_C205	951.00	949.16	994.12
mod_C206	935.67	942.47	985.72
mod_C207	934.76	934.76	965.74
mod_C208	936.23	936.23	868.53
mod_R101	1637.33	1632.41	1625.17
mod_R102	1534.22	1543.66	1591.16
mod_R103	1416.46	1420.53	1418.01
mod_R104	1320.58	1293.59	1302.45
mod_R105	1641.61	1654.09	1647.96
mod_R106	1515.40	1466.60	1548.10
mod_R107	1396.35	1413.32	1467.95
mod_R108	1254.37	1260.20	1294.50
mod_R109	1624.37	1600.85	1668.91
mod_R110	1481.70	1464.22	1424.78
mod_R111	1412.97	1405.67	1484.84
mod_R112	1383.62	1381.01	1345.31
mod_R201	2012.20	2015.72	1983.45
mod_R202	1288.03	1280.32	1299.80
mod_R203	1255.93	1200.66	1203.74
mod_R204	1042.29	1035.23	1065.08
mod_R205	1837.93	1852.33	1827.38
mod_R206	1277.18	1259.60	1266.57
mod_R207	1100.71	1103.45	1108.00
mod_R208	1006.32	1008.44	1022.97
mod_R209	1541.69	1553.52	1500.32
mod_R210	1376.00	1383.07	1384.90
mod_R211	1293.55	1314.23	1349.86
mod_RC101	1692.19	1702.56	1735.13
mod_RC102	1679.07	1655.00	1692.43
mod_RC103	1519.37	1539.22	1579.52
mod_RC104	1395.77	1384.81	1397.49
mod_RC105	1704.00	1699.20	1713.03
mod_RC106	1744.90	1726.21	1735.33
mod_RC107	1551.65	1529.71	1570.03
mod_RC108	1530.87	1490.59	1527.91
mod_RC201	2123.94	2121.66	2171.71
mod_RC202	1922.00	1906.48	1970.61
mod_RC203	1480.60	1489.49	1484.22
mod_RC204	1324.09	1341.85	1377.29
mod_RC205	2031.46	2033.57	2024.44
mod_RC206	2273.16	2273.16	2219.31
mod_RC207	1889.69	1929.41	1950.10
mod_RC208	1741.12	1776.27	1751.54

Appendix G

Computational Results of 56 instances on PS-VRPRDD

Table G.1: Big-Fleet Size

Instance	**Focus on Tardiness, $\alpha = 0.1$**		**Focus Equally, $\alpha = 0.5$**		**Focus on Traveling, $\alpha = 0.9$**	
	Alternate	**InOneMove**	**Alternate**	**InOneMove**	**Alternate**	**InOneMove**
mod_C101	783.93	464.42	1630.31	1367.81	1886.28	1765.44
mod_C102	244.40	237.64	1147.24	1432.13	1480.11	1489.71
mod_C103	154.98	158.47	735.67	726.41	1175.89	1120.91
mod_C104	123.64	120.37	622.57	680.77	956.73	969.90
mod_C105	744.19	530.56	1679.91	1450.98	1945.66	1818.51
mod_C106	660.24	506.06	1535.45	1408.57	1831.43	1912.70
mod_C107	425.28	308.95	1561.65	1335.74	1849.72	1676.87
mod_C108	291.61	262.13	1311.84	1192.05	1669.63	1569.06
mod_C109	209.05	204.68	1059.92	988.93	1343.32	1422.02
mod_C201	103.56	429.28	533.39	603.05	898.78	1022.84
mod_C202	107.09	107.01	527.24	519.18	944.07	901.53
mod_C203	112.03	118.78	546.71	562.71	907.36	1138.81
mod_C204	102.29	113.46	524.48	567.37	937.63	932.65
mod_C205	98.37	98.70	490.79	493.51	884.08	907.69
mod_C206	102.49	102.87	502.71	507.05	911.40	914.27
mod_C207	99.06	103.27	491.04	496.51	877.60	889.44
mod_C208	101.50	101.73	492.88	497.29	879.90	893.07
mod_R101	434.15	399.87	1085.70	930.53	1248.44	1161.12
mod_R102	287.10	189.69	930.93	850.18	1255.83	1132.00
mod_R103	205.56	156.83	811.91	742.18	1184.81	987.05
mod_R104	182.85	153.10	770.45	731.86	1083.69	997.51
mod_R105	530.53	350.23	1068.40	997.87	1332.36	1160.35
mod_R106	243.34	164.42	893.96	837.30	1204.05	1094.97
mod_R107	201.54	164.92	805.25	764.69	1116.39	1031.45
mod_R108	170.33	135.92	705.51	676.59	1112.13	1000.73
mod_R109	467.51	264.14	1032.75	964.31	1204.50	1069.56
mod_R110	283.12	196.99	921.04	963.88	1173.07	1046.84
mod_R111	223.40	173.15	812.63	858.21	1119.51	960.84
mod_R112	203.47	172.06	804.85	741.99	1017.96	1002.98
mod_R201	637.43	424.27	1196.55	1175.83	1626.95	1793.25
mod_R202	155.45	166.51	759.99	775.21	1168.91	1135.51
mod_R203	136.29	128.50	694.01	660.32	1101.22	1028.05
mod_R204	111.42	118.64	586.34	723.80	953.63	948.87
mod_R205	349.64	282.02	1073.49	945.96	1514.06	1490.76
mod_R206	138.90	136.61	734.36	676.54	1123.68	1001.77
mod_R207	127.79	131.07	629.29	617.40	996.18	969.72
mod_R208	115.66	126.88	539.36	546.68	912.52	927.63
mod_R209	185.70	195.22	909.13	943.60	1276.44	1308.58
mod_R210	159.23	159.00	799.73	822.32	1190.98	1225.66
mod_R211	159.22	157.73	732.73	813.68	1169.35	1331.72
mod_RC101	283.13	255.38	1166.77	1199.41	1303.73	1264.12
mod_RC102	383.60	383.51	1093.54	1185.49	1345.31	1274.76
mod_RC103	375.73	345.94	1005.53	1069.35	1328.63	1329.14
mod_RC104	250.59	213.05	873.63	899.50	1152.63	1096.25
mod_RC105	335.08	232.20	1136.40	1144.10	1257.52	1233.67
mod_RC106	319.31	395.83	1201.90	1204.86	1307.63	1329.31
mod_RC107	249.17	223.60	1112.32	1044.17	1242.19	1185.85
mod_RC108	248.07	255.43	1015.45	1040.86	1195.16	1125.64
mod_RC201	344.17	316.84	1396.23	1487.69	1839.30	1743.48
mod_RC202	276.44	223.94	1102.09	1151.64	1652.53	1642.93
mod_RC203	188.95	195.49	909.75	932.24	1342.40	1220.58
mod_RC204	146.43	158.03	760.25	779.70	1234.54	1199.43
mod_RC205	289.59	282.29	1303.42	1362.18	1677.63	1763.78
mod_RC206	630.84	440.35	1522.02	1593.14	1777.61	1829.28
mod_RC207	309.88	269.71	1202.83	1228.56	1633.43	1668.59
mod_RC208	208.83	228.93	1033.72	1087.62	1501.89	1483.16

Table G.2: Medium-Fleet Size

Instance	**Focus on Tardiness,** $\alpha = 0.1$		**Focus Equally,** $\alpha = 0.5$		**Focus on Traveling,** $\alpha = 0.9$	
	Alternate	**InOneMove**	**Alternate**	**InOneMove**	**Alternate**	**InOneMove**
mod_C101	2720.10	1493.31	2364.34	1739.03	1869.23	1619.00
mod_C102	523.51	285.41	1304.50	1183.57	1495.97	1486.46
mod_C103	155.91	153.39	764.49	705.76	1160.02	1142.40
mod_C104	124.28	119.66	628.79	677.46	964.08	967.68
mod_C105	2643.51	1627.65	2512.56	1721.47	1877.63	1727.30
mod_C106	2907.71	1682.59	2690.36	1974.05	1860.43	1816.04
mod_C107	2055.93	1192.60	2052.13	1603.30	1766.32	1653.09
mod_C108	773.38	561.85	1524.46	1145.40	1642.21	1544.86
mod_C109	319.15	208.58	1006.69	929.92	1369.06	1357.79
mod_C201	102.43	121.63	529.56	587.29	892.06	1035.14
mod_C202	104.61	107.48	526.30	521.48	936.56	901.53
mod_C203	112.75	115.18	537.69	562.71	904.34	1138.81
mod_C204	102.87	113.75	518.83	567.37	934.75	932.65
mod_C205	98.96	98.64	490.62	493.51	882.37	903.99
mod_C206	102.62	102.87	496.27	507.05	890.48	914.76
mod_C207	97.55	374.06	486.52	498.31	876.83	889.44
mod_C208	99.98	102.00	492.72	493.72	882.40	893.23
mod_R101	1241.33	842.11	1354.24	1084.86	1263.89	1148.05
mod_R102	1043.22	629.13	1074.94	954.43	1263.10	1161.45
mod_R103	495.37	425.88	903.26	771.81	1180.31	969.55
mod_R104	428.04	289.74	830.20	698.05	1084.52	1045.26
mod_R105	1256.03	718.93	1266.21	1081.32	1280.60	1108.26
mod_R106	773.64	453.93	1039.93	809.18	1193.85	1051.01
mod_R107	546.36	368.69	801.08	761.69	1116.75	1056.10
mod_R108	442.33	283.30	786.26	692.45	1110.19	1046.39
mod_R109	1035.96	595.54	1120.46	900.39	1210.15	1054.33
mod_R110	821.36	450.55	1008.38	857.20	1169.34	1055.04
mod_R111	624.65	377.38	827.41	711.52	1123.77	1091.61
mod_R112	555.05	373.93	864.95	699.14	1030.06	1011.83
mod_R201	1967.86	1459.44	1789.75	1546.55	1506.00	1371.64
mod_R202	169.66	148.15	778.20	698.76	1176.03	1131.99
mod_R203	140.86	129.31	665.67	651.82	1097.44	1022.47
mod_R204	120.07	121.86	584.08	572.14	941.30	920.42
mod_R205	1048.70	667.58	1355.05	1303.91	1448.95	1333.67
mod_R206	144.77	133.96	692.58	664.50	1122.29	988.10
mod_R207	126.49	386.56	630.02	880.20	995.63	1003.09
mod_R208	113.41	122.88	532.79	546.68	916.86	923.54
mod_R209	320.97	209.37	919.78	807.04	1293.27	1226.79
mod_R210	164.42	143.53	764.66	670.11	1208.44	1218.44
mod_R211	153.72	140.01	718.07	714.83	1157.07	1125.27
mod_RC101	842.57	557.00	1225.36	1072.10	1298.23	1228.09
mod_RC102	972.87	640.72	1285.17	1150.09	1340.42	1201.41
mod_RC103	993.45	566.52	1175.99	905.07	1335.63	1347.25
mod_RC104	716.77	539.23	944.43	791.77	1149.71	1169.35
mod_RC105	706.01	396.52	1178.73	1085.54	1251.66	1233.76
mod_RC106	760.88	461.53	1330.61	1029.40	1323.08	1328.75
mod_RC107	672.35	388.15	1072.54	912.84	1238.30	1159.85
mod_RC108	774.03	471.43	986.80	855.33	1185.54	1104.44
mod_RC201	1123.64	700.26	1833.59	1432.56	1814.02	1682.08
mod_RC202	920.67	448.12	1241.32	1142.51	1624.04	1733.02
mod_RC203	508.92	431.91	974.24	1033.31	1342.23	1289.19
mod_RC204	147.73	145.06	759.24	684.16	1233.19	1129.84
mod_RC205	644.94	373.72	1360.88	1117.54	1668.30	1778.47
mod_RC206	1296.84	942.26	1833.60	1466.87	1745.74	1777.64
mod_RC207	723.44	421.95	1295.97	1045.70	1685.08	1708.37
mod_RC208	411.65	289.84	1241.66	1166.81	1495.33	1367.49

TABLE G.3: Small-Fleet Size

Instance	Focus on Tardiness, $\alpha = 0.1$		Focus Equally, $\alpha = 0.5$		Focus on Traveling, $\alpha = 0.9$	
	Alternate	InOneMove	Alternate	InOneMove	Alternate	InOneMove
mod_C101	5927.02	3702.04	3776.82	2863.63	1939.88	1809.98
mod_C102	2698.75	1151.58	1999.69	1724.72	1565.77	1450.63
mod_C103	169.14	157.57	766.06	739.87	1165.68	1133.02
mod_C104	121.14	119.12	625.11	621.64	958.92	966.99
mod_C105	6265.34	4007.96	4066.93	3225.71	1965.32	1741.79
mod_C106	6492.27	4053.84	4374.99	3082.26	2015.16	1795.37
mod_C107	5105.80	3239.85	3893.56	2686.84	1983.96	1754.86
mod_C108	3609.48	2203.12	2804.78	2000.69	1873.31	1677.28
mod_C109	1079.81	557.27	1433.31	1066.99	1310.78	1344.73
mod_C201	104.38	120.98	521.27	587.83	875.73	1022.84
mod_C202	106.99	107.99	532.99	520.11	932.00	899.15
mod_C203	111.94	118.27	543.50	562.71	909.34	1138.81
mod_C204	104.02	116.14	522.10	567.61	948.34	932.65
mod_C205	98.48	98.70	488.80	493.51	889.88	903.99
mod_C206	102.50	102.87	501.19	507.05	886.14	914.27
mod_C207	98.75	99.37	491.73	496.83	883.09	892.59
mod_C208	99.73	101.73	483.18	494.69	884.98	888.69
mod_R101	2259.46	1850.54	1802.08	1449.22	1242.25	1196.97
mod_R102	2120.19	1477.29	1686.13	1267.07	1164.84	1130.58
mod_R103	1504.17	880.38	1294.17	1037.51	1086.48	1037.78
mod_R104	880.93	645.95	1000.31	862.75	1089.23	1099.66
mod_R105	2468.98	1493.81	1835.79	1345.08	1252.43	1194.18
mod_R106	1640.10	1113.44	1352.73	1167.77	1215.82	1092.33
mod_R107	1304.69	859.81	1102.28	955.04	1093.45	1048.98
mod_R108	984.51	665.26	988.86	809.64	1064.16	946.86
mod_R109	2323.62	1257.46	1842.01	1322.74	1235.30	1123.19
mod_R110	1517.96	1076.46	1364.18	1136.04	1125.25	1090.29
mod_R111	1198.67	829.45	1007.32	914.16	1096.26	1038.29
mod_R112	1112.00	632.42	1156.51	872.82	1026.66	1011.77
mod_R201	3616.62	2295.18	2733.00	2095.49	1587.54	1485.89
mod_R202	189.94	139.89	782.04	634.21	1148.18	1023.68
mod_R203	150.67	187.34	625.55	590.10	1065.36	996.32
mod_R204	112.97	108.34	548.44	561.94	955.39	946.73
mod_R205	2524.18	1737.15	2182.25	1802.48	1504.50	1361.17
mod_R206	170.35	142.38	701.20	782.52	1104.33	975.51
mod_R207	132.64	200.78	615.44	584.22	1001.57	961.17
mod_R208	110.07	109.06	525.53	527.22	906.59	905.04
mod_R209	897.23	699.98	1275.34	1150.37	1287.35	1227.75
mod_R210	443.91	192.78	916.50	688.21	1152.73	1097.94
mod_R211	189.31	187.92	774.29	705.64	1142.88	1078.17
mod_RC101	2042.78	1244.43	1678.16	1297.34	1313.02	1292.98
mod_RC102	2080.31	1377.90	1697.57	1373.82	1351.33	1298.85
mod_RC103	1870.34	1194.02	1333.66	1260.82	1316.96	1260.20
mod_RC104	1339.20	820.99	1449.76	1088.23	1154.90	1158.55
mod_RC105	1550.86	867.94	1383.61	1134.74	1292.36	1182.41
mod_RC106	2055.95	1163.55	1458.82	1309.28	1330.42	1367.50
mod_RC107	1069.83	832.99	1352.84	1071.98	1245.33	1190.89
mod_RC108	1298.80	819.37	1195.23	1044.72	1185.02	1141.00
mod_RC201	2190.02	1511.51	2494.78	1878.04	1707.43	1613.78
mod_RC202	1327.93	698.20	1835.51	1344.06	1619.51	1591.34
mod_RC203	885.48	444.41	1391.22	925.98	1344.05	1233.86
mod_RC204	323.47	194.24	1032.68	1017.49	1222.07	1032.53
mod_RC205	1688.51	865.65	1753.79	1318.67	1652.15	1516.37
mod_RC206	2708.10	1849.26	2495.85	1911.00	1716.05	1558.13
mod_RC207	1666.46	1114.14	1777.96	1370.56	1564.70	1525.96
mod_RC208	903.15	603.96	1246.79	1100.41	1386.26	1378.47