SOLVING THE MULTI-PERIOD VEHICLE ROUTING PROBLEM WITH TIME WINDOWS AND DELAYED SHIPMENTS USING DELAY ROUTES

SOO KUO YANG

MASTER OF COMPUTER SCIENCE

FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN
JULY 2013

SOLVING THE MULTI-PERIOD VEHICLE ROUTING PROBLEM WITH TIME WINDOWS AND DELAYED SHIPMENTS USING DELAY ROUTES

By

SOO KUO YANG

A thesis submitted to the
Department of Internet Engineering and Computer Science,
Faculty of Engineering and Science,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Computer Science
July 2013

ABSTRACT

SOLVING THE MULTI-PERIOD VEHICLE ROUTING PROBLEM WITH TIME WINDOWS AND DELAYED SHIPMENTS USING DELAY ROUTES

Soo Kuo Yang

This thesis presents a logistic real world problem which has not been addressed in traditional vehicle routing problem with time windows. Here, we present a new constraint to the problem known as clustering constraint. This constraint states that the planner has a need to plan all customers' stops within the given time windows but only a selective number of them can be completed on time for a day. The others have to be postponed. With such constraint in place, typically the planner would choose to postpone orders in the same cluster of areas. So here, we look at the problem across the temporal dimension as well where conventional VRPTW only looks at the spatial dimension of the problem. A typical logistics company would normally have trouble planning trips when they are overloaded with customers' orders that they simply cannot finish all of them today and would have to decide to leave some of them for tomorrow's trip. This thesis presents a methodology to solve this problem by introducing a delay route concept to help forecast the pattern of tomorrow's trip in order to best select the orders to fit into today's trip and also the orders to leave out from today's trip to be postponed until tomorrow. The experiment's focus here is to show the effectiveness of using delay route on an industry accepted algorithm used widely to solve daily logistics VRPTW problem. The main focus of this thesis is on presenting the clustering constraint problem and proposing a simple and yet reusable approach in solving the problem which is to use a delay route approach. To prove this concept, an algorithm that is capable of solving VRPTW effectively is presented as well. As a result, a new way of testing VRPTW solvers is presented with the use of multi day data from the conventional benchmark.

ACKNOWLEDGEMENTS

I would like to express my gratitude to the University, Universiti Tunku Abdul Rahman (UTAR) for enrolling my research for this study and also Quintiq for approving the use of their sole proprietary algorithm for the purpose of this study. My gratitude extends to my supervisor Dr. Tay Yong Haur for his continuous support and patience in guiding me to finally complete the research work done here.

I would also like to take this opportunity to thank Victor Allis, CEO of Quintiq for coming up with the idea of delay route and for his guidance to how we could prove the effectiveness of the idea for a particular customer who came up with the clustering constraint requirements.

Also, heart-felt thanks to Wim Nuijten, VP of optimization technology of Quintiq for his publications on record breaking story and approval for the high-level disclosure on some of the logic used in Path Optimization Algorithm from Quintiq which are sole proprietary.

Also, special thanks to Kasper Kisjes who shared his findings and his model in Sequential Insertion Heuristics algorithm to be used to generate good start solution for VRPTW solvers.

Additionally, I would like to thank the authors of the reference lists of papers I've referred to in this thesis. Without their work I would have never been able to complete mine.

Also, to my wife and daughter for their moral support that motivates me to not give up on this and continue to pursue this until the very end.

Last but not least I would like to thank the examiners and all the board of director for Institute of Postgraduate Studies & Research department of Universiti Tunku Abdul Rahman for their patience and continuous support in my work until completion.

APPROVAL SHEET

This dissertation/thesis entitled "**SOLVING THE MULTI-PERIOD VEHICLE ROUTING PROBLEM WITH TIME WINDOWS AND DELAYED SHIPMENTS USING DELAY ROUTES"** was prepared by SOO KUO YANG and submitted as partial fulfillment of the requirements for the degree of Master of Computer Science at Universiti Tunku Abdul Rahman.

Approved by:

_____

(Assoc. Prof. Dr. Tay Yong Haur)
Date:……..
Supervisor
Department of Internet Engineering and Computer Science
Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

FACULTY OF ENGINEERING AND SCIENCE

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 23 JULY 2013

SUBMISSION OF THESIS

It is hereby certified ***Raymond K. Y. Soo*** (ID No: ***06UIM02020***) has completed this thesis entitled "Solving The Multi-Period Vehicle Routing Problem With Time Windows And Delayed Shipments Using Delay Routes" under the supervision of Dr. Tay Yong Haur from the Department of Internet Engineering and Computer Science, Faculty of Engineering and Science.

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

Yours truly,

_____

(*Raymond K. Y. Soo*)

DECLARATION

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

Name ____Soo Kuo Yang_____

Date _____23 July 2013_____

TABLE OF CONTENTS

		Page
ABSTRACT		**ii**
ACKNOWLEDGEMENTS		**iii**
APPROVAL SHEET		**iv**
SUBMISSION SHEET		**v**
DECLARATION		**vi**
TABLE OF CONTENTS		**vii**
LIST OF TABLES		**ix**
LIST OF FIGURES		**x**
LIST OF ABBREVIATIONS		**xi**
CHAPTER		
1.0	**INTRODUCTION**	**1**
1.1	Motivation	1
1.2	Objectives	2
1.3	Outline	2
2.0	**LITERATURE REVIEW**	**5**
2.1	History of VRPTW	5
2.2	Why is it hard?	7
2.3	How to represent it mathematically?	9
2.4	Algorithms for standard VRPTW solver	11
3.0	**THE MULTI-PERIOD VRPTW**	**13**
3.1	Why multi-period VRPTW?	13
3.2	What is clustering constraint?	14
3.3	Adding delay routes to VRPTW	15
4.0	**APPROACH AND ALGORITHMS**	**17**
4.1	Selected approach to prove delay route concept	17
4.2	Selected algorithm to be used for standard VRPTW solver	18
4.3	Sequential Insertion Heuristics	19
4.4	Path Optimization Algorithm	26
4.5	Summary	29
5.0	**DATA PREPARATION**	**30**
5.1	Initial research work with Solomon R207	30

5.2 Gehring and Homberger's extended VRPTW benchmark 32
5.3 Multiday data for clustering constraint 33

6.0 EXPERIMENTAL RESULTS AND ANALYSIS 36
6.1 Convergence test 36
6.2 Conventional VRPTW solver test 38
6.3 Delay route VRPTW solver test with delay penalty = 1000 41
6.4 Comparison between conventional VRPTW vs. Delay route VRPTW 42
6.5 Additional runs for delay route VRPTW with different penalties 48
6.6 Comparison between all runs 54

7.0 CONCLUSIONS AND FUTURE WORKS 57
7.1 Conclusions 57
7.2 Future works 58

REFERENCES 60

LIST OF TABLES

Table		Page
6.1	Results of convergence test for algorithm used in C1_2_1	36
6.2	Results of day 1 using conventional VRPTW solver	39
6.3	Results of day 2 using conventional VRPTW solver	40
6.4	Results of day 1 using delay route VRPTW	41
6.5	Results of day 2 using delay route VRPTW	42
6.6	Comparison between the results	43
6.7	Results of day 1 using delay route VRPTW with 2000 penalty	48
6.8	Results of day 2 using delay route VRPTW with 2000 penalty	49
6.9	Results of day 1 using delay route VRPTW with 500 penalty	52
6.10	Results of day 2 using delay route VRPTW with 500 penalty	52
6.11	Comparisons between all the results	55
6.12	Further analysis on the comparisons	55

LIST OF FIGURES

Figures		Page
2.1	A diagram displaying a routing solution	6
2.2	A diagram to demonstrate the time window limit	10
4.1	Steps in high level for approach used to build VRPTW solver	19
4.2	Round 1 of Sequential Insertion Heuristics	20
4.3	Round 2 of Sequential Insertion Heuristics	21
4.4	Round 3 of Sequential Insertion Heuristics	22
4.5	Round 4 of Sequential Insertion Heuristics	23
4.6	POA's actions flow chart	26
5.1	The pattern of R207 Solomon VRPTW benchmark	31
5.2	The pattern of C1_2_1 Gehring and Homberger's extended VRPTW benchmark	32
6.1	Day 1 of best solution for conventional VRPTW, run#3	44
6.2	Day 2 of best solution for conventional VRPTW, run#3	45
6.3	Day 1 of best solution for delay route VRPTW (penalty=1000), run#20	46
6.4	Day 2 of best solution for delay route VRPTW (penalty=1000), run#20	47
6.5	Day 1 of best solution for delay route VRPTW (penalty=2000), run#23	50
6.6	Day 2 of best solution for delay route VRPTW (penalty=2000), run#23	51
6.7	Day 1 of best solution for delay route VRPTW (penalty=500), run#9	53
6.8	Day 2 of best solution for delay route VRPTW (penalty=500), run#9	54

LIST OF ABBREVIATIONS

VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Window constraint
POA	Path Optimization Algorithm
API	Application Programming Interface
2OPT	A local search operator that performs a swap of two nodes to untangle the part of a route that crosses. (Croes, 1958)
RTDVRP	Real-time dynamic vehicle routing problem
VRPPD	Vehicle routing problem with Pickup and Delivery
PDP	Pickup and Delivery Problem

CHAPTER 1.0

INTRODUCTION

1.1 Motivation

In logistics industry it is a very common case for a very large logistics company, when the demand of customers' deliveries can never be fully committed and delivered on time in a same day. Some orders has to be postponed and delivered a day or two later. This is especially true when the orders are far away from the depot. In current practice and research, this problem has not been deeply looked into with a good generic approach to solve it. A conventional VRPTW solver does not effectively selects good orders to postpone to the next day. An experienced logistics planner would call this a cluster selection constraint or clustering constraint. This is because typically based on their domain experience; they would choose to postpone a group of orders that are geographically clustered in an area so their fleet of vehicles do not need to go to the same area in two days. This will save some travel distance overall across two days of costs measurement. This is the main motivation for our work here. Thus, we want to propose a generic solution to the problem without the need of making too complex design of algorithms which is to simply add a delay route to the conventional VRPTW solver to forecast the pattern of the travelling that is intended for the next day.

1.2 Objectives

The objectives here can be categorized into three:

Firstly, it is to present the clustering constraint to VRPTW to show case the real world problem of logistics companies that have too many orders that needs to be delivered on a day and thus have the need to postpone some to the next day. They would prefer to postpone a cluster of orders instead of randomly scattered group of orders.

Secondly, it is to present a generic approach to solve the problem of clustering constraints in VRPTW known as the delay route approach.

Third, it is to simulate the problem of having too many orders to deliver on the same day that some has to be postponed for the next day. Then, the next step is to experiment with two approaches to solving the problem to prove the effectiveness of the delay route approach.

1.3 Outline

The structure of the thesis will be separated as follows:

Literature review: We discuss and look at the few previous problem transformations that can be read from the literature to our relevant problem statement. In order to achieve the third objective, we need to build an effective conventional VRPTW solver to begin with. Therefore, we look at a few types of algorithm used and disclose slightly in high-level how the chosen algorithm and approach known as sequential insertion heuristics and path optimization algorithm (POA) for our experiment works but it’s a sole proprietary

algorithm by Quintiq, (a software solution provider for advanced planning and scheduling industry) that cannot be fully disclosed.

The multi period VRPTW and clustering constraint: In this chapter we describe the multi period VRPTW variant and the clustering constraint which is in-line with our first objective. Also discussed here is the approach that was proposed to solve this new constraint found.

Approach and algorithms: In this chapter, we describe the selected approach to prove the delay route concept. Then, we take a deeper look into how the selected algorithm which combines Sequential Insertion Heuristics and Path Optimization Algorithm (POA) works. Our POA implementation uses Random Nearest Neighbour construction technique with a 2OPT local search operator and some other destruction techniques. This chapter mainly describes about the approach and algorithm chosen as our standard VRPTW solver.

Data preparation: Next, we look into how we prepare the data for our experiments. We explain how we had chosen the set of dataset from the huge data benchmark from Solomon VRPTW benchmarks and why it is suitable for our studies. Also disclosed in this chapter is how we set up the experiments for proving this concept.

Experimental results and analysis: We report a few experimental steps we designed before we ensure our conclusion in the studies is not biased or null and void due to whatever invalidity of the decision we took in choosing the dataset from benchmark and also in choosing the algorithm to use. Then, we report the results of the experiments.

Conclusions and future works: We conclude and analytically translate the results of our experiments for this research. We also highlight the contributions of our work followed by a list of possible extensions for further studies and other researcher's path to continue the study of this new transformation of the VRPTW and study the methodology of Delay Routes.

CHAPTER 2.0

LITERATURE REVIEW

2.1 History of VRPTW

In earlier academic research, the problem statement of Vehicle Routing Problem (VRP) is commonly and actively research in many fields of optimization algorithms including those motivated by Artificial Intelligence and Operation Research. (Vladimir & Tarek, 2002) presented a clear story of how Vehicle Routing Problem was explained. Based from their explanation in Vehicle Routing Problem with Time Windows (Vladimir & Tarek, 2002) VRPTW is an extension of VRP with time window constraints which makes the problem more realistic and yet more complex to be solved. Vehicle routing problem (VRP) was first introduced with a problem definition which is the simplest variant with a goal to optimize and some side constraints like the following:

a) Each customer is visited exactly once.

b) All routes start and end at the depot.

The goal was originally just to minimize the total travelling distance of the route required to service all locations/customers. It's actually a travelling salesman problem only in a different context but with a fleet of vehicles instead of a single salesman. So we have orders which we need to plan on a

route, and we want to optimize the travel distance of the routes we are going to build.

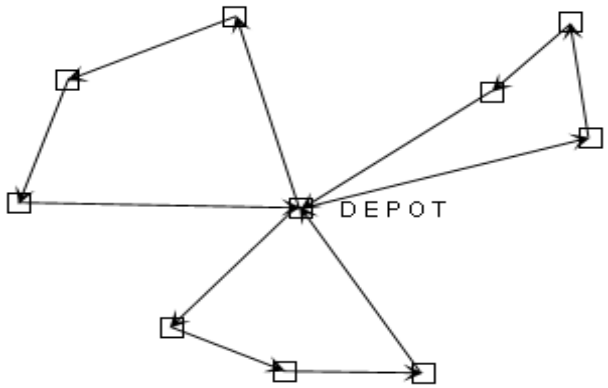

Figure 2.1: A diagram displaying a routing solution.

Figure 2.1 shows a classic example of a solution of the vehicle routing problem. The depot lies in the centre of the Cartesian grid. The orders are scattered around the depot in different directions. Vehicles are assumed to start from the depot to pick up or deliver orders that are scattered around the grid. The example in the figure here shows three separate routes that are built where each handles three different stops or orders. Eventually, the routes are assumed to always return to the depot. This may not always be the case but for the context of this thesis, we will assume that this is always true.

A more relevant variant was named capacitated vehicle routing problem where the routes are limited to the capacity it is allowed to carry, so the number of orders allowed per route will be limited instead of infinite. This makes the problem closer to the real-world scenario of logistics distribution and transportation industry. However, often there are different types of logistics industry which has pickups and deliveries at different locations. Some researchers name this type of variant the Vehicle routing problem with Pickup and Delivery (VRPPD) or some simply name it Pickup and Delivery problem (PDP) whereby the goal is to find optimal routes for a fleet of

vehicles to visit the pickup and drop-off locations. In this thesis, we will discuss only on pure distribution variant of the problem thus only original Vehicle routing problem (VRP) where the assumption is that all orders are picked up in the same depot or distribution centre and eventually only sent to deliveries stops. Typically a customer of the logistics company would require the order to be delivered within a certain preferred time. We call this time period the time window and thus the original VRP problem is transformed into vehicle routing problem with time windows (VRPTW). We will now focus only on this type of variant from the problem domain.

2.2 Why is it hard?

Firstly, to quote from (Wikipedia, n.d.) a description of VRP to the layman observer, the vehicle routing problem does not seem to be very difficult to be solved. Surely, it is just a case of trying all combinations of visits and seeing which one is best to say the least? And the goal thus would be one of these "shortest/fastest/cheapest/etc".

(MJC^2, n.d.) had documented a clear description and nice online presentation with a clear example to present why is VRP difficult to solve. In order for us to demonstrate just how difficult vehicle routing problems can be, imagine a vehicle that has to deliver to 3 different locations X, Y and Z. The problem is to decide which order the vehicle should visit each location to minimise the overall travel distance. There are 6 possible solutions:

X followed by Y followed by Z

Or X folloed by Z followed by Y

Or Y followed by X followed by Z

Or Y followed by Z followed by X

Or Z followed by X followed by Y

Or Z followed by Y followed by X

So, the simplistic approach is to consider all 6 cases, work out the distance travelled for each one and choose the shortest. Actually, often only 3 of the 6 cases need to be considered assuming the distance X->Y->Z is probably going to be the same as the distance for Z->Y->X, unless one-way streets or non symmetric distances are involved.

This sample simpler variant of the problem would take a modern computer almost no time to solve. However, the complexity increases extremely quickly as the number of stops increases:

4 stops would have 24 possible solutions

5 stops would have 120 possible solutions

6 stops would have 720 possible solutions

7 stops would have 5040 possible solutions and so forth

.

.

N stops have N x (N-1) x (N-2) x .... 3 x 2 x 1 solutions. This is known as a factorial dependence. For a route which might make hundreds of stops for deliveries in a day the number of possible routes/sequences is very large. Also, to multiply that further with the number of routes we have in the problem it becomes even larger. It gets even when it increases to thousands of stops, it can be as large as the number of atoms in the universe. The typical computation is simply too large for computer to compute within a reasonable

amount of time for planners to make a planning for the day in a typical logistics company's delivery plan per day.

To add to the complexity, we now are having more real world constraints to the original VRP and thus the pruning of the possible solution requires computation power as well. So not all sequencing is actually possible as time window and capacity needs to be computed accumulatively depending on the previous sequences choices made per route.

2.3 How to represent it mathematically?

To look into detail on the problem formulation of the puzzle we are trying to solve, we first look at how (Li & Lim, 2003) defined the formulation of the original VRPTW variation of the problem. A VRPTW problem can be viewed as a digraph of nodes V and arcs A.

Let $V = \{v_0, v_1, v_2, \ldots v_n\}$ where v_0 denotes the depot and $v_1, v_2, \ldots v_n$ denotes the stops for customer 1, 2 and so on until n-th number of customers.

Each node $v_i \in V$ has an associated customer demand q_i ($q_0 = 0$), a service time s_i ($s_0 = 0$) and a service-time window $[e_i, l_i]$ whereby e_i denotes the earliest service time can start for node v_i while l_i denotes the latest it must be serviced or in other words the due date of the order.

For each pair of nodes $\{v_i, v_j\}$ ($i \neq j$, $i, j = 0, 1, 2, \ldots, n$), a non-negative distance d_{ij} and a nonnegative travel time t_{ij} are known. Due to the time window constraints, arcs may not exist between some node pairs. Therefore, the arc set can be defined as

$$A = \{\{v_i, v_j\} | \, v_i, v_j \in V, v_i \neq v_j, \max(e_i, t_{0i}) + s_i + t_{ij} \leq l_j\}$$

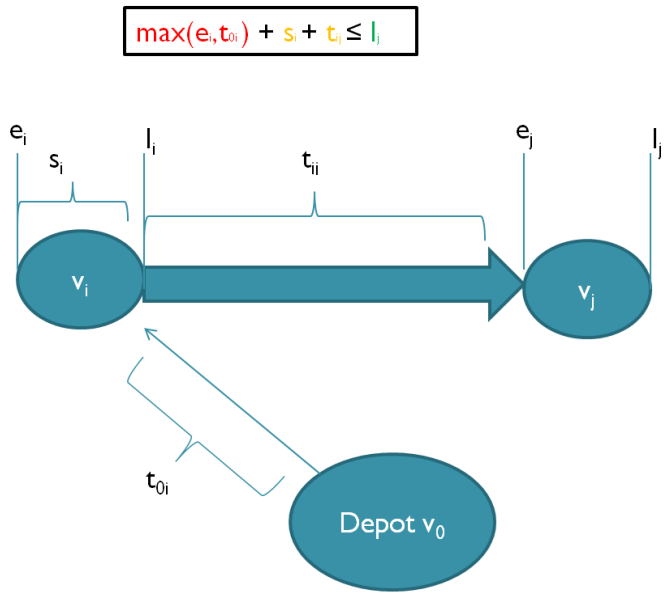

Figure 2.2: A diagram to demonstrate the time window limit.

To better demonstrate the equation, Figure 2.2 helps to show the rationale behind that equation. Assuming v_0 is the depot, t_{0i} is therefore the travelling from the depot to the 1st node v_i. The arc going from 1st node v_i to 2nd node v_j only exists when $\max(e_i, t_{0i}) + s_i + t_{ij} \leq l_j$ where the first term in the left hand side of the equation obtains the latest time of whichever is later; whether we travel from depot to 1st node v_i thus the t_{0i} is later or the earliest start of node e_i itself is later. The next term is the service time s_i itself for the 1st node v_i. The last term is the travel time of going from 1st node v_i to 2nd node v_j. If the sum of all these terms results in a later time than the latest end window or the due date l_j of the 2nd node v_j, then this arc definitely need not to exists as 2nd node v_j would therefore never be on time if such arc is selected in the solution. If a vehicle reaches a customer v_i before e_i, it needs to wait until e_i in order to service the customer. The schedule time of a route is the sum of the waiting time, the service time and the travel time. The objective of the VRPTW is to service all customers without violating vehicle capacity constraints and time window constraints with a minimum number of vehicles

and, for the same number of routes with the minimum travel distance, followed by the minimum schedule time and the minimum waiting time.

2.4 Algorithms for standard VRPTW solver

A wide range of algorithms has been researched to solve the conventional VRPTW problem.

Some used simulated annealing (Li & Lim, 2003) and (Yiqing & Xiao, 2007), some opt for Ant Colony Optimization a.k.a Ant System (Bullnheimer & Strauss, 1997) while others used Genetic Algorithms (Qili, 2000) according to a preliminary survey done in a survey of VRPTW progress paper (Soo & Tay, 2009).

A thesis (Kisjes, 2012) also reported various construction techniques used for generating efficient start solution and their performances for some large scale VRPTW problems. There, they included construction techniques like Sequential Insertion Heuristics, Savings Algorithm, Nearest Neighbour, etc. From the conclusion of the research done there, the technique known as Sequential Insertion Heuristics seems to give rather promising start solution to the search problem of a standard VRPTW.

Sequential Insertion Heuristics is a construction technique widely used for constructing a good start solution for VRPTW problem statements. Quintiq, an organization specialized in advance planning and scheduling software has conducted a research competition effort to find how well the algorithm experts there can solve a large VRPTW benchmark instance from the Gehring and Homberger benchmarks. It was found that techniques using

Sequential Insertion Heuristics combined with Path Optimization Algorithm is able to get to world best known results of certain Gehring and Homberger extension (Gehring & Homberger, 1999) of Solomon benchmark (Solomon, 1983) problem for VRPTW. Solomon benchmark set is a well-known set of benchmark dataset that researchers used for evaluating their algorithms and approaches to solving VRPTW. Quintiq (Quintiq, 2012) managed to publish some new best known solutions as a result of that study using some similar approaches.

Path Optimization Algorithm is a set of API available to help solve problems which can be represented as sequencing a set of nodes on a set of paths. In that competition the Sequential Insertion Heuristics is a specially customized version developed by (Kisjes, 2012) specially designed for clustered based VRPTW benchmark datasets from the Gehring and Homberger's extension of Solomon benchmarks. An example would be the C1_10_4 problem instance of benchmark datasets.

Thus, in this research we decided to utilize similar approach in solving a particular Solomon benchmark. This approach uses the new Sequential Insertion Heuristics developed by (Kisjes, 2012) for generating an initial valid solution for the problem. Then we use Path Optimization Algorithm (POA) which is a built-in tool in the Quintiq software base that provides quick and flexible tuning of some local search actions such as destructing a few nodes in the solution and re-constructing them based on certain heuristics. A node here represents a customer or a stop defined in the VRPTW problem statement.

CHAPTER 3.0

THE MULTI-PERIOD VRPTW

3.1 Why Multi-period VRPTW?

Traditional VRPTW was defined without a temporal dimension. Here we present a multi period variant of the VRPTW which was seen in a practical logistics industry business. In a normal logistic business, the problem of having too many orders to be delivered in a particular day or the problem of having some trucks break down and you cannot delivered all orders as planned happens quite often. Conventional VRPTW solver's approach solved the problem statement ignoring the link between the multi period data across multiple days of data. It does not take into account the temporal dimension across days but rather the time windows within a certain day. In the real life environment, you do not get tomorrow's data today. Note that the term "today" here means the day we are planning the data for; not exactly meaning the exact day as of when the planning was to be made. For example we are planning orders for Tuesday when we are in a Sunday, meaning we plan 2 days up front always. So "today" in this context means the data for Tuesday and "tomorrow" means Wednesday. To keep the problem statement simple here, we only look at 2 days of data. Today and tomorrow's data. Typically, we do not get tomorrow's data today. According to the example used, on Sunday we do not get data for Wednesday yet but we only get data for

Tuesday. So we can make a planning for Tuesday but not yet for Wednesday. A conventional VRPTW solver will just solve Tuesday's data and optimizing locally based on data received on Tuesday. When we are in Monday, data for Wednesday comes in and then we need optimize again based on the data then. However, if Tuesday is a peak day or a peak holiday season where you get a lot of delivery orders. We may not be able to deliver all orders for Tuesday and may need to postpone some order to Wednesday. This was the link across multi days of the problem statement that was not considered in conventional VRPTW solvers.

3.2 What is clustering constraint?

Clustering constraint is when a logistic company has too many orders to deliver on a particular day that it has no choice but to postpone some orders to deliver on the next day. When this happens, typically, a logistics planner would plan to postpone orders based on some heuristics from their domain knowledge. A good example would be based on geographical information of those orders. So if the orders can be grouped in clusters, an experienced logistics planner would probably choose to postpone orders in clusters instead of selecting those scattering around in a number of different clusters to make efficient routing. In a real world typical logistics delivery service provider, there can be a lot of other dimensions that they also may take into account instead of purely looking at clusters of areas certain orders belong to. For example they may consider priority customers that paid a premium price for the delivery to have their orders delivered first instead of others. In this

research effort we make an assumption to ignore this priority favouring constraint for a start to avoid complicating the problem statements. We first want to be able to solve the constraint of wanting to deliver orders efficiently today per clusters and also for selecting best sets of orders to postpone for tomorrow so we can choose to postpone similar orders that are belonging to certain cluster of area together instead of randomly selecting orders to postpone.

3.3 Adding delay routes to VRPTW

In our earlier published work (Soo & Tay, 2011) a solution approach was proposed to solve this constraint which is by simply adding a delay route to the conventional VRPTW solver regardless of what algorithm used. Similar to the new approach of adding delay routes in VRPTW in our earlier work, we also introduced a new variable p_i for a particular node v_i to introduce additional penalty scores if any customers are not serviced by the actual routes but by the delay routes. Therefore if a node v_i is reached by a delay route vehicle then it will increment p_i by a certain value to be experimented with later. The new objective is therefore having the sum of this new term p_i to be minimized as well.

In order to allow us to experiment with this approach, we need to have an effective VRPTW solver to begin with. Only with that, can we then add the delay route to it and begin experimenting with the penalty and to compare its effectiveness against without having it. The plan was to find a good VRPTW solver first and then find a good benchmark to experiment with. After having

that, we need to modify the benchmark to make it a 2 days dataset. Next, we can first compare how the conventional VRPTW solver performs across 2 days and then eventually experiment with adding delay routes to it and see if it improves the measurements. After that, we can still fine tune the delay penalty to experiment further to see what is the best value to assign to for optimum performance.

CHAPTER 4.0

APPROACH AND ALGORITHMS

4.1 Selected approach to prove delay route concept

To achieve the third objective mentioned earlier in introduction, we first use a standard VRPTW solver methodology to just run the algorithm for the 2 days of orders separately and see what's the total distance travelled. Then, we introduce a new methodology that is generic to be easily applied to any VRPTW solver known as a delay route methodology that we hypotheses to be able to better select orders to be postpone compared to conventional standard VRPTW solver methodology. This is the approach mentioned in the second objective. The approach of delay route was proven successful in a specific Quintiq project for a certain customer. It was done initially with the customer's data which is business specific. Unfortunately, due to non-disclosure nature of the business we are not allowed to disclose any data publicly. The aim here is to find out if such approach can be generic and also can be proven to be effective on academic benchmark datasets as well like the Solomon as well as the Gehring and Homberger benchmarks. With this approach, we can finally achieve all the three objectives mentioned earlier in the introduction chapter of this thesis. In order for us to be able to test the approach of delay route, we first need to build a standard VRPTW solver that can effectively solves standard VRPTW benchmark datasets.

4.2 Selected algorithm to be used for standard VRPTW solver

This section will focus and describe how we developed and build our base standard VRPTW solver for that purpose. We first used a new sequential insertion heuristics that was designed by (Kisjes, 2012) during the study for Quintiq's research for C1_10_4 which is an instance from the Gehring and Homberger extension of the original Solomon benchmark for VRPTW problems. C1_10_4 is a problem instance with 1000 orders and 200 routes. Its best known solution was recently recorded by Quintiq (Quintiq, 2012). Quintiq organized an internal competition for its algorithm experts to attempt to solve this problem instance in the most efficient way. Kisjes's design was not the winner of the competition but was widely used by participants and was also used by the 2nd place winner whom had also set a new best known from the previously published record but only a few points worse than the absolute best solution found throughout the whole competition. After we used Sequential Insertion Heuristics to generate a start solution, we used the Path Optimization Algorithm (POA) which is a sole proprietary technology from Quintiq which contains a rich set of API for local search operators tailored-made for similar problems to these. It can be used to solve any problems which can be represented as sequencing optimally a sequence of nodes on paths. The following diagram shows in high level the approach steps we took in attempt to build a valid VRPTW solver as our first step for our experiment:

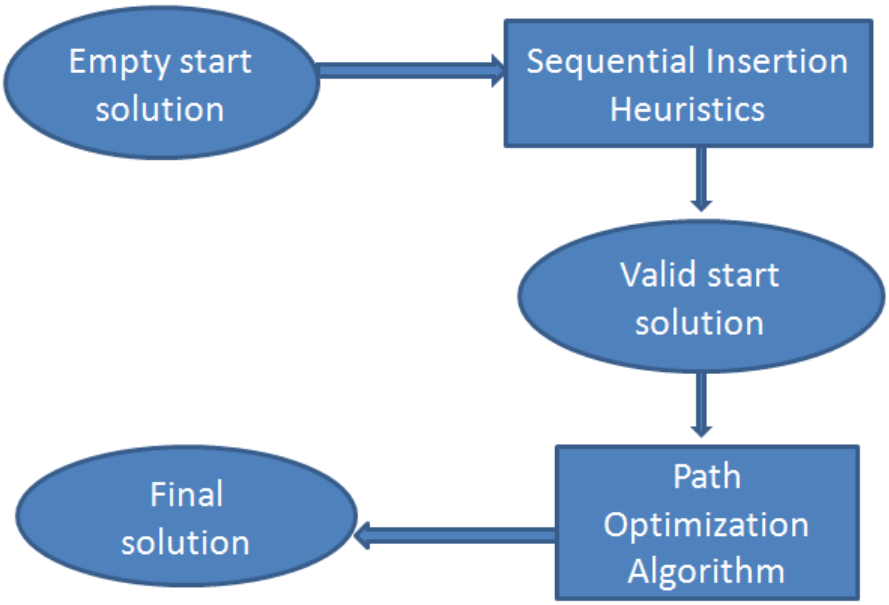

Figure 4.1: Steps in high level for approach used to build VRPTW solver.

Figure 4.1 illustrates how our approach for solving VRPTW works. First we start from an empty state where all orders are not planned on any routes. We use sequential insertion heuristics which builds a valid initial start solution one route by one route, then pass in this solution into POA to perform further improvements on it before reaching the final solution. Next, we look into details on the two selected algorithms described which is the sequential insertion heuristics followed by path optimization algorithm. Note that the latter is a sole proprietary algorithm component of Quintiq and thus cannot be disclosed in too detailed way.

4.3 Sequential Insertion Heuristics

The sequential insertion heuristic was made popular by Solomon and is widely used for creating decent starting solutions for meta-heuristics for

vehicle routing problems. In Kisjes's implementation, the heuristic creates one route at a time, in four rounds. Firstly, the problem is clustered into clusters of orders. Orders are considered to belong to the same cluster if the distance between them is within a certain amount of distance unit. This is measured by looking at the 2D map which plots the x, y coordinate of the orders, and then calculating a nice threshold to nicely cluster them into groups that are obviously apart from one another. For our case, this value was assumed to be 15 after we did some analysis on the data that we are using to experiment with. First round: all clusters with exactly the sum of demands equivalent to the maximum capacity of the routes. For the problem instances of the benchmark for VRPTW that we are experimenting, these sum up to be exactly 200 capacities. These can probably be efficiently combined into one route.

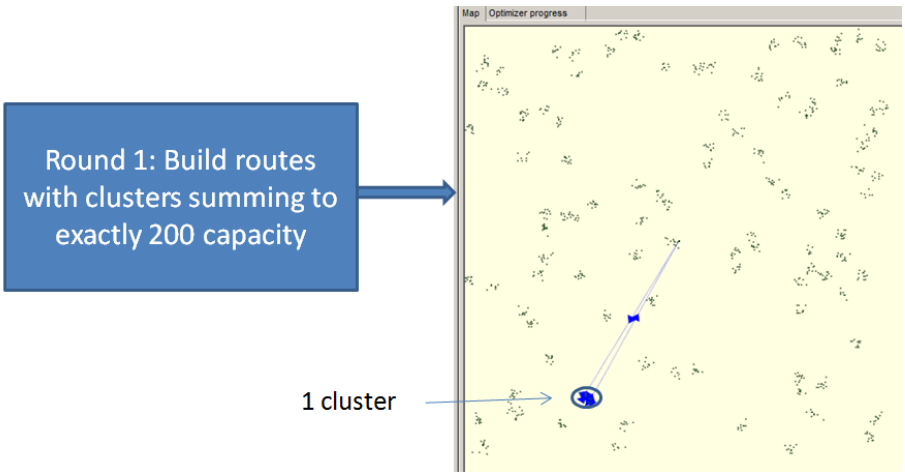

Figure 4.2: Round 1 of Sequential Insertion Heuristics.

In Figure 4.2, we can see an example of a route that was build in round 1 of the sequential insertion heuristic. In this round, Phase1: Tries to insert as much of orders within the same cluster into one route. This route travels to only one cluster as the cluster will have a sum of capacity 200 only by having the orders within it planned.

Second round: all clusters with exactly the sum of demands filling up 95% of the routes' capacity. For our case here they need to sum up to be exactly 190 demands. Same story with first round here, but now they require a pickup with demand size of 10 to fill up the 5% balance of the routes' capacities. These are relatively scarce and are best assigned as fillers for route with only 10 slacks.

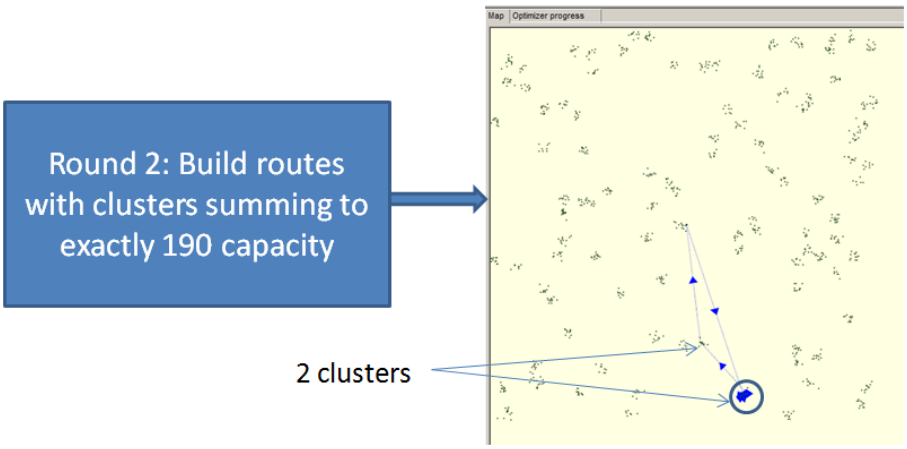

Figure 4.3: Round 2 of Sequential Insertion Heuristics.

In Figure 4.3, we can see an example of a route that was build in round 2 of the sequential insertion heuristic. In this round, Phase1: Tries to insert as much of orders within the same cluster into one route. Phase2: Tries to insert as much as possible other orders while avoiding large detours. This route needs to travel to 2 clusters, one having exactly the sum of capacity 190 but the other fits as an extra order of exact fit with exactly 10 demands to the route.

Then in the third round, new seed routes are selected based on a 'Priority' expression that includes both distance to depot and relative angle to the last seed, filtering out the most centrally located orders).

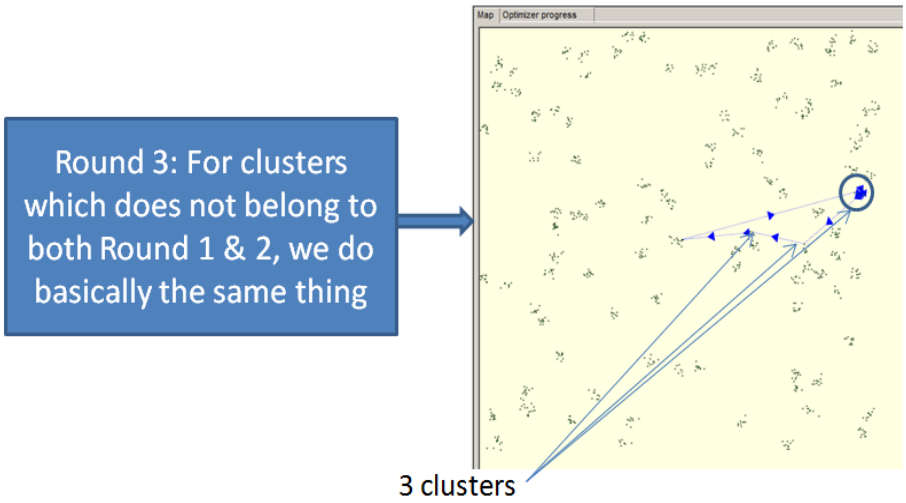

Figure 4.4: Round 3 of Sequential Insertion Heuristics.

In Figure 4.4, we can see an example of a route that was build in round 3 of the sequential insertion heuristic. In this round, Phase1: Tries to insert as much of orders within the same cluster into one route. Phase2: Tries to insert as much as possible other orders while avoiding large detours. This route needs to travel to 3 clusters as the sum of orders' demand from it is not 200 and not 190. The sum of the capacity in the clusters can be either more than 200 or less than 190. When it is more than 200, some orders in the cluster will be left out and the route will need to travel to other clusters as shown in the example to get other their smaller orders. Similarly when it's less than 190, the route also needs to travel to other clusters to pick up their smaller orders that still fit for its capacity.

In the fourth and last round, remaining orders close to the depot are combined into some last routes.

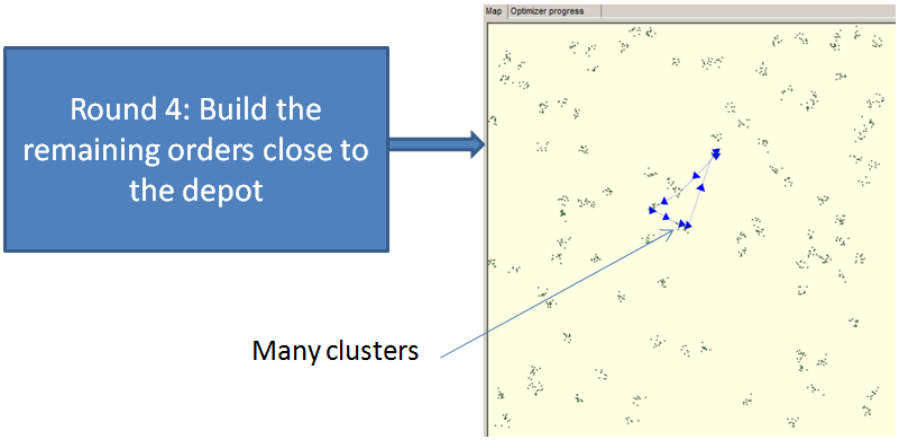

Figure 4.5: Round 4 of Sequential Insertion Heuristics.

In Figure 4.5, we can see an example of a route that was build in round 4 of the sequential insertion heuristic. In this final round, no more clustering priority logic runs so Phase 1 and 2 is no longer needed, thus only Phase 3 is run. Phase3: Select orders close to the depot remaining to construct efficient routes based on heuristics. This route needs to travel to a few clusters but they are all nearby to the depot.

Each time after selecting a new seed (in either round), the seed route is filled up as much as possible. Adding unplanned orders to the route until this is no longer possible. In the first three rounds, a route is constructed during two sequential phases. In phase 1, the heuristic attempts to insert as many as possible of the other orders within the cluster that the seed order belongs to. When this is no longer possible, phase 2 attempts to fill up the route with other orders, avoiding large detours as much as possible. Phase 1 and 2 iterate until the full clockwise sweep has been completed (end of round 3), leaving only some orders close to the depot. Those are assigned to routes in round 4 (construction phase 3 in the implementation). All three phases use different combinations of selection measures and weights.

The heuristic or selection weight and measures or the ‘Priority’ expression is calculated based on:

1. Additional distance (AD) - cost measure, used in all three phases.

2. Distance between the order and the depot (DTD) - static regret measure, used in phase 2 and 3.

3. Effective time window width of the order (TWW) - static regret measure, used in all three phases but with an additional multiplier in phase 1 and 2.

4. Demand/load size of the order (L) - static regret measure, used in phase 1.

5. Exact fit (this order is exactly large enough to fill up the vehicle thus we introduce a bonus preference score to it (EF) - dynamic regret measure, used in all three phases but with more emphasis in phase 2 for filling up the exact 5% balance of the route capacity, 10 in our case.

6. Distance to seed - semi-static regret measure, used in phase 1.

7. Alignedness (relative angle between seed and order) - semi-static regret measure, used in phase 2.

8. Isolatedness (number of unplanned orders within cluster-threshold-distance (here we used 15 distance unit)) - dynamic regret measure, used in phase 1 and 2.

The values for these weights that we used to set up our experiment runs were the following:

AD = 1.0

DTD = 2.0

TWW = 0.15

L = 0.15

EF = 0.5

Distance to seed = 0.15

Alignedness = 0.15

Isolatedness = 1.0

Note that the value suggested and used in this research is completely randomly picked and experimented with for a few times to see which set of values can generate a reasonably good initial solution with no constraint violation thus a valid initial solution for C1_2_1. C1_2_1 is a much smaller problem compared to C1_10_4. It has only 50 routes to plan 200 orders in total as compared to C1_10_4's 200 routes to plan 1000 orders. Both of them have routes capped at 200 maximum capacities too and both of them are clustered data variants from the set of Gehring and Homberger benchmark problems. Kisjes's (Kisjes, 2012) original implementation was intended on C1_10_4 and his design allows tuning of these heuristics to match different problem instance of the benchmark. In our work, we have fine tuned it

preliminary to get a good initial solution for C1_2_1 which is the problem instance we picked to use for the purpose of our study. More details about why C1_2_1 was picked will be discussed in the next chapter of this thesis.

4.4 Path Optimization Algorithm

After Sequential Insertion Heuristics generates a valid initial solution, we run Path Optimization Algorithm (POA) which is a sole proprietary algorithm designed by Quintiq which has a rich set of API for local search operators. The concept behind it is removing a few nodes from the constructed initial solution and then re-constructs them back randomly.

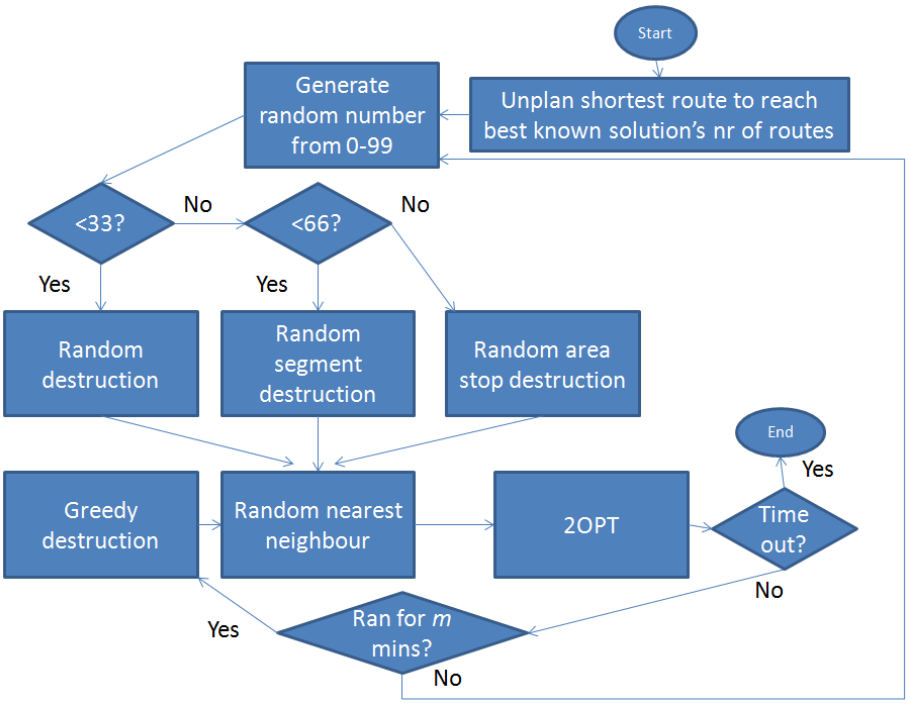

Figure 4.6: POA's actions flow chart.

Figure 4.6 shows a logical flow of the steps of actions took in Path Optimization Algorithm. When received the initial solution from Sequential Insertion Heuristics the first step is to unplan the shortest routes to reach the

number of route used to be equivalent to that of the best known solution (this number is a setting that can be tuned manually). Then, it randomly performs Random destruction, Random Segment destruction or Random Area stop destruction with the same probability. After which, a random nearest neighbour construction takes place followed by a 2OPT and then repeats the destructions again and loops until the end. Periodically, it jumps out and does a greedy destruction instead and calls a fast reconstruction doing a random nearest neighbour as well and followed by a 2OPT. Then again repeats the same steps of destruction and reconstruction until it times out.

To explain in a little more detailed, let's look at each of the actions mentioned in details, the destruction heuristics are simple, here we combined a few of them so we did one third of the time a pure random destruction, one third of the time we do a random segment destruction and remaining one third of the time we do random area stop destruction.

Random destruction – completely randomly selects *x* nodes out of the current solution and unplan them. It was set here to also distribute the destruction of the nodes to be distributed across paths/routes. So this ensures that we do not delete *x* nodes from the same path. For our experiments here we set $x = 3$.

Random segment destruction – randomly pick a "Segment" of the solution defined by a certain expression based on the travel time between the nodes. Each segment contains nodes close to each other, but between the segments there is relative much distance. A threshold of *d* distance unit is used here to define a segment. Also, we need to set *x* number of segments of at most *y*

number of nodes to be destructed. Here it was tuned to have $d = 40$, $x = 3$ and $y = 4$. This means 3 segments of at most 4 nodes each which have 40 distance unit apart from each other will be destructed.

Random area stop destruction – randomly picks an area in the coordinates with a certain predefined radius of d distance unit, and destructs all nodes that are in that area. A threshold of 20 distance unit was used here, so d was set to 20 here.

In the re-construction, a Random Nearest Neighbour construction selects nodes from the destructed pool of nodes, and then constructs them in a sequence based on the distance from the last planned node. It first starts with a completely random selection of the seed node to construct and then selects the nearest neighbour node of this seed node to construct and this newly constructed node will be the reference of the next node and the logic follows until all nodes are constructed.

Following that a 2OPT operator will act on the resulting solution. 2OPT is a basic local search operator that takes a route that crosses over itself and reorder it so that it does not. It was first proposed by Croes in 1958 (Croes, 1958) for solving the travelling salesman problem. All of these are readily available in the POA sets of API.

After POA has run for m minutes, the algorithm then runs a greedy destruction which is not provided in the standard API of POA. Here the logic un-plans any overloaded routes by removing all visits in them, also if any orders are late then they will be unplanned. Next, for visits that requires

travelling more than *d* distance unit, we also un-plan the destination visit. This is continuously iterated until there is no longer any visit that requires travelling longer than *d* distance unit. Similarly, any visits that result in a waiting of *d* time unit will be removed iteratively until there are no such occurrences. Here we set $m = 2$ and $d = 150$, so every 2 minutes of POA run, these greedy destruction will occur to remove travelling and waiting of more than 150 distance unit.

After these greedy destructions, a quick simple POA re-construct is called again to perform random nearest neighbour and 2OPT operator on the solution. Then the standard POA is called again to perform the random destruction and re-construction again. Then the algorithm continues to run for as long as we set it to run.

4.5 Summary

After designing the standard VRPTW solver using the two mentioned algorithm and approach discussed, we are now ready to prepare our data for the experiments. We first need to test if our VRPTW solver is good enough to solve a conventional VRPTW benchmark dataset by checking if it can find the best known solution published thus far. Only then we can be convinced that it is a good VRPTW solver to use to continue to prove our concept of delay route with our selected approach to experiment with 2 days version of the data. The next chapter describes our earlier work done with the delay route concept and also why we feel we need a new approach to justify it followed by how we design this new approach.

CHAPTER 5.0

DATA PREPARATION

5.1 Initial research work with Solomon R207

As mentioned in the literature, Solomon's benchmark (Solomon, 1983) is a widely used common benchmark dataset for solving VRPTW. Thus, it makes perfect sense to reuse such common dataset for the purpose of this study. In an earlier work (Soo & Tay, 2011) we studied the delay route concept on a smaller dataset in the Solomon benchmark R207. R207 is a dataset which has 100 orders and 50 routes to plan them. The best known solution was 890.61 on total travel distance and was planned on only 2 routes. In the study, we duplicated the dataset and tested the delay route concept by adding an additional delay route to study but found out the problem statement has a tight window constraint which causes the solution unable to be solved with just an additional delay route. Thus we needed two delay routes to eventually solve the problem. In that study, we used only POA and did not use Sequential Insertion Heuristics at all thus using only pure Random Nearest Neighbour construction from POA to generate a full initial solution randomly. After finishing the study, we found out the work was not very conclusive and later it was found that a clustered based data will be deemed more suitable for the purpose of studying this concept. R207 was instead a random based data

and thus does not show clearly an example of clustering constraint which was the main motivation for studying the delay route approach in the first place.

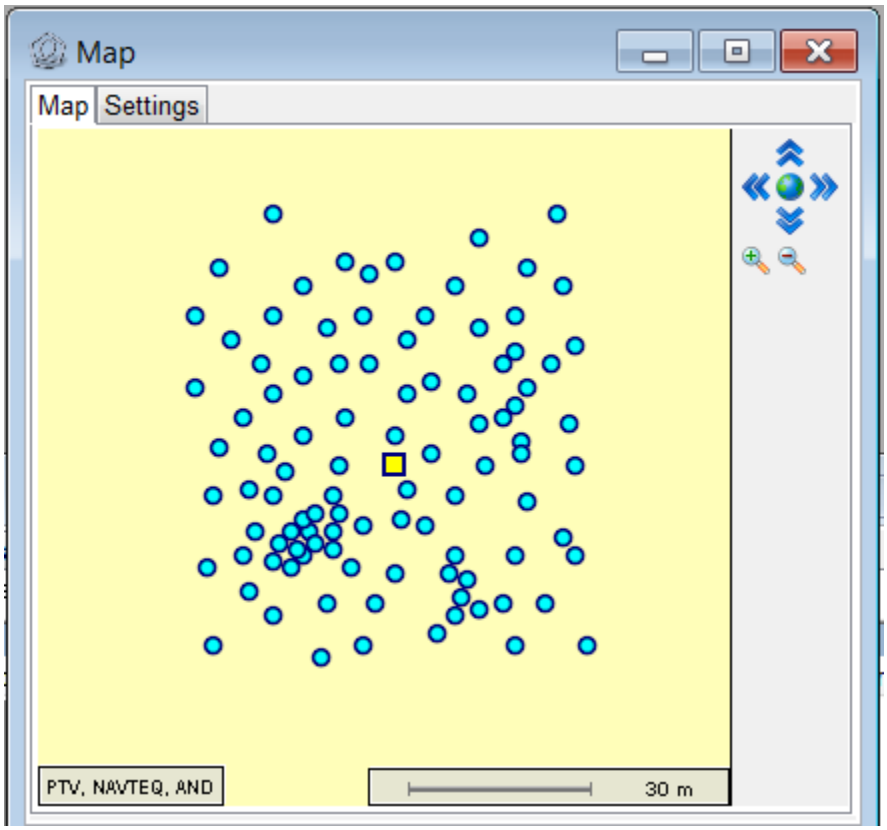

Figure 5.1: The pattern of R207 Solomon VRPTW benchmark.

Figure 5.1 shows how the pattern of R207 dataset looks like, the orders are randomly scattered around the plane and thus is not really reflecting the clustering constraint definition of our problem here. To better present our work we have selected instead C1_2_1 from the Gehring and Homberger's extension of the VRPTW benchmark.

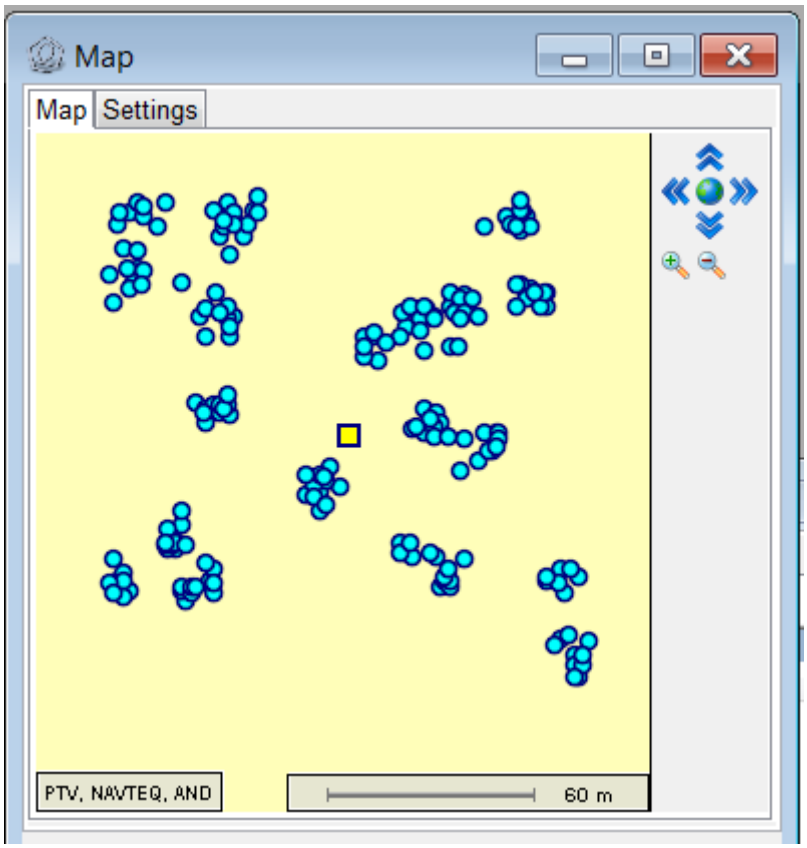

Figure 5.2: The pattern of C1_2_1 in Gehring and Homberger's extended VRPTW benchmark.

Figure 5.2 shows how the pattern of C1_2_1 dataset looks like, this time the orders are clearly clustered and thus suits our first objective to present the clustering constraint problem of the VRPTW problem statement.

5.2 Gehring and Homberger's extended VRPTW benchmark

This time, we have selected C1_2_1 from the instances of Gehring and Homberger's extended VRPTW benchmark (Gehring & Homberger, 1999) to be used for the purpose of this research. C1_2_1 is just a slightly larger dataset than R207 by 100 orders. It also has 50 routes to plan these 200 orders. It was intentionally selected for this study because we do not want one that is too

large of a problem to be solved. This is to avoid unnecessary timely tuning effort needed to ensure we can solve the problem good enough or even get to the best known solution after running for a very long time. We keep the size small so a simple algorithm would be able to solve the problem good enough (for example within less than 1% of the best known published) and in the best case, to be able to converge to the best known solution consistently. Also, it was kept simple so more tweaks and steps can be taken to study a multi day problem statement. Note that this experiment is carried out to study the effectiveness on academic benchmark problem sets to check if the approach is generic enough to be proven with also academic data. The approach was already proven in a Quintiq project for a specific customer when this problem statement was introduced with purely customers' data of 1 week. The delay route approached has been proven to be able to provide much more costs savings for the customer's planning of vehicles and drivers. However sometimes, customer's real-world data is business specific and thus we want to experiment here if the same approach is generic enough that it can also be proven with academic standard benchmark problems dataset. Then this approach can be made generic for future works to be researched on multi day problem statement which is what we want to discuss next.

5.3 Multiday data for clustering constraint

Here we introduce a multi day version of the problem which was seen in a practical logistics industry business. A multi day problem is suitable to display the problem of having too many orders to be delivered in a particular

day or the problem of having some trucks break down and you cannot delivered all orders as planned. This supports our first objective to present the problem of clustering constraint and also our third objective that is to simulate the problem to be experimented with to prove the approach of delay route. The design we used to create a multi day problem with C1_2_1 is as follows:

We had checked in the latest website (Gehring & Homberger, 1999) publishing the Gehring and Homberger benchmark's the current best known solution to the problem C1_2_1. It has a total travel distance of 2704.57 and uses only 20 routes. Since the best solution is known to have utilized 20 routes, we simulate our day 1 data to have only 19 routes so to ensure we do not have enough to perform all the orders and some needs to be postponed. Day 2 data is simulated as a fresh copy of C1_2_1 again but with additional orders that are postponed from day 1. Orders that are postponed from day 1 are assumed to have still the same preferred time window or location opening hours for delivery. For example, if an order is due at 16:00 for day 1, and it was planned late or was not planned by the optimizer in day 1, it will be postponed to day 2 but still having its due set as 16:00 but of day 2. This is simulating a preferred time window for the order or perhaps simulating the location's opening hours per day. If a location is only opened until 16:00 in day 1 it is likely that it also opens only until 16:00 in the next day. In day 2 we try to give as many routes as possible as we want the algorithm to really optimize the result and then we can compare the total distance travelled in both days using the same amount of trucks in total. The motivation behind this study is suppose a logistics company has a truck break down and has to postpone some orders, the conventional VRPTW vs. the delay route VRPTW

is tested to see if the delay route approach can help gives a better selection on which orders to postpone to the next day. In the next day, since there are many orders to be delivered, typically a logistics company would either rent a few extra trucks as back up for this day or outsource some of the work to other companies. In the former case, an improvement of the conventional VRPTW's optimization solution would mean significant savings of costs for the logistic company.

CHAPTER 6.0

EXPERIMENTAL RESULTS AND ANALYSIS

6.1 Convergence test

To ensure our chosen algorithm, which is a combination of sequential insertion heuristics followed by the path optimization algorithm can solve the problem efficiently and most importantly can find the best known solution as published; we first tested it on solely the pure C1_2_1 VRPTW problem statement.

The result of running it for 50 times is as follows:

Table 6.1: Results of convergence test for algorithm used in C1_2_1.

RunNr	ActualTimeRun	Distance
1	00:33:21	2704.57
2	00:33:20	2704.57
3	00:33:20	2704.57
4	00:33:26	2785.63
5	00:33:20	2704.57
6	00:33:20	2805.86
7	00:33:19	2704.57
8	00:33:19	2704.57
9	00:33:20	2704.57
10	00:33:21	2704.57
11	00:33:21	2704.57
12	00:33:20	2704.57
13	00:33:21	2704.57
14	00:33:22	2714.59
15	00:33:21	2704.57
16	00:33:21	2704.57
17	00:33:21	2704.57
18	00:33:21	2704.57
19	00:33:21	2704.57
20	00:33:21	2704.57

21	00:33:21	2704.57
22	00:33:21	2704.57
23	00:33:21	2714.59
24	00:33:21	2704.57
25	00:33:21	2704.57
26	00:33:21	2805.86
27	00:33:21	2704.57
28	00:33:20	2704.57
29	00:33:20	2704.57
30	00:33:21	2704.57
31	00:33:21	2704.57
32	00:33:20	2704.57
33	00:33:19	2704.57
34	00:33:18	2704.57
35	00:33:20	2704.57
36	00:33:19	2807.24
37	00:33:19	2704.57
38	00:33:19	2704.57
39	00:33:16	2704.57
40	00:33:18	2704.57
41	00:33:19	2707.42
42	00:33:17	2704.57
43	00:33:19	2704.57
44	00:33:21	2704.57
45	00:33:20	2704.57
46	00:33:20	2704.57
47	00:33:20	2807.24
48	00:33:20	2704.57
49	00:33:20	2704.57
50	00:33:20	2704.57

On average we can find the best known in less than 34 minutes. In Table 6.1; the first column shows a run number that uniquely identifies the separate runs made. The second column shows the total duration of the run respectively, followed by the goal score or the total travel distance of the final solution of that run. All final solutions of the runs planned all 200 orders on time and used only 20 routes. Thus, all of them are valid solutions and uses the minimum number of routes equivalent to the best known solution which is 20 routes.

The results was promising as out of all these 50 runs, only 8 of the runs did not converge to the best known solution. On average we get a distance of 2715.02 which is less than 0.4% away from the best known 2704.57. Looking in details, most of them converge to best known solution in less than 30 minutes and just get stuck in the final 3 minutes doing nothing. To conclude this study, we are confident that running for 34 minutes of the algorithm is sufficient to get a converged best solution out from the problem statement with a success rate of at least 84%. So this means if we run about 10 times, we should be able to get at least 8 times converged to the best known solution.

Motivated by the success of this first step, we proceed to model the experiment as mentioned in the previous "Data preparation" chapter of the thesis earlier. From the results we observe in this first test, this means we should do the next step for 66 minutes because we need 33 minutes for each day's data. The assumption made here is that this should allow us to converge at not less than 84% success rate.

6.2 Conventional VRPTW solver test

First we ran the conventional VRPTW solver thus utilizing only the algorithm we mentioned which is Sequential Insertion Heuristics for creating a start solution then improving it with Path Optimization Algorithm (POA). To limit the sequential insertion heuristics from generating 20 routes, the algorithm first run with unlimited number of routes until a valid solution is created. Next, it is reduced by selecting the route with least number of orders on them and removes this route from the solution. This is iterated until the

solution has only 19 routes remaining. Note that this step is different from the logic mentioned in the "Approach and algorithms" chapter previously. There, we only reduced the number of routes to the best known solution's number of routes used but instead here, we reduce to 1 route less than that. Then, we proceed directly to POA's destruction and construction steps with a solution that is not yet valid and consists of only 19 routes.

The result of 10 runs of the day 1 data (C1_2_1 with limit of 19 routes only) is shown in Table 6.2:

Table 6.2: Results of day 1 using conventional VRPTW solver.

RunNr	Score	NrOrders	NrOrders Planned	NrOrders OnTime	NrRoutes	NrRoutes Used	NrRoutes WithinCapacity	Distance
1	1502795.70	200	185	185	19	19	19	2795.70
2	902706.29	200	191	191	19	19	19	2706.29
3	1002591.71	200	190	190	19	19	19	2591.71
4	1002696.47	200	190	190	19	19	19	2696.47
5	1202797.99	200	188	188	19	19	19	2797.99
6	1002614.30	200	190	190	19	19	19	2614.30
7	1202673.25	200	188	188	19	19	19	2673.25
8	802763.82	200	192	192	19	19	19	2763.82
9	902896.61	200	191	191	19	19	19	2896.61
10	902615.06	200	191	191	19	19	19	2615.06

First column shows the run number which uniquely identifies each run from one another, second reports the score of the optimizer, third shows the number of orders, followed by those that are planned, and then those that are on time. Next, it shows the number of routes generated at start followed by number of routes used and those that are within the capacity limit. The last column shows the distance travelled in the solution. From the results, it seems that we are not really always converging at a consistent result as the algorithm is still having difficulties in selecting which orders to sacrifice to be postponed to next day. All the orders that are not on time will be postponed to the day 2 run.

We first designed the day 2 data to have only 21 routes, because best known solution consist of 20 routes for C1_2_1 and we thought having 1 extra should be enough to cater for the postponed orders. However from a first few experimental runs we found out 21 is still insufficient. Sometimes more than 10 orders were postponed and thus they may not fit in just 1 extra route. So to make the problem realistic, we added 2 extra routes to ensure day 2 data is a valid VRPTW that can still be solved. Typically that is how a logistic company would have estimated how many extra trucks to rent for the second day anyway.

The result of 10 runs of the day 2 data (C1_2_1 with limit of 22 routes only) is shown in Table 6.3:

Table 6.3: Results of day 2 using conventional VRPTW solver.

RunNr	Score	NrOrders	NrOrders Planned	NrOrders OnTime	NrRoutes	NrRoutes Used	NrRoutes WithinCapacity	Distance	Feasible	nr of late	total distance
1	5008.80	215	215	213	22	22	22	3188.47	violated	2	5984.17
2	3004.85	209	209	209	22	22	22	3004.85	satisfied	0	5711.14
3	2898.02	210	210	210	22	22	22	2898.02	satisfied	0	5489.73
4	2900.37	210	210	210	22	22	22	2900.37	satisfied	0	5596.84
5	6259.84	212	212	209	22	22	22	3298.31	violated	3	6096.30
6	2974.53	210	210	210	22	22	22	2974.53	satisfied	0	5588.83
7	3026.02	212	212	212	22	22	22	3026.02	satisfied	0	5699.27
8	2964.48	208	208	208	22	22	22	2964.48	satisfied	0	5728.30
9	3009.77	209	209	209	22	22	22	3009.77	satisfied	0	5906.38
10	2899.63	209	209	209	22	22	22	2899.63	satisfied	0	5514.69

As the result shown, despite having 2 extra routes on the second day sometimes the conventional VRPTW may still be unable to solve the problem completely and leaves some orders still un-planned. The first and fifth run shows the case when this happens. There are still 2 orders late in the first run

while there were still 3 orders late in the fifth. Instead of trying to increase further the number of routes to use for this day, we proceed to try with the delay route approach first to see if it can consistently solves the problem instead.

6.3 Delay route VRPTW solver test with delay penalty = 1000

Next, we moved on the experiment with the delay route approach. To do this, we did not change anything at all in the Sequential Insertion Heuristics but only in POA we added an extra "virtual" delayed route that is computed in the algorithm for scoring but any nodes planned in this path is not actually planned in the actual result when POA returns the solution. We first experimented this with a delay penalty of 1000 points each time an order is planned on the delay route.

The result of 10 runs of the day 1 data (C1_2_1 with limit of 19 routes only) is shown in Table 6.4:

Table 6.4: Results of day 1 using delay route VRPTW.

RunNr	Score	NrOrders	NrOrders Planned	NrOrders OnTime	NrRoutes	NrRoutes Used	NrRoutes WithinCapacity	Distance
11	802741.93	200	192	192	19	19	19	2741.93
12	1102690.09	200	189	189	19	19	19	2690.09
13	702698.95	200	193	193	19	19	19	2698.95
14	702708.42	200	193	193	19	19	19	2708.42
15	1202828.92	200	188	188	19	19	19	2828.92
16	702684.65	200	193	193	19	19	19	2684.65
17	902780.20	200	191	191	19	19	19	2780.20
18	2002721.05	200	180	180	19	19	19	2721.05
19	702730.32	200	193	193	19	19	19	2730.32
20	702635.22	200	193	193	19	19	19	2635.22

From the result, it shows that even in day 1 data's result alone, having the delay route already postpone less number of orders to the second day. Similarly, for each of those runs we have a corresponding day 2 run.

The result of 10 runs of the day 2 data (C1_2_1 with limit of 22 routes only) is shown in Table 6.5:

Table 6.5: Results of day 2 using delay route VRPTW.

RunNr	Score	NrOrders	NrOrders Planned	NrOrders OnTime	NrRoutes	NrRoutes Used	NrRoutes WithinCapacity	Distance	Feasible	nr of late	total distance
11	2909.18	208	208	208	22	22	22	2909.18	satisfied	0	5651.11
12	2879.23	211	211	211	22	22	22	2879.23	satisfied	0	5569.32
13	2825.20	207	207	207	21	21	21	2825.2	satisfied	0	5524.15
14	2934.95	207	207	207	22	22	22	2934.95	satisfied	0	5643.37
15	2987.33	212	212	212	22	22	22	2987.33	satisfied	0	5816.25
16	2819.73	207	207	207	22	21	21	2819.73	satisfied	0	5504.38
17	3000.55	209	209	209	22	22	22	3000.55	satisfied	0	5780.75
18	3069.72	220	220	220	22	22	22	3069.72	satisfied	0	5790.77
19	2819.73	207	207	207	21	21	21	2819.73	satisfied	0	5550.05
20	2819.73	207	207	207	22	21	21	2819.73	satisfied	0	5454.95

From the result, it proves introducing the delay route approach managed to consistently solve the problem and at times even reduced the need of that extra route by solving the problem of day 2 data with only 21 routes used. The 18th run shows we postponed 20 orders from the first day and even with that large amount of postponed orders, it still manages to find a feasible solution with only 22 routes in day 2.

6.4 Comparison between conventional VRPTW vs. Delay route VRPTW.

After some analysis are done, in order to compare the two approaches, we had tabulated the summarized data in Table 6.6:

Table 6.6: Comparison between the results.

	average distance	std dev	average route used	average nr of late orders
conventional VRPTW	5731.57	203.56	22	1
with delay route VRPTW	5628.51	129.77	21.6	0

In term of distance coverage, the conventional VRPTW loses out by at least 100 distance unit on average and has more inconsistent results by the looks of its standard deviation. Also, since all its solution uses 22 routes, the delay route VRPTW wins also in this number on average by 0.4 route count. The conventional VRPTW solver also loses out in term of average number of orders that still end up late after the day 2 optimization, by an average of 1 order having the risk of being late still compared to a guaranteed on time solution using the delay route VRPTW. There was about 1.8% improvement in term of the quality of the results (measured in average distance) obtained with delay route compared to the conventional VRPTW. Meanwhile, consistency was improved by 36.2% with the delay routes.

We have only looked at numbers and figures so far, how bout geographical difference between the conventional VRPTW versus the delay route VRPTW? To answer this question we captured the screenshot from our Quintiq application's model made for this purposes in Figure 5.1 to show the two best results of both days' run for comparison.

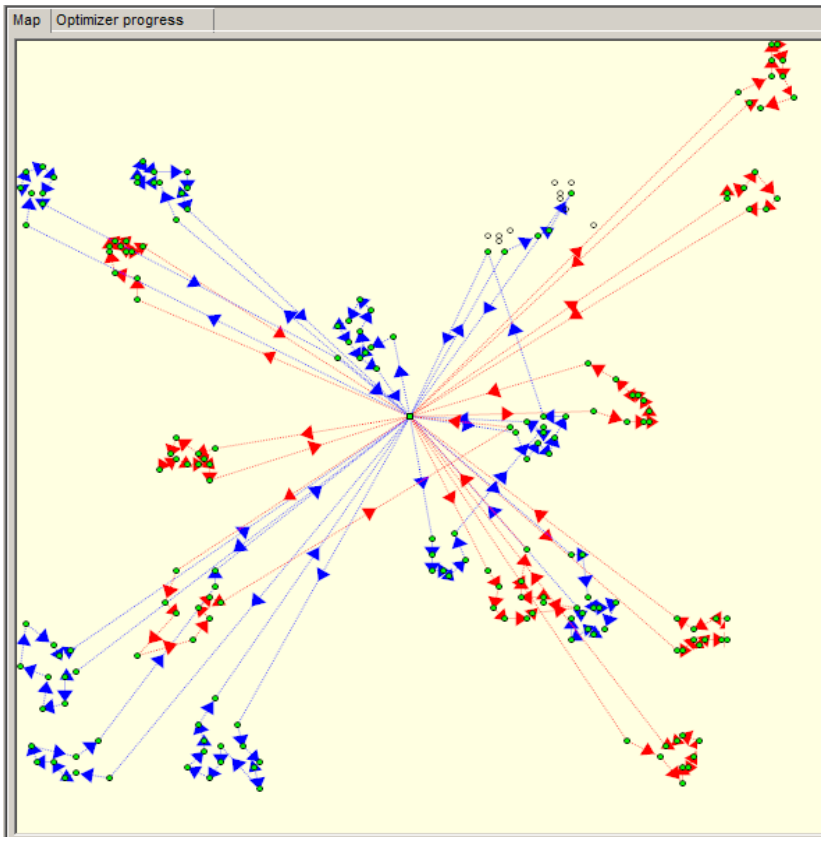

Figure 6.1: Day 1 of best solution for conventional VRPTW, run#3

From the screenshot in Figure 5.1 we can see that the chosen orders to postpone are somewhat clustered but still scattered around 2 clusters of orders and one more that is a sole order that can be easily picked up for the route that is on the way up to the top right corner clusters of orders. The dots represent the order or customer locations while the arrows and lines shows a travel for a particular route, the routes are colour-coded alternatively between red and blue to distinguish individual routes. The colour choice of red and blue is merely randomly picked for helping users distinguish routes when 2 or more are selected to be shown in the component. They carry completely no meaning for the purpose of this research and can be ignored totally.

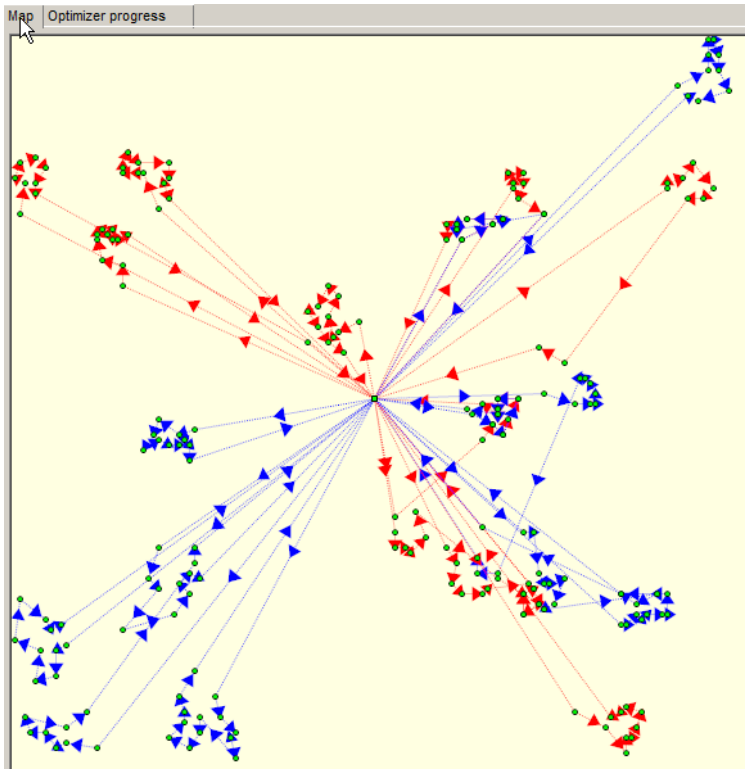

Figure 6.2: Day 2 of best solution for conventional VRPTW, run#3

From the screenshot we still see some travelling across clusters which show inefficient travelling planned on the second day. These are screenshot of the best run found out of the 10 reported for the conventional VRPTW which is the run with runNr=3 that has a total distance of 5489.73 (day 1 = 2591.71, day 2 = 2898.02) and has 10 orders that was late in the first day which is postponed to second day to plan which eventually all get planned on time in second day. Day 2 utilized all 22 routes generated at the start also in the final solution.

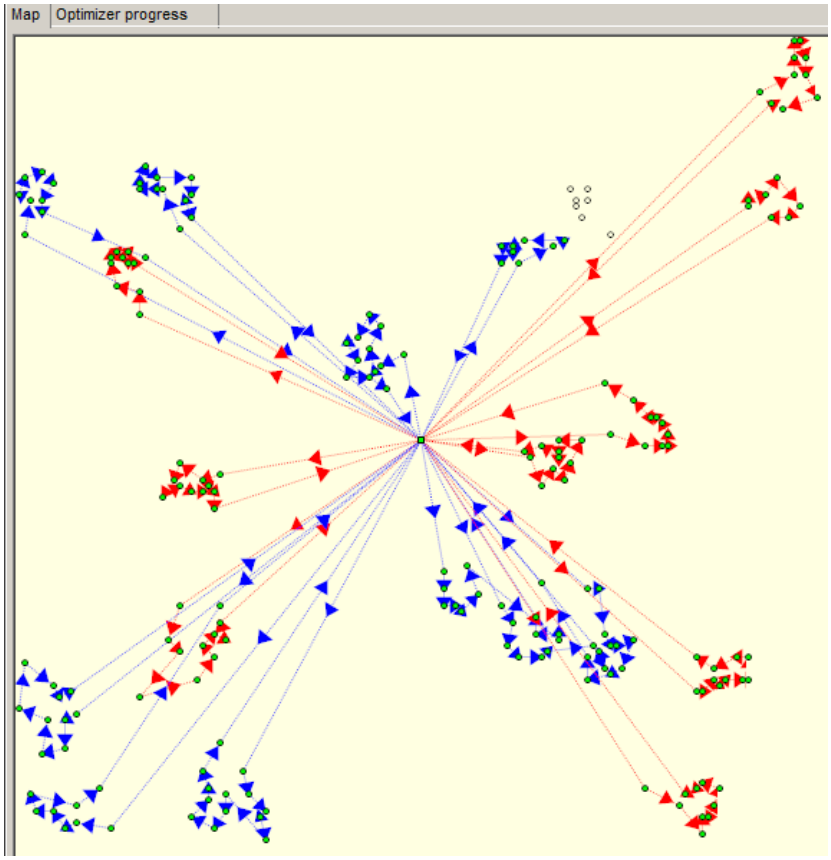

Figure 6.3: Day 1 of best solution for delay route VRPTW (penalty=1000), run#20

This time with delay route of delay penalty set to 1000, we see the best case scenario's day 1 pattern is clearly much more efficient by having a clear cluster chosen to be postponed. However, that one order is still cannot be picked up by that top right corner cluster, however it is now considerably a close by order to the cluster chosen to be postponed.

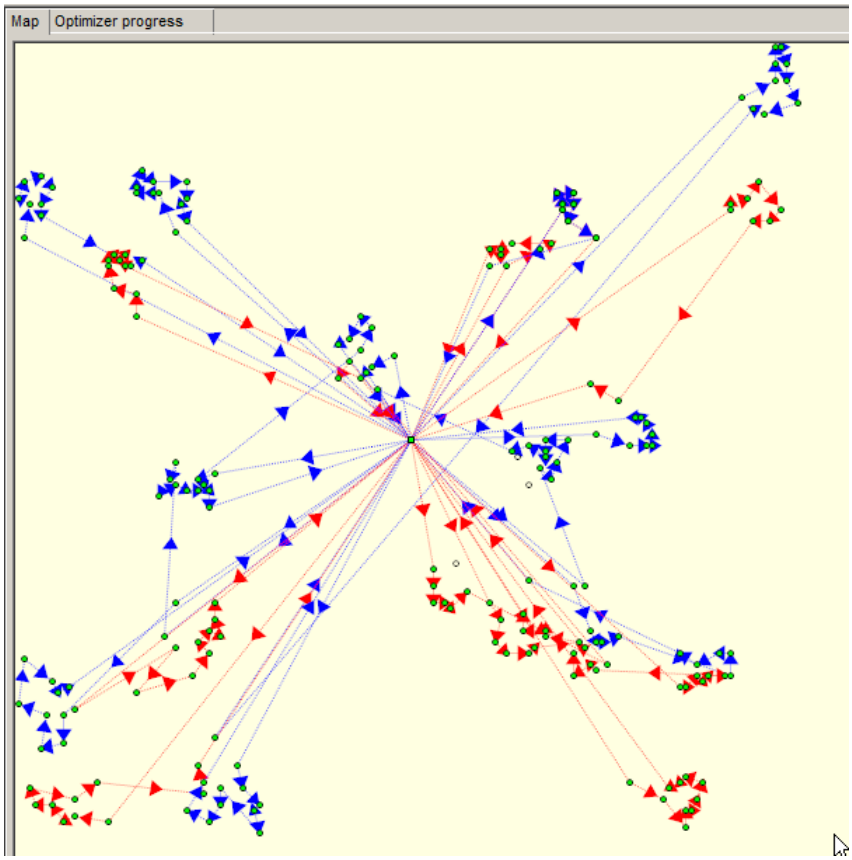

Figure 6.4: Day 2 of best solution for delay route VRPTW (penalty=1000), run#20

These are the best solution for the delay route VRPTW with penalty 1000 which is the final run with runNr=20. We see also still there are travelling cross region but however the results was much better in numbers with total distance at 5454.95 (day 1 = 2635.22, day 2 = 2819.73) and has only 7 orders that was postponed from day 1 to day 2. Day 2 only utilizes 21 routes from the 22 generated at start solution. This saves 1 route in day 2 in comparison to the conventional VRPTW.

6.5 Additional runs for delay route VRPTW with different penalties

After the experiments concluded delay route can get better results as compared to conventional VRPTW, the next question is how much should the delay penalty be? To be able to answer that, we have to continue further experiments with first a higher penalty set at 2000 and followed by again on a lower penalty set at 500 to see their difference as compared to 1000.

First we look at the results with penalty set to 2000.

The result of 10 runs of the day 1 data (C1_2_1 with limit of 19 routes only) is as follows:

Table 6.7: Results of day 1 using delay route VRPTW with 2000 penalty

RunNr	Score	NrOrders	NrOrders Planned	NrOrders OnTime	NrRoutes	NrRoutes Used	NrRoutes WithinCapacity	Distance
21	702710.3	200	193	193	19	19	19	2710.28
22	702691.2	200	193	193	19	19	19	2691.21
23	702613	200	193	193	19	19	19	2613
24	702704.5	200	193	193	19	19	19	2704.51
25	702709.5	200	193	193	19	19	19	2709.52
26	802747.8	200	192	192	19	19	19	2747.84
27	902822.9	200	191	191	19	19	19	2822.88
28	702705	200	193	193	19	19	19	2705
29	702723	200	193	193	19	19	19	2722.97
30	702667.6	200	193	193	19	19	19	2667.57

It seems in general, the number of orders that are postponed has reduced as compared to our initial gut feeling of setting delay penalty to 1000. We set it to 1000 initially because the best known score for the C1_2_1 problem was 2704.57. We do not want it to be too significant that it overtakes the whole solution's distances while at the same time we do not want it to be too low. So just taking about 1000 seems like a good start. However from the results, it seems that having 2000 as the penalty is a much better choice. All

its' runs' results showed that orders that are postponed to next day are not more than 9 and majority being at 7 which was the number we found in the best known solution so far.

The result of 10 runs of the day 2 data (C1_2_1 with limit of 22 routes only) is shown in Table 6.8:

Table 6.8: Results of day 2 using delay route VRPTW with 2000 penalty

RunNr	Score	NrOrders	NrOrders Planned	NrOrders OnTime	NrRoutes	NrRoutes Used	NrRoutes WithinCapacity	Distance	Feasible	nr of late	total distance
21	2819.73	207	207	207	22	21	21	2819.73	satisfied	0	5530.01
22	2819.73	207	207	207	22	21	21	2819.73	satisfied	0	5510.94
23	2819.73	207	207	207	22	21	21	2819.73	satisfied	0	5432.73
24	2955.09	207	207	207	22	22	22	2955.09	satisfied	0	5659.6
25	2818.55	207	207	207	22	21	21	2818.55	satisfied	0	5528.07
26	2931.98	208	208	208	22	22	22	2931.98	satisfied	0	5679.82
27	2867.84	209	209	209	22	22	22	2867.84	satisfied	0	5690.72
28	2843.96	207	207	207	22	21	21	2843.96	satisfied	0	5548.96
29	2819.73	207	207	207	22	21	21	2819.73	satisfied	0	5542.7
30	2819.73	207	207	207	21	21	21	2819.73	satisfied	0	5487.3

From these runs, we managed to find a new best solution with total distance of 5432.73 from the third run of this version, which is runNr=23. Similar to the run with 1000 penalty, most of the results in day 2 only require 21 routes to be used.

Next, we look at the screenshots of the new best found solution in Figure 5.5 for day 1 and Figure 5.6 for day 2 respectively:

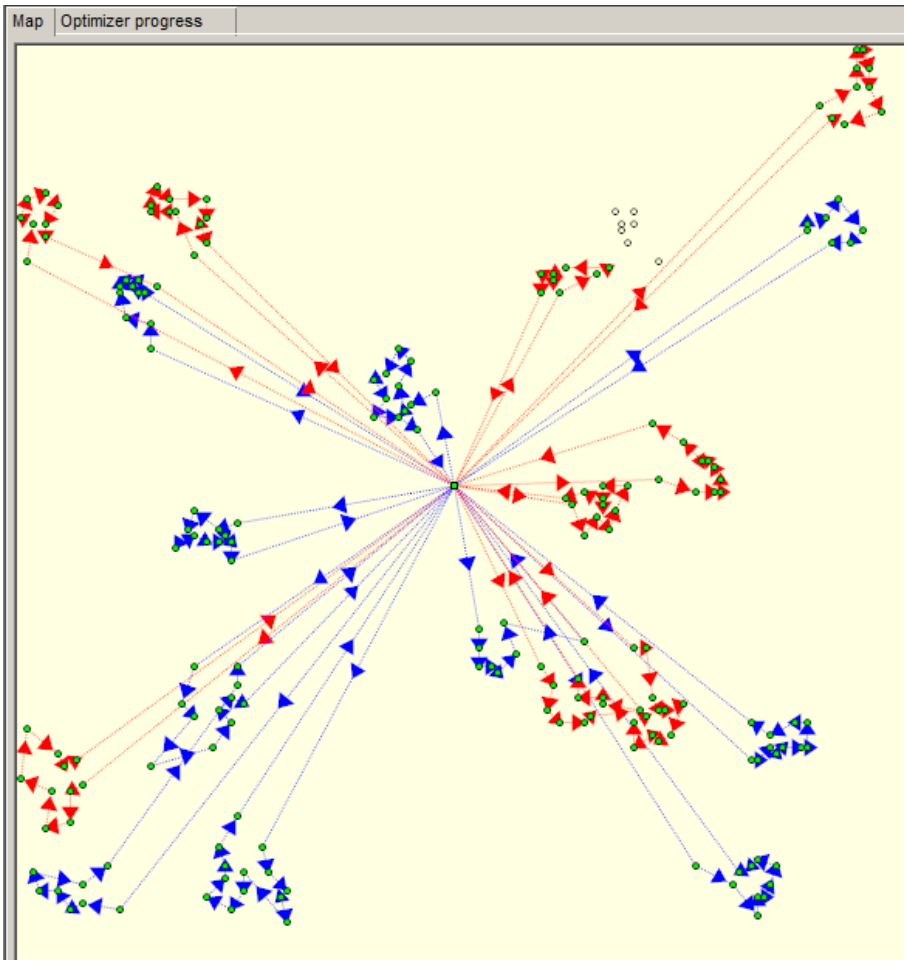

Figure 6.5: Day 1 of best solution for delay route VRPTW (penalty=2000), run#23

From the looks of it, there seem to be not much difference as compared to Figure 6.3. The same cluster of orders was left out to be postponed.

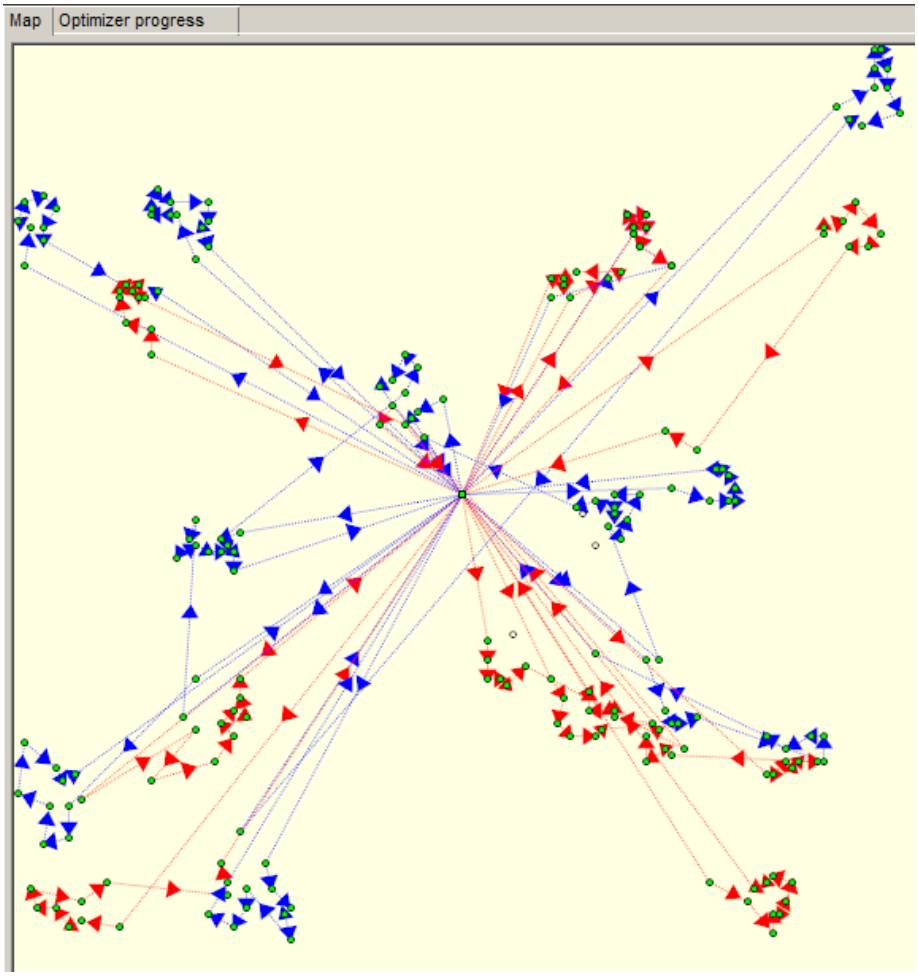

Figure 6.6: Day 2 of best solution for delay route VRPTW (penalty=2000), run#23

Again, there seems to be not much difference from Figure 6.4 which was the results obtained with penalty 1000. Next, just out of curiosity we do again 10 runs with smaller penalty at this time 500. We do not expect to see any better results here. The purpose was just to confirm the hypothesis to be 100% sure.

The result of 10 runs of the day 1 data (C1_2_1 with limit of 19 routes only) is as follows:

Table 6.9: Results of day 1 using delay route VRPTW with 500 penalty

RunNr	Score	NrOrders	NrOrders Planned	NrOrders OnTime	NrRoutes	NrRoutes Used	NrRoutes WithinCapacity	Distance
1	2102754	200	179	179	19	19	19	2754.04
2	702616	200	193	193	19	19	19	2615.99
3	1002839	200	190	190	19	19	19	2838.52
4	802759.6	200	192	192	19	19	19	2759.55
5	702676.5	200	193	193	19	19	19	2676.46
6	802607.8	200	192	192	19	19	19	2607.77
7	702753.4	200	193	193	19	19	19	2753.42
8	702675.9	200	193	193	19	19	19	2675.89
9	702592.3	200	193	193	19	19	19	2592.26
10	702738.7	200	193	193	19	19	19	2738.69

It seems to have similar results to the run with penalty set to 1000 with less consistent result in term of the number of orders that gets delayed.

The result of 10 runs of the day 2 data (C1_2_1 with limit of 22 routes only) is as follows:

Table 6.10: Results of day 2 using delay route VRPTW with 500 penalty

RunNr	Score	NrOrders	NrOrders Planned	NrOrders OnTime	NrRoutes	NrRoutes Used	NrRoutes WithinCapacity	Distance	Feasible	nr of late	total distance
1	8195.7	221	221	215	22	22	22	3071.89	violated	6	5825.93
2	2838.27	207	207	207	22	21	21	2838.27	satisfied	0	5454.26
3	4255.71	210	210	208	22	22	22	3264.29	violated	2	6102.81
4	2937.83	208	208	208	22	22	22	2937.83	satisfied	0	5697.38
5	2819.73	207	207	207	21	21	21	2819.73	satisfied	0	5496.19
6	2835.93	208	208	208	21	21	21	2835.93	satisfied	0	5443.7
7	2838.27	207	207	207	21	21	21	2838.27	satisfied	0	5591.69
8	2838.84	207	207	207	21	21	21	2838.84	satisfied	0	5514.73
9	2819.73	207	207	207	21	21	21	2819.73	satisfied	0	5411.99
10	2819.73	207	207	207	22	21	21	2819.73	satisfied	0	5558.42

The result surprises us as it seems like we have found another new best solution with the settings of delay penalty at 500. This time we found a solution with 5411.99 as total travel distance. This is runNr = 9. Note that runNr was reset in this run due to the RAM on the experiment machine ran out of memory for the application to be able to hold the results of the runs.

Next we took screenshot of the best known solution again.

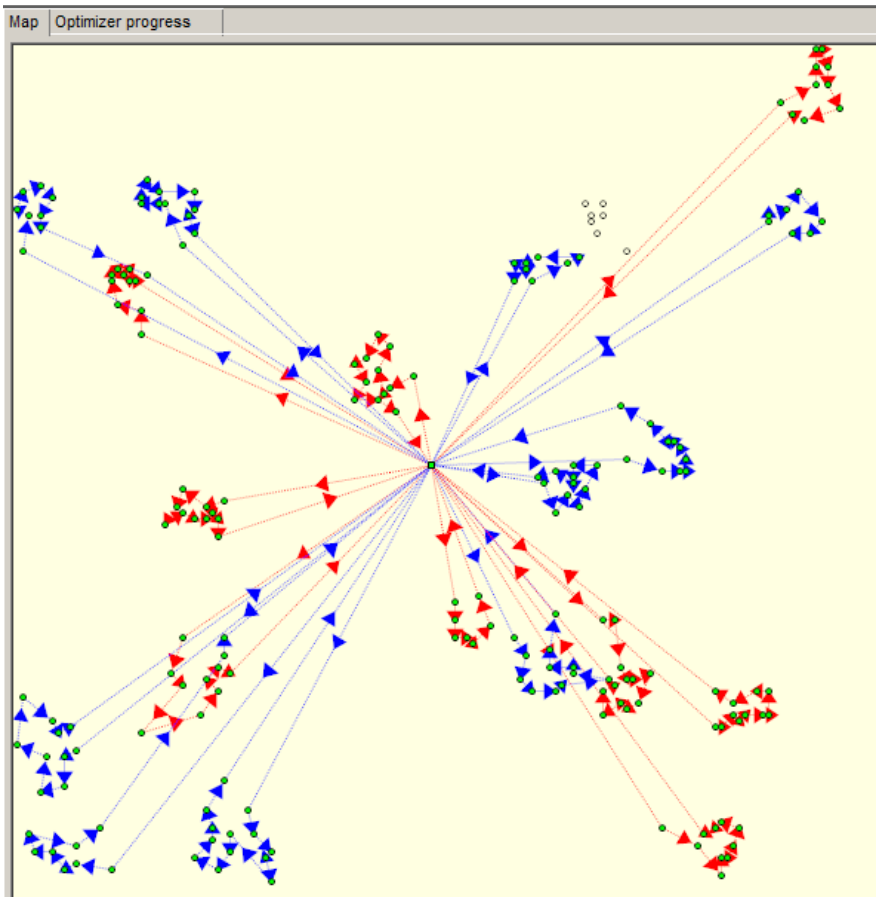

Figure 6.7: Day 1 of best solution for delay route VRPTW (penalty=500), run#9

There seems to be again not much difference with the screenshot of the coordinate representation of the solution that can be seen obvious with naked eye. The same cluster was chosen to be postponed again as well in this new best solution found.

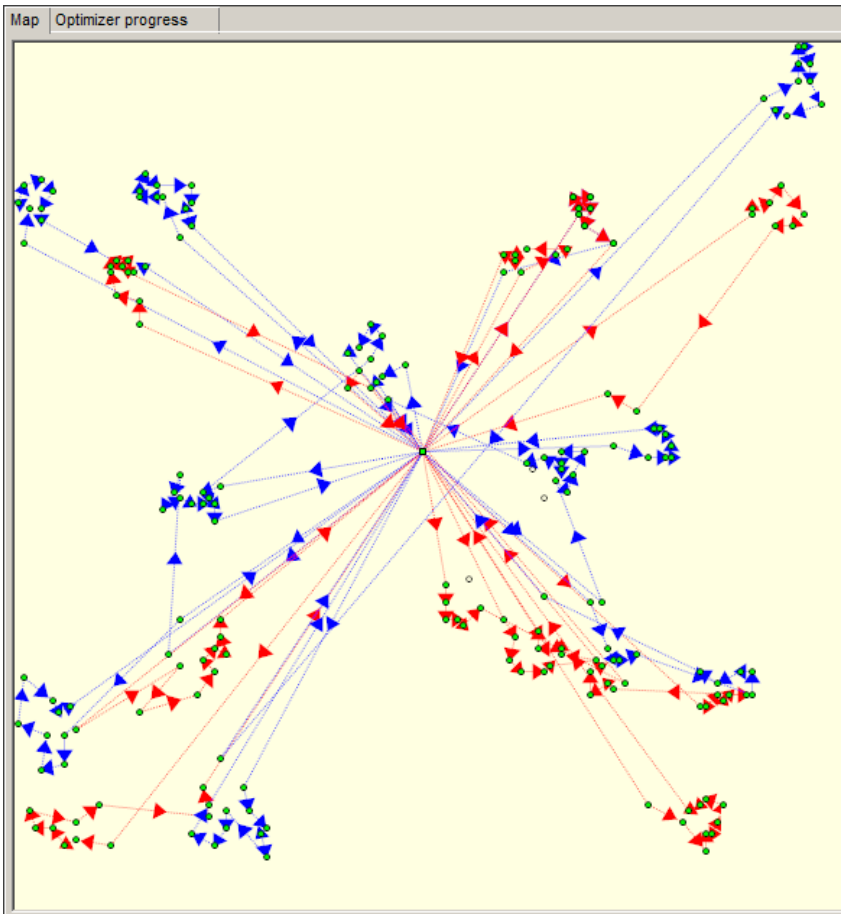

Figure 6.8: Day 2 of best solution for delay route VRPTW (penalty=500), run#9

Also, similarly in day 2 the pattern from the screenshot looks exactly the same, this is rather expected as the score for day 2 was indeed exactly the same. However, this result makes it interesting to make further analysis to compare all the results that was gathered so far.

6.6 Comparison between all results

Using similar comparison analysis done between conventional VRPTW against delay route VRPTW we now look at how each settings is doing:

Table 6.11: Comparisons between all the results.

	penalty value	std dev	average distance	average route used	average nr of late orders
conventional VRPTW	0	203.56	5731.57	22	1
with delay route VRPTW	500	214.29	5609.71	21.3	0.8
with delay route VRPTW	1000	129.77	5628.51	21.6	0
with delay route VRPTW	2000	86.66	5561.09	21.3	0

By looking at only these numbers it is fair to draw conclusion that with the penalty value set largest we get most consistency out of the solver however in term of average distance obtain has to be a give and take.

All the runs with delay route, regardless of its penalty value, have a better average distance against conventional VRPTW solver. Also all of them reports a better average number of route used. In term of the average number of orders that eventually still ends up late after day 2, it seems with delay penalty set to only 500 is not significant enough to help guarantee the consistency that all runs can solve the problem. This setting of penalty at 500 seems to give worst consistency against all others.

With further analysis done, we tabulate another summarized table of data after looking into more detailed information on the results in Table 6.12:

Table 6.12: Further analysis on the comparisons.

	penalty value	#solutions with total distance < 5.5k	range of #orders postponed from day1	average #orders postponed from day1
conventional VRPTW	0	1	8 to 15	10.4
with delay route VRPTW	500	4	7 to 21	8.9
with delay route VRPTW	1000	1	7 to 20	9.5
with delay route VRPTW	2000	2	7 to 9	7.3

In Table 6.12, we reported the number of solutions with total distance less than the threshold of 5500 (since we know current best is at 5411.99), we want to see how consistently each solver can converge as close as possible to this best result's range. Then, we also report the range and the average of the number of orders that was postponed from day 1 to day 2. What we now see is that with delay route set to lowest (in our case it was 500) the number of times it can converge close to best known solution range at below total distance of less than 5500 is much more. It was also confirmed that in these 4 solutions that met the condition, none of them was among the 2 reported from Table 6.10 that still has orders that eventually ends up late. This makes it very difficult to conclude which delay penalty is the best setting and it depends up to the domain functional decision of the business to decide if extra travelling means worse or having the risk of orders that ends up still late being worse.

CHAPTER 7.0

CONCLUSIONS AND FUTURE WORKS

7.1 Conclusions

This thesis has presented a new business problem that happens often in real world implementation of VRPTW in logistics planning problem. This new problem is known as a clustering constraint problem which happens in multi days data. Clustering constraints here states that suppose if we cannot deliver all orders on time on day 1 we want to postpone orders to day 2 in clusters. A proposed approach known as using the delay route approach has been researched and extensively experimented to prove its effectiveness.

From the results of the experiments run, it is proven that conventional VRPTW cannot solve this problem statement well enough and it is a clear conclusion that utilizing delay route approach solves the problem of having clustering constraint much better. However, in the attempt to find the best settings to use in the penalty for the approach, it was inconclusive on what was the best value to set for delay penalty to be used together in the proposed delay route approach here. Nonetheless, a few conclusions can be drawn out from the experiments' results.

We can conclude that if suppose a logistic company wants to focus in delivering their orders on time to the opening hours or time window of the order in a daily bucket, then using very large delay penalty will be a good

strategy as they tend to give consistent results that only very few orders will be postponed from day 1 to day 2. In this case, the company can rest assure that not much of orders will be left in the backlogs daily and eventually cause a huge problem when this backlogs reached a level where they may need more routes to deliver all of them.

On the other hand, suppose the logistics company has a very good IT / development team that can run the algorithms on a really hi-tech hi-end server machine, then, they can actually use small delay penalty to ensure they can obtain minimized travel distance with parallel runs of the algorithm and picking the best out of 10 will ensure they get the best known solution thus far. This will ensure the inconsistency problem with having delay penalty set too low is taken care of as they can run parallel runs and pick the best solution found out of the 10 parallel runs.

As part of the contribution, this thesis has also presented a new approach to test effectiveness of VRPTW solvers in the temporal dimension. This is achieved by using the multi day data approach for measuring the clustering constraint of VRPTW. This approach is also generic and can reuse the full sets of currently available VRPTW benchmark datasets from Solomon as well as Gehring and Homberger's extension of those.

7.2 Future works

This thesis presented a new real world problem that presents a new dimension to the classic VRPTW problem statement. The new dimension introduced here is the temporal dimension, whereby 2 days of data was taken

into account. This opens up a whole new area of research where many future research directions can be done to experiment with multi day data benchmark versions of the classic Solomon (1983) as well as Gehring and Homberger (1999) benchmarks.

It was presented here a single instance of the benchmark used in Gehring and Homberger benchmark being C1_2_1, a next step to this is to create a whole set of benchmark that duplicates the standard classic VRTPW benchmark and later possibly combining a few benchmark data to represent the dynamic nature of logistics problem whereby you do not have the same set of orders every day.

The experiment presented also used a single approach in solving the VRPTW problem instance. The approach selected was using Sequential Insertion Heuristics + Path Optimization Algorithm. The is another dimension of possible future work from here on which is to test with different sets of algorithm and conventional VRPTW solvers to easily extend by adding the delay route approach into them. The idea presented with delay route is generic and independent of algorithms used. The idea can be easily added to any conventional VRPTW solvers by adding an additional route to it and adding an equation that penalize orders being planned on this additional route. The goal should then also be modified to fit in this new term into it for minimizing.

Another future work would be to extend the problem to add in complexity of order prioritization or customer prioritization to influence the choice of orders to postpone and also to come up with a standard way of measuring this and applying it into the goal function of the solvers.

REFERENCES

Bullnheimer, R. H. B. & Strauss, C., 1997. *Applying the Any System to the Vehicle Routing Problem.* Austria, 2nd International Conference on Metaheuristics-MIC97.

Croes, G., 1958. A method for solving traveling salesman problems. *Operations Res. 6,* Volume 6, pp. 791-812.

Gehring & Homberger, 1999. *Extended Solomon's VRPTW instances.* [Online] Available at: http://www.sintef.no/Projectweb/TOP/VRPTW/Homberger-benchmark/

Kisjes, K., 2012. *A Quantitative Comparison of Generalized Fast Construction Heuristics for the Vehicle Routing Problem with Time Windows,* Rotterdam, Netherland: Erasmus University.

Li, H. & Lim, A., 2003. *Local search with annealing-like restarts to solve the VRPTW.* Singapore, European Journal of Operational Research 150, pp. 115-127.

MJC², n.d. *Why is logistics planning hard?.* [Online] Available at: http://www.mjc2.com/logistics-planning-complexity.htm [Accessed 2012].

Pisinger, D. & Ropke, S., 2007. A general heuristic for vehicle routing problems. In: *Computers & Operations Research Volume 34.* s.l.:s.n., pp. 2403-2435.

Qili, K. Z., 2000. *A New Genetic Algorithm for VRPTW.* Singapore, Proceedings of the International Conference on Artificial Intelligence.

Quintiq, 2012. *Quintiq sets new record for Vehicle Routing Problem with Time Windows.* [Online] Available at: http://www.quintiq.com/news-2012/quintiq-sets-new-record-for-vehicle-routing-problem-with-time-windows.aspx
[Accessed October 2012].

Solomon, M. M., 1983. *VRPTW Benchmark Problems.* [Online] Available at: http://www.sintef.no/Projectweb/TOP/VRPTW/Solomon-benchmark/

Soo, R. K. & Tay, Y. H., 2009. *A Survey on the Progress of Research on Vehicle Routing Problem with Time Window Constraints.* Kuala Lumpur, Malaysia, Symposium on Progress in Information and Communication Technology (SPICT), pp. 142-146.

Soo, R. K. & Tay, Y. H., 2011. *Solving VRPTW with Delay Route to Satisfy Clustering Constraint.* Kuala Lumpur, Malaysia, 2011 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology.

Vladimir, V. & Tarek, S. M., 2002. *Vehicle Routing Problem with Time WIndows.* Bridgeport, USA: Department of Computer Science and Engineering, University of Bridgeport.

Wikipedia, n.d. *Vehicle Routing Problem - Why is it hard?.* [Online] Available at: http://en.wikipedia.org/wiki/Vehicle_routing_problem [Accessed 2012].

Yiqing, Z. & Xiao, P., 2007. *A Hybrid Optimization Solution to VRPTW Based on Simulated Annealing.* Jinan, China, Proceedings of the IEEE International Conference on Automation and Logistics.

Zhu, K. Q. & Ong, K.-L., 2000. *A Reactive Method for Real Time Dynamic Vehicle Routing Problem.* Singapore, 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00).