University of Bremen
Centre for Computing and Communication Technologies TZI
Erasmus Mundus Master's Programme in Pervasive Computing & Communications
for Sustainable Development PERCCOM

Ashraf Abdo

Nested Rollout Policy Adaptation for Optimizing Vehicle Selection in Complex VRPs

2016

Supervisor(s):	Professor Dr. Ing Michael Lawo (University of Bremen) Professor Dr. Stefan Edelkamp (University of Bremen)
Examiners:	Professor Jari Porras (Lappeenranta University of Technology) Professor Eric Rondeau (University of Lorraine) Professor Karl Andersson (Luleå University of Technology)

This thesis is prepared as part of the European Erasmus Mundus programme PERCCOM - Pervasive Computing & Communications for sustainable development.

This thesis has been accepted by partner institutions of the consortium (cf. UDL-DAJ, n°1524, 2012 PERCCOM agreement).

Successful defense of this thesis is obligatory for graduation with the following national diplomas:

- Master in Master in Complex Systems Engineering (University of Lorraine)
- Master of Science in Technology (Lappeenranta University of Technology
- Master in Pervasive Computing and Computers for sustainable development (Luleå University of Technology)

ABSTRACT

Author's name: Ashraf Abdo
Title of thesis: Nested Rollout Policy Adaptation for Optimizing Vehicle Selection in Complex VRPs

Universities: Lappeenranta University of Technology – University of Bremen
Name of the school: School of Business and Management (LUT) - Centre for Computing and Communication Technologies TZI
Name of the degree programme: Computer Science (LUT)
Name of the Master's degree programme: Erasmus Mundus Master's Programme in Pervasive Computing & Communications for Sustainable Development PERCCOM
Master's Thesis, 2016
50 pages, 14 Figures, 3 tables, 3 appendixes
Examiners: *Professor Jari Porras (*LUT), *Professor Eric Rondeau* (University of Lorraine) *Professor Karl Andersson* (Luleå University of Technology)

Keywords: Monte Carlo Tree Search, Nested Rollouts Policy Adaptation, VRP, Logistics.

The goal of Vehicle Routing Problems (VRP) and their variations is to transport a set of orders with the minimum number of vehicles at least cost. Most approaches are designed to solve specific problem variations independently whereas in real world applications, different constraints are handled concurrently.

This research extends solutions obtained for the traveling salesman problem with time windows to a much wider class of route planning problems in logistics. The work describes a novel approach that:

- supports a heterogeneous fleet of vehicles
- dynamically reduces the number of vehicles
- respects individual capacity restrictions
- satisfies pickup and delivery constraints
- takes Hamiltonian paths (rather than cycles)

The proposed approach uses Monte-Carlo Tree Search and in particular Nested Rollout Policy Adaptation. For the evaluation of the work, real data from the industry was obtained and tested and the results are reported.

ACKNOWLEDGEMENTS

I would like to thank the Erasmus+ program in general and the consortium of Erasmus Mundus PERCCOM Programme in particular, for supporting and organizing such a rich master program both academically and culturally.

PERCCOM was indeed a unique journey to seek both academic and cultural attainments, and was full of life lessons and experiences.

I am very thankful to my supervisor Prof. Dr. Michael Lawo who was always giving advice and support with a very affirming and positive attitude which I highly admire.

Thanks to Dr. Max Gath for suggesting this research topic and for his invaluable input and advice throughout the research period.

This work would not have been accomplished without the guidance, the extensive knowledge and scientific feedback provided by Prof. Dr. Stefan Edelkamp.

Thanks also to the TZI team in Bremen for hosting me during this semester and providing me with all the resources and the friendly environment.

I would also like to thank Prof. Eric Rondeau, Prof. Karl Andersson and Prof. Jari Porras for their hard work to organize and run the program. They strived to deliver the best for PERCCOM all the time.

And if I was to thank only one professor from the great team of academic professionals who taught us during the study period in PERCCOM, I would have definitely chosen Professor Joseph Hallberg from LTU. He was both a brilliant professor and an older brother.

Finally, I have received a tremendous support from my family and friends, who were always there for me. I am really grateful to you and it is hard to thank you enough.

Bremen, May, 2016

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

CSV	Comma Separated Values
CVRP	Capacitated Vehicle Routing Problem
Eq.	Equation
JSON	Java Script Object Notation
KML	Keyhole Markup Language
MCS	Monte Carlo Search
MCTS	Monte Carlo Tree Search
MDVRP	Multi-Depot Vehicle Routing Problem
NMCTS	Nested Monte Carlo Tree Search
NRPA	Nested Rollouts with Policy Adaptation
OVRP	Open Vehicle Routing Problem
PDP	Pickup and Delivery
PDVRP	Pickup and Delivery Vehicle Routing Problem
TSP	Travelling Salesman Problem
TSPTW	Travelling Salesman Problem with Time Windows
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

1 INTRODUCTION

1.1 Background

As a result of globalisation and global economic growth, global goods transport is rapidly growing. Complex procedures that range from packaging and storage to final shipping and disposal are all embedded services in logistics, and they massively contribute in the overall inefficiency witnessed in this sector.

As fuel costs, taxes and environmental dangers rise, the emerging need to run logistics that are more efficient in their operations is considered to be increasingly important(The Climate Group, 2008).

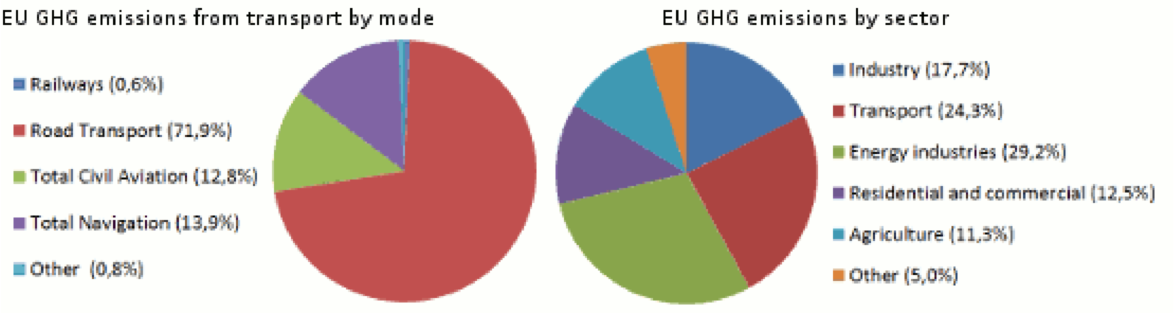

Figure 1 EU28 greenhouse gas emissions by sector and mode of transport, 2012(Reducing emissions from transport - European Commission, 2016).

Changing demographics and expansion of cities led to an overall growth of commodity flows and also of home-deliveries which are concentrated in urban districts and results in more logistics activities.

However, the underlying infrastructure is approaching limitations, especially in megacities. At the same time, urbanization and future city factories lead to more legal regulations and additional constraints on the usage of the traffic infrastructure. In order to handle the rising transport volume, logistics service providers must increase the capacity utilization significantly and try ,if possible, to avoid empty runs as suggested in the Smart 2020 report (The Climate Group, 2008).

Logistics companies should develop innovative and efficient concepts, particularly for last-mile logistics and urban deliveries. For example, transport providers must extend their fleet by more heterogeneous and (as yet) unconventional modes of transport such as e-bikes and trams which fit the specific order situation and circumstances. Beside these issues, the economical use of limited natural resources as well as sustainable transportation is essential to reducing the overall costs.

ICT sector contribution to minimizing CO2 in Transport

According to the Smart 2020 report (The Climate Group, 2008), the majority of logistics emissions come from transport and storage. Optimising logistics using ICT based solutions has the potential to achieve a 16% reduction in emissions caused by the transport sector and a 27% reduction in storage emissions globally.

Applying Smart Logistics measures have the potential of contributing solutions to the problems mentioned above, since it is by definition to "*comprise a range of software and hardware tools that monitor, optimise and manage operations, which help reduce the storage needed for inventory, fuel consumption, kilometres driven and frequency of vehicles travelling empty or partially loaded*" (The Climate Group, 2008).

1.2 Research Question

This research seeks to answer the following question:

Is the Monte Carlo Search algorithm (MCS) -and in particular the Nested Rollout Policy Adaptation variant (NRPA)- applicable in solving complex real-life Vehicle Routing Problems (VRPs) while also optimising vehicles choice?

Any suggested solver should consider the following:

- supporting a heterogeneous fleet of vehicles,
- dynamically reducing the number of vehicles,
- respecting individual volume and capacity restrictions,
- satisfying pickup and delivery constraints,
- supporting Hamiltonian paths (rather than cycles) and
- serving several individual depots.

1.3 Goals and delimitations

There is an emerging need to optimise logistics operation from both business and environmental perspectives and ICT technologies have significant potential in contributing to both parts.

This thesis studies the main dispatching problems in logistics with a particular focus on problems variants closer to common industrial applications.

The objective of the thesis is to develop a cost efficient planning algorithm used for distribution and route planning of a heterogenous fleet of vehicles, applying the constraints that occur in an everyday industrial application, to minimize the number of vehicles and duration of tours travelled.

The thesis will study famous dispatching problems in logistics, and it will investigate the feasibility of using Monte Carlo Tree Search and in particular, Nested Rollouts Policy Adaptation (NRPA) to solve industrial Vehicle Routing Problems.

The research will only look into static and deterministic dispatching problems. In such problems, all orders, fleet size, constraints and distances between customers are already known and fixed during the execution of the program. That is unlike dynamic problems where any of this information might be altered during the operation.

The study and the related tests were based on small and medium sized problems.

The heterogeneous vehicles considered in this research are vehicles that might have distinctive characteristics regarding capacity, starting location and operating times, however vehicles that differ in their average travelling speed - which entails overall different travel costs - are not considered.

1.4 Structure of the thesis

This thesis is organized as follows. Chapter 2 is dedicated to introducing general planning and scheduling problems in logistics and namely the Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP).

Chapter 3 presents the problems mentioned in the previous chapter in more detail while also introducing variations touching upon the common approaches in the literature to solve them. Chapter 3 concludes with describing the limitations of the mentioned approaches found in the literature regarding their applicability to real-world problems.

Trying to overcome the limitations above, Chapter 4 introduces the developed NRPA approach to solving the VRP problem putting into consideration real-world application requirements.

Chapter 5 then presents the results and contains a discussion of the findings in industrial use-cases, as well as a guidance to apply standardized benchmarks. it concludes suggesting research prospects for further investigation. At the end, Chapter 6 summarises the research.

2 INTRODUCING DISPATCHING PROBLEMS

2.1 Introduction

The Travelling Salesman Problem (TSP) and its extension Vehicle Routing Problem (VRP) stand amongst the most studied problems in transport logistics. Understanding these two problems in their simplest form is, therefore, necessary to familiarise the reader with the context of the problems undertaken by this research. This chapter gives a brief description of the two problems.

2.2 The Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is one of the most widely studied combinatorial optimisation problems. Its statement is deceptively simple, and yet it remains one of the most challenging problems in Operational Research (Laporte, 1992a).

TSP aims to look for the shortest path of a single vehicle to visit a set of cities while satisfying given constraints.

Gilbert Laporte (Laporte, 1992a), a prominent researcher of routing problems, defines the TSP as follows:

Let $G = (V, A)$ be a graph where V is a set of n vertices. A is a set of arcs or edges. Let $C: (C_{ij})$ be a distance (or cost) matrix associated with A.

The TSP consists of determining a minimum distance route passing through each vertex once and only once. The TSP is computationally demanding problem, and it is classified as an NP-hard problem (Papadimitriou, 1977).

In its simplest form, the TSP can be mathematically described as follows:

Let S denote a set of stops, which must be visited. Given the costs $c_{i,j}^{v}$ for travelling from $i \in S$ to $j \in S$ and choosing indicator variable as:

$$\begin{gathered} x_{i,j} = 1, \qquad if\ (i,j) are\ part\ of\ the\ tour \\ x_{i,j} = 0, \qquad otherwise \end{gathered} \qquad \text{Eq. 1}$$

the objective function of the basic TSP is:

$$\min \sum_{i \in S} \sum_{j \in S} c_{i,j} \,.\, x_{i,j} \qquad \text{Eq. 2}$$

$subject\ to$:

$$x_{i,j} = \{0,1\}\ for\ all\ i,j\ \in S \quad \text{Eq. 3}$$

$$\sum_{i \in S} x_{i,j} = 1\ for\ all\ j \in S \quad \text{Eq. 4}$$

$$\sum_{j \in S} x_{i,j} = 1\ for\ all\ i \in S \quad \text{Eq. 5}$$

$$\sum_{i \in Y} \sum_{j \in Y} x_{i,j} < |Y|\ for\ all\ Y \subseteq S\ and\ |Y| > 1 \quad \text{Eq. 6}$$

A major application of TSP is in the domain of logistics where a set of goods needs to be transported to different customers' locations while minimising the travel cost.

Figure 2 An abstract visualization of a simple static TSP illustrates a solution for a TSP in its basic form.

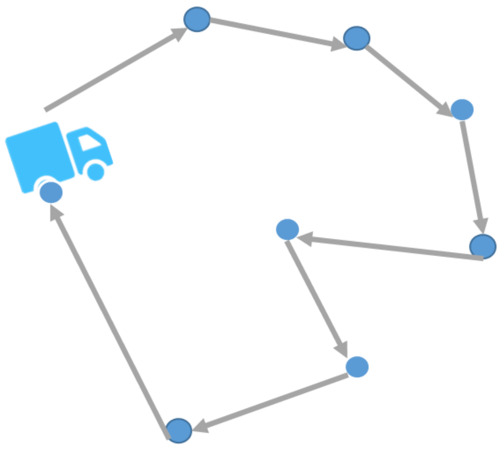

Figure 2 An abstract visualization of a simple static TSP

In the real-world application of logistics, several more constraints and factors have to be considered. To name few, for example, time windows in which the vehicle needs to arrive at the customers, the service time needed at each stop of the vehicle; orders may be picked up from several locations – not only from a central depot – and capacity constraints of the vehicles must be met.

The previous constraints -along with many other- led to the development of different problem variations of the TSP, that will be touched upon in chapter 3.1.

2.3 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is "the problem of designing optimal delivery or collection routes from one or more depots to a number of geographically scattered cities or customers, subject to side constraints" (Laporte, 1992b).

Those constraints concern several facets of the operation, such as vehicles capacities, time windows for pickup and/or deliveries, time availability of the vehicles, etc. (Psaraftis, 1995).

VRP is considered as an NP-hard problem (Lenstra and Kan, 1981).

The VRP can be seen as a generalisation of the TSP with multiple vehicles involved. The goal of TSP was to determine a minimum distance circuit passing through each node once starting at a certain depot and returning to it, whereas in VRP, the goal is to find the best allocation of orders to vehicles in addition to finding the minimum distance route for each of the vehicles, so that both allocations will lead to minimising the total cost.

VRP can be formally defined in a similar way to TSP as follows:

$$x_{i,j}^{v} = 1, \qquad if\ (i,j) are\ part\ of\ the\ tour\ of\ vehicle\ v$$
$$x_{i,j}^{v} = 0, \qquad otherwise \tag{Eq. 7}$$

The objective function of VRP is:

$$min \sum_{v \in V} \sum_{i \in S} \sum_{j \in S} c_i^v,_j \,.\, x_i^v,_j \tag{Eq. 8}$$

$subject\ to$:

$$\sum_{v \in V} \quad x_i^v,_j = \{0,1\}\ for\ all\ i,j\ \in S \tag{Eq. 9}$$

$$\sum_{v \in V} \sum_{i \in S} x_i^v,_j = 1\ for\ all\ j \in S \tag{Eq. 10}$$

$$\sum_{v \in V} \sum_{j \in S} x_i^v,_j = 1\ for\ all\ i\ \in S \tag{Eq. 11}$$

$$\sum_{v \in V} \sum_{i \in Y} \sum_{j \in Y} x_i^v,_j < \ |Y|\ \ for\ all\ Y \subseteq S\ and\ |Y| > 1 \tag{Eq. 12}$$

Figure 3 Representation of a basic VRP with two vehicles shows an example of a basic VRP problem.

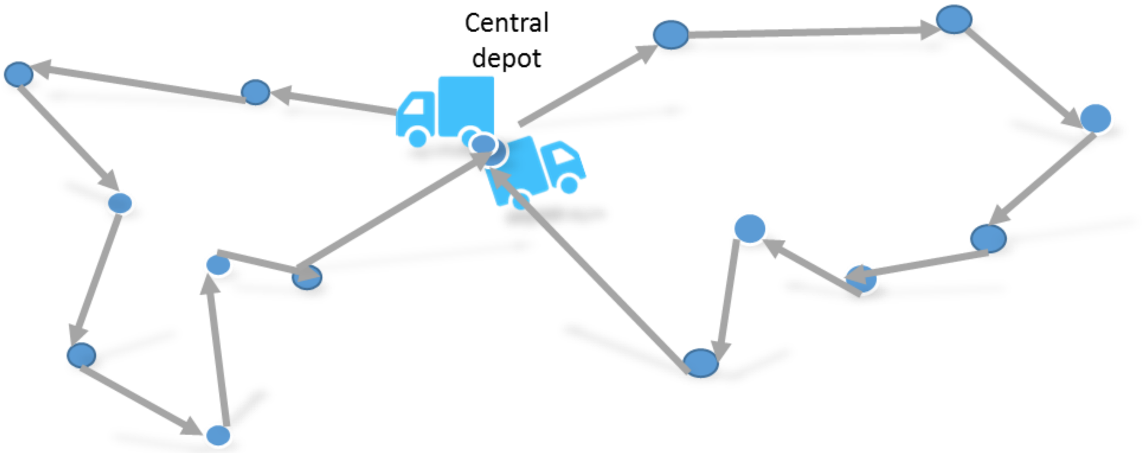

Figure 3 Representation of a basic VRP with two vehicles

3 RELATED WORK

This chapter gives an insight from relevant academic literature to the variants of the TSP and VRP problems, their solving methods and will focus in particular on the methods derived from Monte Carlo search.

3.1 TSP Variations

In the Travelling Salesman Problem with Pickup and Delivery (TSPPD) unlike the normal TSP where transport is made from a central depot, the transport includes delivering items between customers, in which case, the vehicle has to visit a pickup stop before the delivery stop as can be seen in Figure 4 A TSP with pickup and delivery.

It can be defined as follows:

The group of stops forms one set which consists of two different subsets, the one of pickup stops $P \subset S \setminus (D \cup DEP)$ and the other subset of delivery stops $D \subset S \setminus (P \cup DEP)$. Moreover, O denotes a set of orders. An order $o \in O$ contains exactly one pickup stop p_o and one delivery stop d_o.

$$(x_{i,p_o} = 1 \wedge x_{i,d_o} = 1) \Rightarrow l_{p_o} < r_{d_o} \tag{Eq. 13}$$

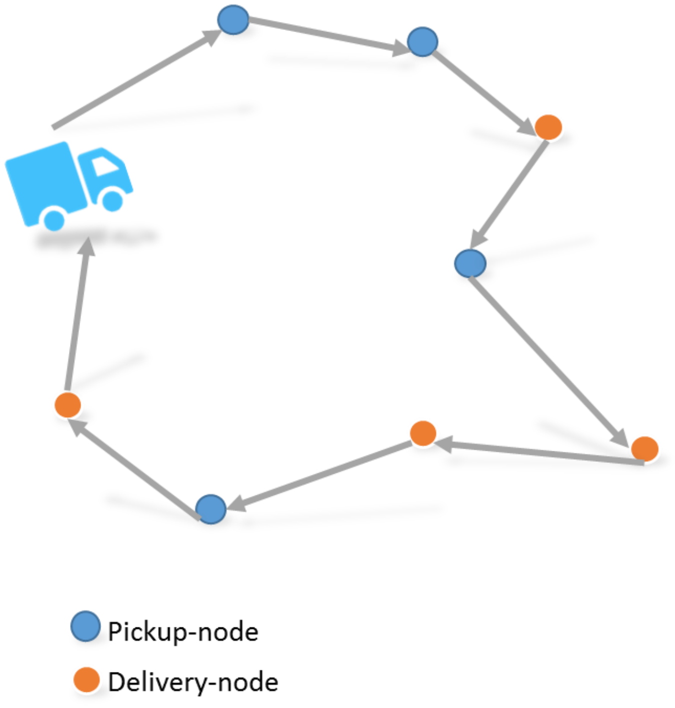

Figure 4 A TSP with pickup and delivery

The Travelling Salesman Problem with Time Windows (TSPTW) involves the design of a minimum cost path for a vehicle which visits a set of nodes. Each of these nodes will be visited once and only once, and the service at a node will be done in the time window defined by the earliest and the latest time when the start of the service is permitted at the node. If a vehicle arrives at a node too early, it has to wait. Also, due dates cannot be violated (Solomon et al., 1995).

In addition to time windows, usually a service time at the customer location has to be considered and added to the overall timespan of a vehicle.

If l_s denotes the latest pickup/delivery time at stop $s \in S,\ t_s$ is the time consumption of the loading or unloading process, r_s the release time at s and $time_{i,j}$ is the vehicle's time for driving from $i\ to\ j$. Then:

$$x_{i,j} = 1 \Rightarrow l_j \geq r_i + t_j + time_{i,j} \qquad \text{Eq. 14}$$

has to be fulfilled.

In TSPTW vehicle starts and ends its tour from a depot $d \in DEP$ where $DEP \subseteq S$ denotes a set of depots. In TSPTW, time windows at the depot d has to be met. This is represented by:

$$r_d < \min_j r_{j \in S \setminus DEP} \qquad \text{Eq. 15}$$

$$l_d > \max_j r_{j \in S \setminus DEP} \qquad \text{Eq. 16}$$

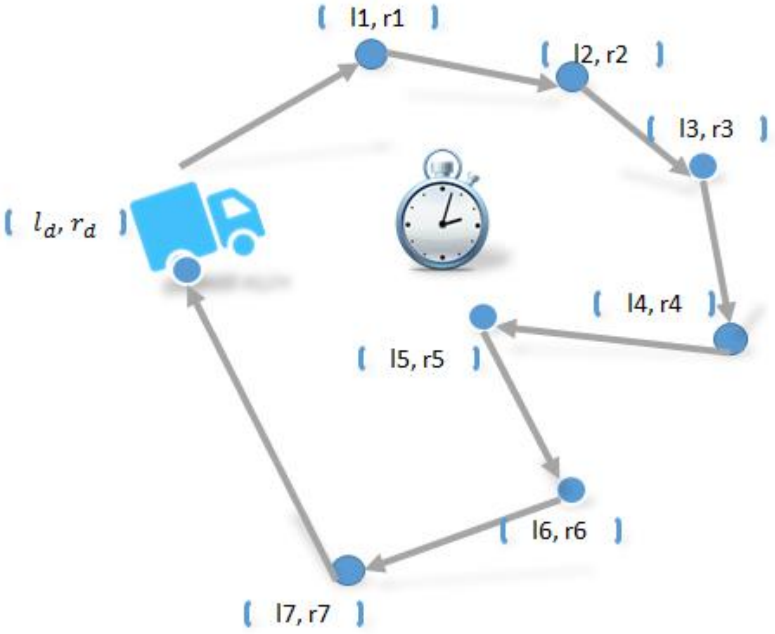

Figure 5 An example of TSP with time windows

The Travelling Salesman Problem with Capacity Constraints (TSPCC) considers the capacity constraint of a vehicle.

CC_s denotes the current capacity of the vehicle at stop $s \in S$ and M the maximum capacity of the vehicle, then the condition:

$$CC_s \leq M \, for \, all \, s \in S \qquad \text{Eq. 17}$$

has to be fulfilled.

The Generalized Travelling Salesman Problem (GTSP) (Snyder and Daskin, 2006) is a variation of the TSP in which it is not compulsory for the tour to visit all nodes. In particular, the set V of nodes is partitioned into m sets, or clusters, $V_1, \ldots, V_m$ with $V_1, \ldots, V_m = V$ and $V_i \cap V_j = \emptyset$; if i ≠ j.

The objective is to find a minimum length tour containing exactly one node from each set V_i

The pool of different TSP variations considers various other constraints, and there exist numerous approaches in the literature to solve TSP, which will be discussed in the succeeding section.

3.1.1 Common Approaches to solving TSP

Although TSP is easy to understand as a concept, it is not an easy task to find an optimal solution. The main difficulty of this problem is its large branching factor considering a vast number of possible tours $(n-1)!/2$ for a symmetric n stops tour. As the number of stops in the problem increases, the numbers of permutations of valid tours also increase. It is this factorial growth that makes the task of solving the TSP very difficult even for modest n sized problems (Razali and Geraghty, 2011).

In the literature, there are numerous approaches to solve the TSP and TSP variants. For example, genetic algorithms (Snyder and Daskin, 2006) to solve the General Travelling Salesman Problem, which was competitive with other heuristics methods in terms of quality and processing time; it has a simple implementation, which can be further extended. Other methods for solving TSP include simulated annealing, tabu-search, ant colony systems, particle swarm algorithms, neural networks approaches, k-opt improvement heuristics, branch-and-bound algorithms. Extensive surveys of the TSP solving methods can be found in (Cook, 2012), (Gutin and Punnen, 2007) and (Laporte, 1992a).

Most of the solver types mentioned above were designed solely to solve one type of TSP. However, and as referred to in chapter 2, the applicability of any TSP-related problem requires solvers that combine multiple constraints.

One recently introduced solver (Edelkamp and Gath, 2013) uses the Monte Carlo Tree search algorithm for the TSPPD and Time window while also considering capacity constraints. Since the resulting implementation of this study was successfully employed in an actual industrial application (Gath, 2015) of the TSP with multiple constraints, it raised interest to discover the method in more detail to better understand its adoption in the logistics sector.

The sections ahead will be dedicated to investigating the Monte Carlo method briefly, in addition to its use.

3.1.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS), at its simplest, is *“a method for finding optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results”*. Monte Carlo method has proved capable in the applications of Artificial Intelligence (AI) domain where problems can be formulated as decision trees. Famous examples of this kind of problems are games and planning problems (Browne et al., 2012).

What makes MCTS stand among other algorithms in AI is its ability to be applied to different kinds of problems without extensive previous knowledge (heuristics) of the application domain. Another factor that plays in favour of MCTS is being a standalone algorithm that exploits processing capacity to perform better. MCTS has proved capable of solving many problems, but probably the most prominent achievement of Monte Carlo methods in the recent time has been achieved in the two players gaming domain where the program called “AlphaGo” written by researchers in Google® using Monte Carlo tree search has been able to beat for the first time a human player in the famous game Go breaking a long-standing world record (Silver et al., 2016).

The MCTS process is simple in concept in its basic format, and is based on iterative sampling; it consists of playing random games until a specified run time is over or until a solution is found (Cazenave, 2009). A tree policy is used to find the most urgent node to go to, starting from the current node in the tree. Representing the search space as a tree makes it easier to consider balancing the search between promising known neighbourhoods and looking into new undiscovered neighbourhoods. The two processes are usually referred to as exploitation and exploration respectively. (Browne et al., 2012).

```
int sample (position)
while not end of game
  position = play (position,
          random move)
return score
```

Algorithm 1 MCTS in its simplest format as seen in (Cazenave, 2009)

Unlike in minimax search, the values of nodes representing transitional states of the search are not assessed, which explains why MCTS does not rely much on domain knowledge. The evaluation (simulation) is performed only at leaf nodes of the tree. Although the basic version of MCTS is considered as a highly competitive algorithm in many application domains, the real potential of MCTS method is evident the most when the algorithm is customized to adapt better to the intended application domain (Browne et al., 2012).

Academic work in the MCTS domain strives to find different algorithm variants and improvements so that each of these improvements matches its application domain the best and can be further extended to be applied on broader perspectives. In the following section, the focus will be only on MCTS variations that have been used to solve transportation logistics problems like TSP and VRP.

3.1.3 Nested Monte Carlo Tree Search

Tristan Cazenave introduced one of the adaptations of the MCTS in 2009; it is called the Nested Monte Carlo Search (NMCS) (Cazenave, 2009). This variation of Monte Carlo algorithm guides the search at a given level of the search tree by the result of the searches at the lower levels. On the base level of the search tree, random moves are used, and then the best sequence of moves is memorised to improve the search of the higher levels.

NMCTS combines nested calls with randomness in the playouts and memorization of the best sequence of moves. The NMCS function performs one gameplay (rollout), choosing at each step in the game, a move that has the highest score of the lower level NMCS. At each step, the algorithm tries all possible moves, plays a nested search at the bottom level after each move, and memorises the move associated with the best score of the lower level searches.

In nested rollouts (playouts), the rollouts are based on a heuristic. It implies that nested rollouts always improve the previously performed rollouts, by simply following the heuristic. In the base level, it is possible that the search does not use heuristics, but rather use random moves, which in turn arises the possibility of a nested search giving worse results than a lower level search. Therefore, the nested search of a high level will lead without certainity to a better result than the previous ones or even lower level searches. To keep track of the best moves and of the corresponding sequences already found by a previous search, the algorithm has to keep the best sequence found so far in order to follow it when the randomised searches give worse results than the best-known sequence so far. Otherwise, and when the search succeeds in finding a better solution, the best sequence will be updated with the newly discovered sequence and the best move is played (Cazenave, 2009).

```
nested (position, level)
best score = -1
while not end of game
 if level is 1
     move = sample (play (position, m))
 else
     move = nested (play (position, m), level - 1)
 if score of move > best score
   best score = score of move
   best sequence = seq. after move
   bestMove = move of best sequence
 position = play (position,bestMove)
return score
```

Algorithm 2 Nested Monte Carlo Search Algorithm as presented by (Cazenave, 2009).

While introducing the NMCTS, Tristan Cazenave demonstrated how memorising the best sequence of moves improves the mean results of the search by running the algorithm on three different problems from games domain. The games were Morpion Solitaire, SameGame, and 16x16 Sudoku. Later, the Nested Monte Carlo search was used to solve the Travelling Salesman Problem with Time Windows (TSPTW). The algorithm was able to compute the state of the art results for problems of a size less than 30 nodes in a small computation time. For a full overview of this implementation, the reader is referred to (Rimmel et al., 2011).

3.1.4 Nested Rollouts Policy Adaptation (NRPA)

The Monte Carlo search algorithm and its enhancement Nested Monte Carlo search uses static uniform random or domain-specific policies to choose the successor moves.

(Rosin, 2011) proposed an enhancement to the Nested Monte Carlo Search Algorithm of (Cazenave, 2009). This new algorithm dynamically adapts the successors move choice policy during the search, called the Nested Rollouts with Policy Adaptation (NRPA).

The playout (rollout) policy in the implementation is a matrix or a vector of decimal values representing weights used to calculate the likelihood of choosing a certain move during the game (Cazenave and Teytaud, 2012). Starting from a random node in the search tree, and trying from there to get an effective policy to guide the search towards good results will not work (Rosin, 2011). However, the policy in NRPA is most effective when it builds on the best found solutions in the search space.The policy would direct the search towards the neighbourhoods of such solutions and help drive future rollouts towards them.

The following resembles a pseudocode of the NRPA with algorithm with parts in bold highlighting the main difference from the latter NMCTS:

```
1  NRPA(level,pol):
2    if level == 0: // base rollout policy
3        node = root(), ply = 0, seq = {}
4        while num_children(node) > 0:
5            CHOOSE seq[ply] = child i with probability
6                proportional to exp(pol[code(node,i)])
7            node = child(node,seq[ply])
8            ply += 1
9    return (score(node),seq)
10
11   else: // for nesting levels>=1
12       best_score = -infinity
13       for N iterations:
14           (result,new) = NRPA(level-1,pol)
15           if result >= best_score THEN:
16               best_score = result
17               seq = new
18               pol = Adapt(pol,seq)
19   return (best_score,seq)
```

Algorithm 3 NRPA algorithm as presented by (Rosin, 2011)

One can observe that the choice between one of the successors of a node has been replaced from a uniform choice, to a choice based on the policy value of the node (lines 5 and 6), which represents the probability to choose a certain node as a next move starting from the current position.

When NRPA succeeds in finding a better solution, the adapt function is called (line 18) to update the policy; so the search can be directed more toward the neighbourhood of this good result.

3.1.5 NRPA for TSP

The work of (Cazenave and Teytaud, 2012) used NRPA to improve the solution of the Travelling Salesman Problem obtained by (Rimmel et al., 2011) using Nested Monte Carlo Search (NMCS).

The algorithm succeeded in giving quality results, which in many cases matched the state of the art solutions without utilizing any expert knowledge or heuristics while solving the problem. When applying NRPA with expert knowledge of the TSP, the solution provided by the algorithm matched the state of the art solutions of 23 out of 30 problems from the standard benchmark set used. Two important works followed (Cazenave and Teytaud, 2012) in using NRPA to solve TSP were the research by (Edelkamp et al., 2013) and (Edelkamp and Gath, 2013).

In the first research, several enhancements on the algorithm were discussed, ranging from simple steps like avoiding the usage of copy constructors, to reducing the memory overhead, in addition to applying other algorithm engineering measures. The solver method presented in the paper was later adopted in an actual implementation of a multi-agent based system dealing with dynamic scheduling problems. In that system, each agent encompasses a version of the solver so that the agent can compute its tour independently and later negotiate with other agents based on the result obtained.

The second research has proven to succeed in using NRPA to solve “hybrid” TSPPD problems that also involved other constraints like capacity and time window. It also implemented all the enhancements suggested by the first research into the solver.

Other examples of the usage of Monte Carlo methods in logistics domain were presented by (Edelkamp et al., 2016).

3.2 Vehicle Routing Problem (VRP)

3.2.1 VRP variations

Just like with TSP, adding additional constraints to the classical definition of VRP would lead to different variations of the problem like VRPTW stands for VRP with time windows and handling times.

CVRP means capacitated VRP. It is a VRP variation with a maximum capacity constraint of each vehicle.

In VRPTW and CVRP, Eq. 13 and Eq. 17 must be fulfilled for each vehicle $v \in V$.

PDVRP is a VRP with Pickup and Delivery. In unpaired PDPs, transported goods are homogeneous and exchangeable. Thus, any item can be delivered to any customer. In paired PDPs every item has a specific sender and recipient. Consequently, the pickup and delivery requests of an order o have to be served by the same vehicle v. This is guaranteed by

$$\sum_{i \in S} x_{i,p_o}^{v} - \sum_{i \in S} x_{i,d_o}^{v} = 0 \; for \; all \; i \in S \; and \; v \in V \qquad \text{Eq. 18}$$

All previous variations of VRP assume that vehicles start from a single depot and at the end of their tour come back to the starting point. However, there exist VRP problems where vehicles might start at different locations, and they may or may not return to the depot.

The Open Vehicle Routing Problem (OVRP) is a VRP problem in which vehicles –after finishing their tours- do not have to come back to the starting point (Pisinger and Ropke, 2007).

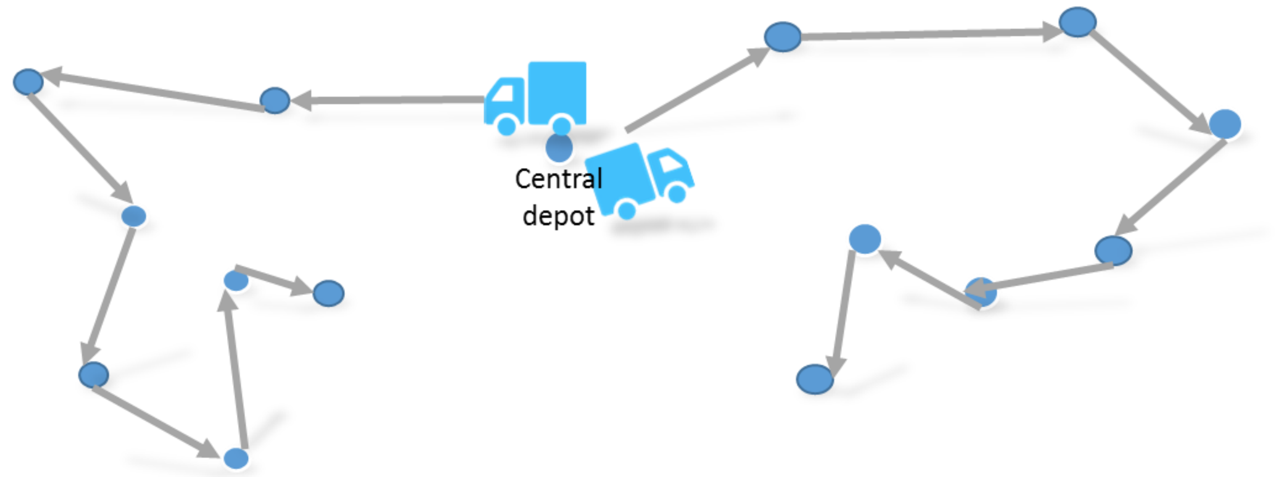

Figure 6 An example of open VRP

Multi-Depot Vehicle Routing Problems (MDVRP) are VRP problems in which vehicles may start from different depots rather than a central depot but vehicles must return to the same depot where they started (Renaud et al., 1996).

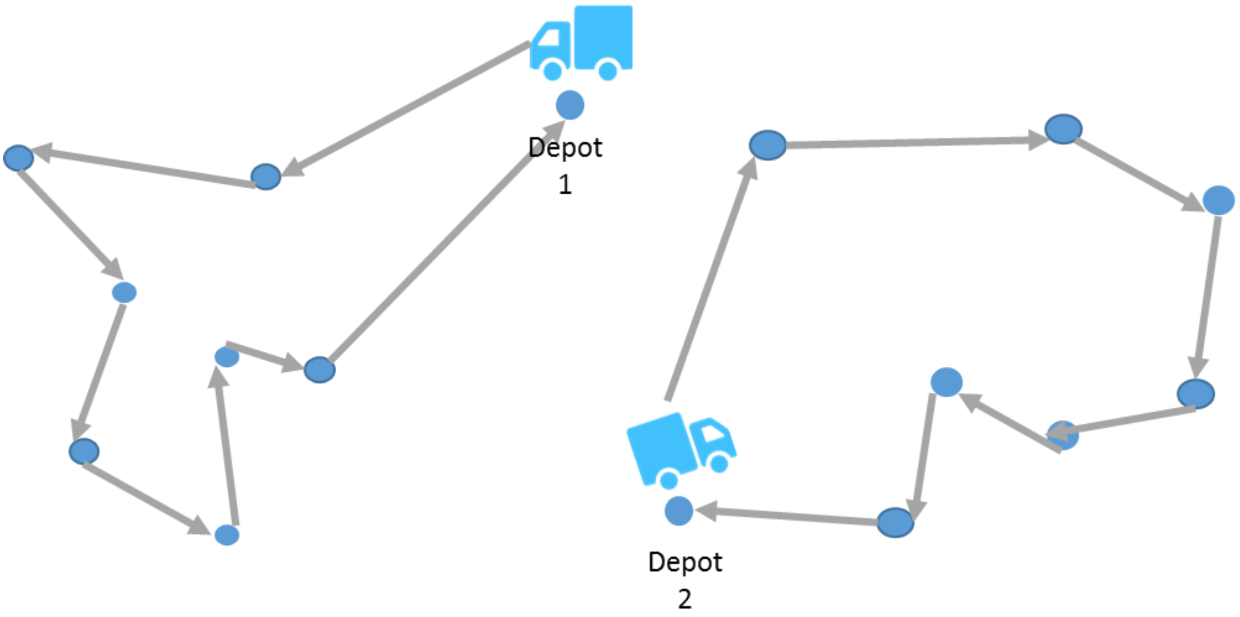

Figure 7 An illustration of multi depot VRP

The VRP variant with heterogeneous fleet does not assume that the vehicles of the fleet are identical and therefore it considers this differences while solving the given problem. The vehicles in this type of VRP can vary in capacity, starting location, ending locations, operation time and average travel speed to name but a few. Figure 8 shows an example of this variant of VRP.

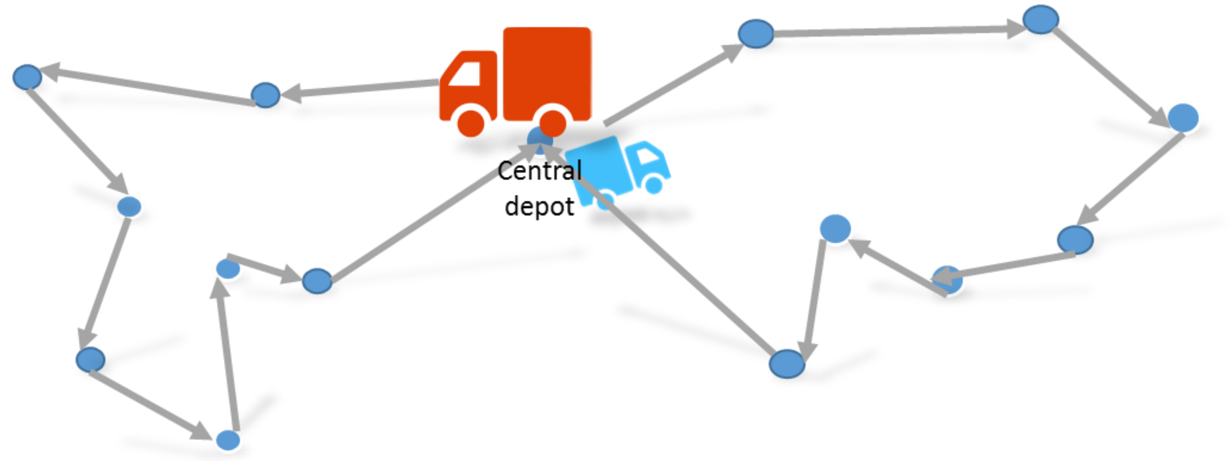

Figure 8 An example of VRP with heterogeneous fleet

Other VRP variations with different objectives are for example (Soonpracha et al., 2014) (Gaur et al., 2013). The reader is referred to the thorough study of VRP by (Toth and Vigo, 2014) for a comprehensive study on vehicle routing problems.

3.2.2 VRP complexity

It has been proven by (Gath, 2015) that the number of possible tours for the plain VRP is given by this formula: $\frac{(n+k)!}{k!}$, where n here denotes the number of stops to be visited, and k stands for the number of available vehicles. This means that even for a simple problem of two vehicles and ten stops, for example, there are as many as 39,916,800 different combinations of vehicles to stops allocations.

That explains the heavy computational capacity that is needed to solve even simple instances of the problem, not to mention that in case of PDVRP the number of stops equals twice the number of orders!

3.2.3 Common methods to solving VRP

As with TSP, its bigger sister the VRP also has numerous methods to solve using a vast variety of algorithms. It is hard to survey all the different methods to solve VRP and VRP variants in work within the scope of this research, since there are books which are in whole fully dedicated to survey these methods and be as inclusive and as comprehensive as possible. However, it is worth to mention some of the prominent methods.

So far there were three main options for solving VRPs:

1. Using a heuristic solver (tabu-search, genetic algorithms, large neighbourhood searches, simulated annealing, and ant colony based systems in combination with a local

optimizer) to first produce a valid and, then, an optimized tour. While these algorithms are working very well in less constraint vehicle routing problems, in highly constraint problems, they can get stuck.

2. Using a centralized solver for the vehicle routing problem by inserting a mathematical programming model to a competent solver. There are state-of-the-art VRP solvers based on integer programming libraries (for example SCILP and CPLEX) but substantial engineering in form of branch-and-cut support is needed to make the black-box solving mechanism terminate for larger problems.

3. Using agents that solve the problems individually and that trade the objects to be moved based on frequent solver calls. Here depth-first branch-and-bound for smaller sized sub tours and NRPA for larger-sized sub tours work well. The advantage of the agent system to the centralized solver is mainly in the robustness of the solution, as dynamic changes can be worked out well. On the other hand, the solution quality is not always sufficient.

Also, branch-and-bound based solvers require high computational capacity and are usually used to obtain optimal VRP solutions.

A comprehensive and recommendable overview of solution methods for solving multiple variations of VRPs are provided by (Baldacci et al., 2010), and the extensive work of (Toth and Vigo, 2014) among many others.

3.3 Conclusion

After concisely studying the state of the art, one can draw the following conclusions:

- The TSP and VRP as examples of well-known dispatching problems in transport logistics have been thoroughly studied in literature for a long time.
- VRP is a natural extension to TSP and even harder to solve.
- There exist many variants of the two problems, and each TSP variant does exist in the VRP while the opposite is not true.
- The majority of the solvers that exist in literature concentrate on finding a solution for one of the problems.
- VRP solvers are more concerned with performance, and the technical implementation and testing are performed on toy-problems.

Using NRPA in solving TSP and other logistics-related problems like the ones presented by (Edelkamp et al., 2016) does not directly entail its success in its extended sister problem (VRP) and its different variations. Therefore, motivated by the work of (Edelkamp and Gath, 2013), and considering the identified gaps in the literature, this research is required to test applicability of NRPA to solve complex highly constrained VRPs.

4 THE NRPA APPROACH TO COMPLEX VRP

The NRPA approach described in this section aims to fill the identified gap in terms of:

1- Being unique in using NRPA to solve the vehicle routing problem building on the previous research using NRPA to solve TSP.
2- Creating a VRP solver that handles multiple constraints usually handled separately.
3- Considering vehicle choice.

In the following, a formal mathematical representation of the problem with all the constraints is given. Details of the implantation are then described in terms of choosing the successors, selecting the vehicles, describing the policy matrix and the adapt function and the measures taken when the search for a tour reaches its end.

4.1 Formal Problem Definition

The mathematical formulation of the proposed algorithm and how problem constraints are handled is based on (Pisinger and Ropke, 2007) work, which is in turn based on the book of (Toth and Vigo, 2014).

For a VRP problem with pickup and delivery, multiple depots, capacity constraints, time windows, free ends, and heterogeneous vehicles (in terms of starting locations, capacity, and operating times).

There are n number of requests which will be served by a maximum of m available vehicles.

The problem is defined on a graph where:

$K = \{0,\dots,m-1\}$ is the set of vehicles.

$P = \{2(m+i),\ \dots, 2(m+n)-2\}\ for\ i\ \in \{0,\dots,n-1\}$ is the set of pickups.

$D = \{2(m+i)+1,\ \dots, 2(m+n)-1\}\ for\ i\ \in \{0,\dots,n-1\}$ is the set of deliveries.

Each vehicle k belongs to the set V and has a start and an end terminal represented on the graph by nodes: $\mathrm{T_k}$ and T_k` respectively.

$$t_k = 2\,k\ \ for\ \ k\ \in\ \{0,\dots,m-1\}$$

$$t`_k = 2k + 1\,for\,k\ \in\ \{0,\dots,m-1\}$$

Each request i is represented by two nodes on the graph, the
pickup location $P_i = 2(m + i)\,for\,i\ \in\ \{0,\dots,n-1\}\ and\ P_i \in \mathrm{P}$
and
delivery location $D_i = 2(m + i) + 1\ \mathrm{for}\ i\ \in\ \{0,\dots,n-1\}\ and\ D_i \in \mathrm{D}$.

As one vehicle might not be able to serve all requests, one has to maintain that the vehicle which served the pickup would be the one to serve the delivery. These kinds of limitations are modelled by letting $P_k \subseteq P$ and $D_k \subseteq D$ be subsets of pickups and deliveries served by vehicle k. Since every request is serviced by the same vehicle, it can be assumed that $i \in P_k \Leftrightarrow i + 1 \in D_k$, i.e. that both the pickup and delivery can be serviced by vehicle k. Define $N = P \cup D$ and $Nk = Pk \cup Dk$ the directed graph $G = (V, A)$ consists of the nodes $V = \{\tau_0, \dots, \tau_{m-1}\} \cup \{\tau`_0, \dots, \tau`_{m-1}\} \cup N$ and the arcs $A = V \times V$.

For each vehicle, there exists a subgraph $G_k = (V_k, A_k)$, with $V_k = \{t_k\} \cup \{t`_k\} \cup N_k$ and $A_k = V_k \times V_k$. For each edge $(i, j) \in A$ a travel time is assigned so that $t_{ij} \geq 0$.

It is assumed that all travel times satisfy the triangle inequality i.e. $t_{ij} \leq t_{il} + t_{lj}\ for\ all\ i, j, l \in V$.

A service time s_i and a time window $[a_i, b_i]$ is assigned to each node $i \in V \setminus (T_k \cup T_k`)$. This means that service time is considered at all locations except at the start and end location of each vehicle. The service time represents the time needed for loading and unloading and the time window indicates when the visit at the particular site should start.

A visit to node i can only take place between time a_i and b_i. A vehicle is allowed to arrive at a site before the start of the time window, but it has to wait until the start of the time window before the visit can be performed.

For each node $i \in N$, a weight w_i of goods that should be loaded onto the vehicle at the particular node.

Providing that $w_i \geq 0\ for\ i \in P\ and\ w_i = -w_{i-1}\ for\ i \in D$ each vehicle $k \in K$ has room for a certain amount of goods; this capacity is given by C_k. Each vehicle k should follow a legal route from its start terminal t_k to its destination terminal $t`_k$. A legal route r is a simple (loop-free) path:

$$\bar{r} = (t_k \longrightarrow v_1, v_2, \dots, v_h \longrightarrow t`_k) \tag{Eq. 19}$$

That loop-free path satisfies the time window constraints set by the customers, the capacity of the vehicle, the precedence of a pickup over delivery, and only requests that can be serviced with vehicle k.

$$i \leq j, v_i \in P_k, \qquad v_j \in D_k, \qquad vj = vi + 1 \tag{Eq. 20}$$

The previous condition represents the pickup and delivery precedence condition and that a pickup-delivery pair should be served by the same vehicle.

To ensure that time windows are satisfied, $S_i \in Z_0^+$ is defined to denote when the vehicle starts the service at site v_i. The above will be translated into the following constraints

$$\bar{r} = (t_k \longrightarrow v_1, v_2, \ldots, v_h \longrightarrow t`_k) \quad \text{Eq. 21}$$

With

$$a_{vi} \leq S_i \leq b_{vi} \; i = 1, \ldots, h \quad \text{Eq. 22}$$

$$S_{i+1} \geq S_i + s_i + t_{vi,vi+1} \quad \text{Eq. 23}$$

$$a_{t_k} \leq S_1 \leq b_{t_k} \quad \text{Eq. 24}$$

$$a_{t`_k} \leq S_h \leq b_{t`_k} \quad \text{Eq. 25}$$

Where $[a_{t_k}, b_{t_k}]$ is the time window of terminal t_k (the depot) and $[a_{t`_k}, b_{t`_k}]$ is the time window of terminal $t`_k$. Finally, the capacity of the vehicle should be respected throughout the path. For this purpose, $L_i \in Z_0^+$ is introduced to denote the load of the vehicle at node i after serving this node i. This leads to:

$$L_i \leq C_k \; i = 1, \ldots, h \quad \text{Eq. 26}$$

$$L_{i+1} = L_i + l_{i+1} \quad \text{Eq. 27}$$

$$L_1 = 0 \quad \text{Eq. 28}$$

$$L_h = 0 \quad \text{Eq. 29}$$

The travel cost of a given route $\bar{r}$ is

$$c_{\bar{r}} = \sum_{i=0}^{h} t_{v_i, v_{i+1}} \quad \text{Eq. 30}$$

If there were still any unmet requests after the search ends, they would be added to the end of the tour without having their cost calculated.

The whole problem one can formulate as follows: let R be the set of all feasible routes. The Boolean matrix $(\alpha_{j\bar{r}})$ for $\bar{r} \in R$ and $j = 1, \ldots, n$ is used to indicate whether request j is serviced using route r. The Boolean matrix $(\beta_{k\bar{r}})$ for $\bar{r} \in R \; and \; k = 1, \ldots, m$ is used to indicate whether the route $\bar{r}$ is carried out by vehicle k. Using binary variables $x_{\bar{r}}$ to indicate whether route $\bar{r}$ is used in the solution, the following model is introduced:

$$\min f(x) = \sum_{\bar{r} \in R} c_{\bar{r}} x_{\bar{r}} \tag{Eq. 31}$$

$$Subject\ to \sum_{\bar{r} \in R} \alpha_{j\bar{r}} x_{\bar{r}} = 1 \qquad j = 0, \dots, n-1 \tag{Eq. 32}$$

$$\sum_{\bar{r} \in R} \beta_{k\bar{r}} x_{\bar{r}} = 1 \qquad k = 0, \dots, m-1 \tag{Eq. 33}$$

$$x_{\bar{r}} \in \{0,1\} \ \ \bar{r} \in R \tag{Eq. 34}$$

4.2 Algorithm Description

In this section, the algorithm developed to solve the problem defined in the previous section will be explained in detail.

The algorithm is in part an extension to (Edelkamp and Gath, 2013) algorithm solving the TSP by using NMCTS with policy adaptation.

The core function in the search process is the search function.

```
procedure search(level,iterations)
    TOUR R
    R.score = MAX VALUE
    if level = 0 then
        R = ROLLOUT()
        Runs ++
    else
        policy[level] ← polGlobal // polGlobal denotes the global policy
    for all i = {1, ..., iterations} do// for all iterations in current level
        R ← search(level - 1, iterations)
        if  R.score < best.score then
            best.score ← R.score
            best.tour ← R.tour
            adapt(best.tour, level)
        end if
    if (Time Limits exceeded or Max Number of Runs excceded) then
        return best
    end if
    end for
    polGlobal ← policy[level]
    end if
 return best
 end procedure
```

Algorithm 4 NRPA search function

The *search* function presented in (Algorithm 4 NRPA search function) is a recursive function and is the main procedure of this implementation. It shows how the level-specific search policy of a level is adapted if a better solution at level − 1 has been found. As soon as all iterations have been executed, the global policy, applied by the *rollout* function is overwritten. If the algorithm reaches the lower level 0, the search function performs a rollout. A simplified pseudocode of the rollout function is available in APPENDIX 2. Pseudocode of the rollout function). The procedures followed by the rollout function are explained in the following sections. In order to understand how the search and the rollout functions work, some variables used will be introduced.

$Policy: \{0, \ldots, N-1\} \times \{0, \ldots, N-1\} \rightarrow a_{i,j}$

Global policy is a backup three-dimensional policy matrix that keeps a copy of the global policy of at each level.

Vehicle choices policy vector$\{0, \ldots, m-1\} \rightarrow a_{i,j}$

Instead of a policy matrix for selecting vehicles, a policy vector of length m has been implemented. The reason behind choosing one dimensional vector is that there is no direct correspondence between vehicles, unlike the case between nodes representing stops, where the distance between the nodes and the type of the stop contribute to the values of the policy matrix.

N refers to the total number of nodes in the search graph. $N = 2(m + n)$

Checked vector with the size $[0, \ldots, 2(m+n)-1]$ is a vector designed to prevent a costly backtrack operation in case the inclusion of a node in a tour resulted in a new violation. The check vector marks the nodes that have already been evaluated during search for candidate nodes to include in a vehicle's tour. This variable is reset each time a new vehicle is selected.

Visited vector with the size $[0, \ldots, 2(m+n)-1]$ is a Boolean vector that marks all the nodes that have already been visited by the solver and now part of the tour that represents the current solution.

Weight vector according to the definition, each pickup node has a weight $w_i \geq 0 \; for \; i \in P \; and \; w_i = -w_{i-1} \; for \; i \in D$. A Successor set contains all the candidate nodes that can be selected to extend the search starting from the current node. The choice between these nodes will be based on the policy and will be explained in more detail in section 4.2.1

The algorithm will count the number of the times when any of the problem constraints were violated during the search process and this value will be kept in a violation counter variable.

4.2.1 Successor Selection

After the starting point of the vehicle, each following node of a vehicle's tour has to be selected according to the problem rules in a way that will guarantee a maximum number of customer's requests met in the most efficient way.

Here we are going through the nodes selection process of the implemented algorithm.

In order for a node i to be considered as a valid move from current node n it has either to be:

- A pickup node that has not been checked yet;
- Or a delivery node for an order that has already been picked up.

Considering the specificity of the problem tackled help identify areas where performance can be enhanced. One of these improvements for the VRP in our case is to simply prioritise choosing a successor node to the current node by first looking for candidates among the pickup or delivery nodes in the same geographical position as the current node:

```
if (successors = 0)
    for all nodes i < N
       check_valid_move(i)
          if  node i location = currentnode location
                Add i to the possible moves
```

If there were no nodes located in the same geographical location as the current node, this would lead to having an empty successors set. Therefore the search for successors will be directed toward any valid possible move starting from the current node:

```
if (successors = 0)
    for all nodes i < N
       check_valid_move(i)
                Add i to the possible moves
```

If the successor set is still empty because of no candidate nodes satisfying the valid moves check, this means that the search for nodes to extend the tour of the current tour has come to an end, and the next step would be choosing the next vehicle. However, if there are no more vehicles available, the number of violations increases with respect to the number of the nodes not included in the overall tour:

```
        violations += N - tourSize;
        break SearchLoop;
```

Assuming that the successor set is not empty, a choice between the successors will be made according to the current policy. This inclined selection of a successor node is similar to the roulette wheel (sometimes called fitness selection) in genetic algorithms (more on this selection method is in the work of (Lipowski et.al , 2012)).

```
      for (all successors) do
            probability[i] = Math.exp(policy[node][moves[i]])
            sum += probability[i];
      end for

       mrand = random.nextDouble() * sum

       i = 0
       sum = probability [0]
       while (sum < mrand)
       sum += probability [++i]

            node = moves[i]
```

If the successor is determined, the tour will be extended by one node (either pickup or delivery location), and all violations are counted.

Adding the new node to the tour, might lead to violation of one of the rules, therefore, a check will take place to determine the consequences of adding this node on the maximum vehicle's capacity or time windows. Sometimes, extending the tour with a node might not immediately result in a violation. That is exactly the case when there are several orders located in the same geographical place needed to be handled together. That increases the service time, which is proportional to the number of these orders in the same place.

In case that extending the tour with the selected node will result in an increase in the violations, the added node needs to be removed by directly eliminating it from the tour if it was a pickup node. Whereas, in the case of a delivery node, the node will be eliminated from the tour as well as its corresponding pickup node of the tour. Removing a delivery node requires looking up its pickup node in the tour $\bar{r} = (t_k \longrightarrow v_1, v_2, \ldots, v_h)$ which is located in a position $x < h$ and removing it. Doing so means that the makespan of the tour has to be recalculated and the load of the current tour has to be adjusted. The distance between the v_{x-1} and v_k and between v_x and v_{x+1} will be substituted by the distance between v_{x-1} and v_{x+1}. The process is guaranteed not to increase the makespan of the tour providing that the triangle equality is assured as we first assume.

4.2.2 Rollout Policy and the adapt function

The policy matrix is used during the search process to guide it towards promising directions, and the values stored in the policy matrix determines the likelihood of choosing a certain move between candidate nodes. Three other well-known heuristics for VRPs derived from (Solomon et al., 1987) contribute to the choice of the successor node:

1. the distance from the last visited node to the next node,
2. the amount of wasted time, if a node is visited too early, and

3. the remaining time until the latest possible visiting time of the following node.

The policy matrix has the size of N * N of floating-point values. That matrix's values are updated by the adapt function whenever the search led to a better solution than the current best. The adapt function follows a procedure similar to the ones followed in (Rosin, 2011), (Edelkamp and Gath, 2013) and (Edelkamp et al., 2013) as it adjusts the policy values representing the qualification -how right is it- to choosing each of the nodes based on the best-known solution so far. However, a noted difference here is that the adapt function also performs the policy adaptation processes for the vehicle choice policy vector as well.

4.2.3 Vehicle Selection

Once one vehicle finishes its tour, the algorithm will try to complete the tours of the remaining vehicles in the fleet if there were any left. As the set of possible successors remaining vehicles are fixed, a choice based on the current vehicles policy vector is applied also using a choice based on roulette wheel selection just like the case with node selection. Once a new vehicle has been determined, the overall tour is extended by one stop (The new vehicle start location stop). The tour parameters will be reset; all violations are reset, makespan or the time elapsed so far, as well as the vehicle load will all be reset to zero.

Also, all the checked nodes not visited can be set back to 'not checked'. Performing this action means that the nodes not integrated into the tours of the previous vehicle(s) can be re-assessed for inclusion in the new vehicle tour. In case there were no more vehicles left in the fleet, the search terminates.

Once the search terminates, any node that was not checked is counted as a violation. Also, the difference between the total number of nodes and size of the tour resulting from the search will be counted as violations.

4.2.4 End of Search

The rollout function terminates when all nodes have been checked for inclusion in the tour and there are no more available vehicles to choose from. Whereas, the overall search terminates once the maximum number of runs has been reached. The result of the search would be a tour vector having the values representing the number of nodes. The sequence of their occurrences in the resulting tour represents their precedence sequence.

4.2.5 Parameters and Configurations

Series of pre-tests have been conducted to figure out the parameters that give the best results on different series of tests. In other words, the parameters that provide the balance between exploration and exploitation rates.

The problem minimizes the number of vehicles involved in the transportation process as its priority number one. Meeting the maximum number possible of demands comes next.

Due to the reason above, the penalty cost of adding a new vehicle was set to six times the penalty of missing an order. This difference helps in directing the algorithm towards minimizing the number of vehicles in the early stages of calculations first, and then concentrating -during the remaining number of rollouts- on fitting the maximum number of orders in the tour of each vehicle.

The pre-tests did not consider changing the learning rate parameter of the policy used in the ‘adapt’ function. Instead the value of the learning rate alpha that has been used in a previous publication using NRPA -see (Edelkamp and Gath, 2013), (Cazenave and Teytaud, 2012) (Rosin, 2011)- was adopted.

The pre-tests led to the adoption of the following configuration:
Number of the iterations in the search tree: 128,
Depth level of the search tree: 5,
Missing an order violation cost: $1 * 10^8$,
Violation of adding a vehicle: $6 * 10^8$,
Learning rate in the adapt function: 1.

5 RESULTS AND DISCUSSION

5.1 Introduction

To validate the performance of the implemented algorithm presented in the previous chapter, we discuss here the results of running the solver on several datasets obtained from industry. The chapter also presents how evaluations on standard benchmarks can be applied to the developed approach. The chapter further lists future perspectives of the research.

5.2 Evaluation with Data from Industry

Obtaining real data and info that is directly related to real-world applications has a potential to challenge and test the developed algorithm capability.

An agreement has been signed with XTL Kommunikationssysteme GmbH to provide this data. XTL Kommunikationssysteme GmbH is a German company placed in Bremen and working in delivering innovative solutions based on artificial intelligence for the transport logistics industry, and they have kindly offered the data sets required for testing and later validation of the quality of results obtained.

In 0APPENDIX 1. NDA agreement the reader can find a copy of the Non-Disclosure Agreement that was signed with (XTL Kommunikationssysteme GmbH) to get test data for the developed solver.

5.2.1 The input file

The input data obtained to verify the applicability of the algorithm is written as JSON file. Each file is structured with the following information:

Orders information: each order is defined by its pickup location and delivery locations, its weight as well as the time windows in which each order has to be serviced.

The fleet of the vehicles that are available from the logistics company to serve the orders with a description of each vehicle containing its: start and end locations and the average speed.

5.2.2 Scenarios and results:

All experiments run a PC Laptop equipped with a 5th generation Core i5 Intel® processor model 5200u with two cores running at 2.2 GHz and 16GB of RAM and Win 10 Professional OS.

Each experiment results in two kinds of outputs, first a tour plan containing the orders of the visited stops as well as their location and the overall distance covered. This tour plan is encoded as a Keyhole Markup Language (KML) file which can be illustrated on a map.

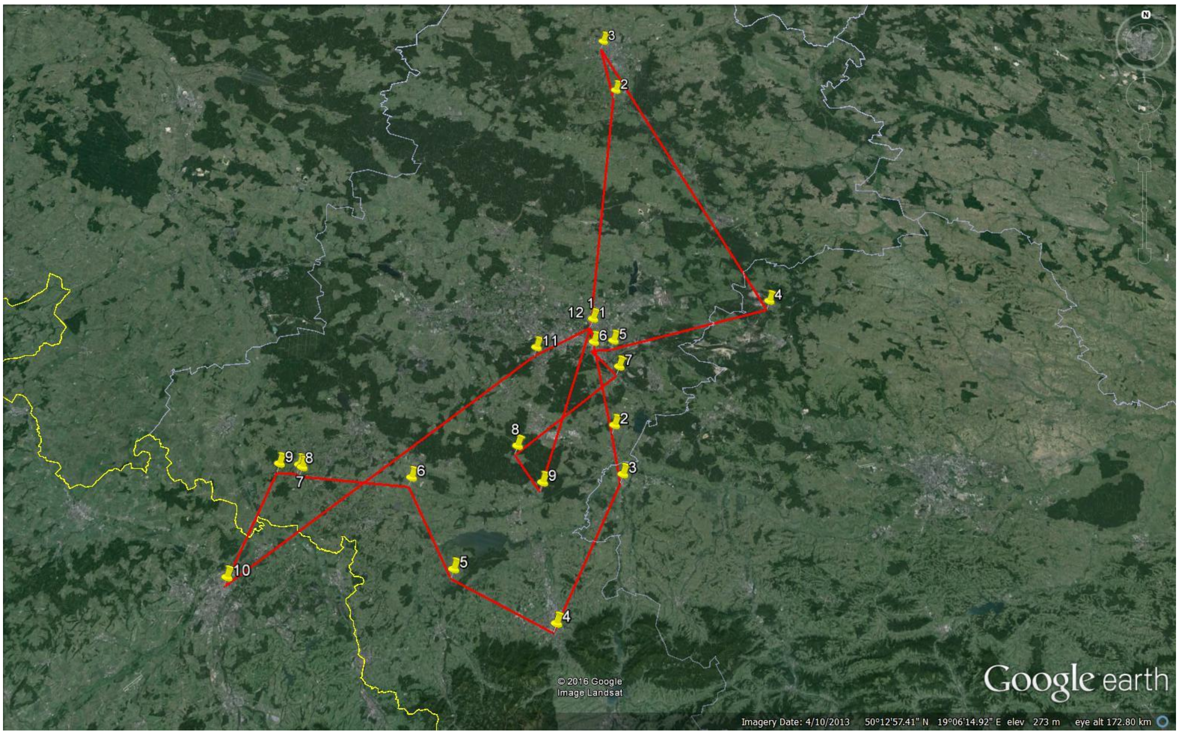
3
2
4
12
1
1
11
6
5
7
2
8
9
3
9
8
7
6
5
10
4
© 2016 Google
Image Landsat
Google earth
Imagery Date: 4/10/2013 50°12'57.41" N 19°06'14.92" E elev 273 m eye alt 172.80 km

Figure 9 Shows a sample of the KML file output listing different stops with the orders of the visit

The other type of output is a comma separated file (CSV) summarising the total distance of the calculated tour in kilometres, listing the number of orders served and unserved and the number of unique stops (stops not at the same location) included in the tour.

The total distance of a tour is the sum of distances of all sub-tours of participating vehicles.

In each of the scenarios listed below, the results are obtained from running the algorithm with the exact parameters and configurations settings mentioned in 4.2.5.

Results presented of scenarios 2 to 5 are 20 randomized samples of multiple runs.

Scenario 1 Investigating policy matrix initialization
Number of orders: 18;

Number of available vehicles: 2 identical vehicles.

After investigating in the pre-tests the best parameters to run the algorithm, this scenario evaluates the effects of policy matrix initialization on the quality of the solution.

The assumption is that giving the initial policy some “guidance” would result in a quicker convergence and an improved overall performance reflected by higher quality solutions.

To evaluate this assumption, the developed solver is given a scenario with two identical vehicles (so the effects of vehicle policy choice will be neutralized). However, two different policy initializations will be applied each time. First, the policy will be initialized with zero values. Second, we apply a heuristic in every node’s entry in the policy matrix. The applied heuristic gives small positive values to the closest corresponding nodes so that policy can guide the search, in the beginning, to look for solutions in the neighbourhood of these nodes. The experiments were set to run 100 times for each initialization technique and results are presented in Figure 10 Results obtained from initialization based on a closest distance heuristic. and Figure 11 Results achieved with initializing policy with 0 values).

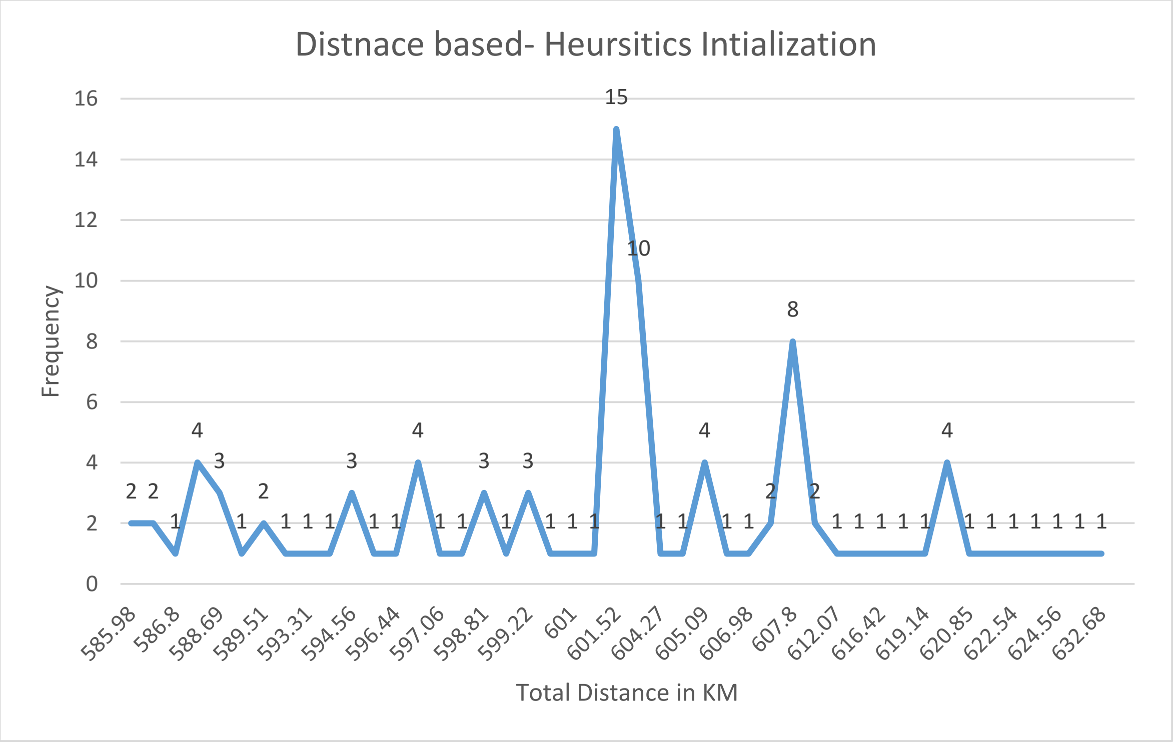

Figure 10 Results obtained from initialization based on a closest distance heuristic.

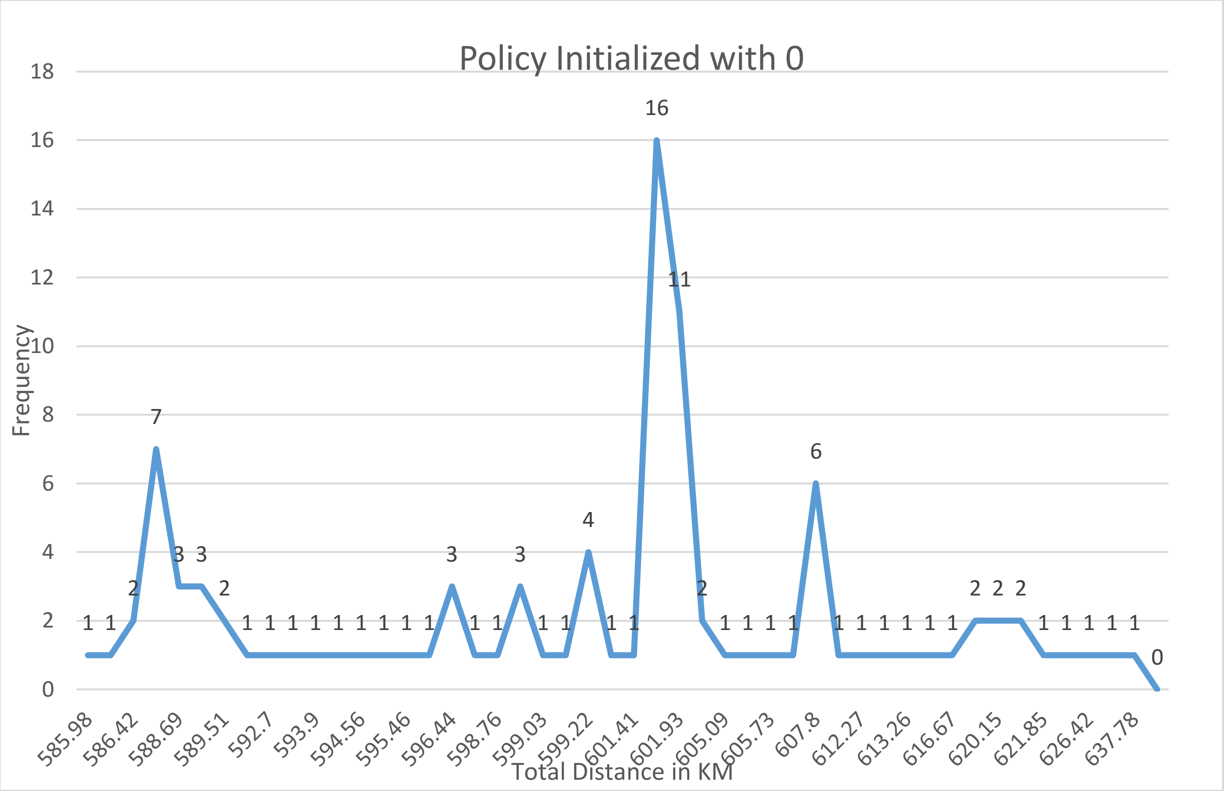

Figure 11 Results achieved with initializing policy with 0 values

Comparing the two graphs, one can see that they show similar overall behaviour in a sense that many results are obtained around a total distance of 600 km. However, a major difference is the number of obtained solutions in the first half of the graph that stands for lower values of the distance covered. Here, more results were obtained when initialization policy with zero than compared to the other initialization.

The explanation is that looking in the neighbourhood of the nearest next stop may give better results for small sized problems. However it is in the interest of the search to find a solution in unexamined neighbourhoods of larger problems (increasing exploration rate), especially in highly constrained problems.

Since a minimized overall distance covered in tours is preferred, the conclusion drawn is that starting the search with values of zero stored in the policy matrix led to more occurrences of preferred results. Based on that, this method of initialization will be adopted in all further runs of the solver.

Scenario 2
Number of orders: 13;

Number of available vehicles: 10 differ in starting locations, end locations, operation times and capacities, but all have the same average speed.

Number of vehicles in the resulting solution: 2.

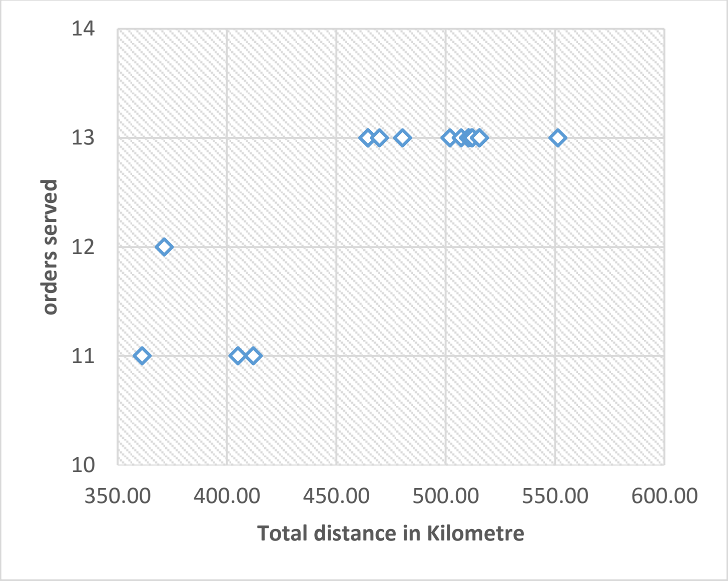

Figure 12 Scenario 2

Scenario 2 utilizes two vehicles out of the available pool of ten. This scenario contains a relatively small number of orders. The samples taken show that the algorithm could find a valid solution satisfying all orders using two vehicles in 16 out of 20 cases. In the remaining four cases, not all orders were met, three of the remaining four cases produced results with two unserved orders and one solution with one unserved order. Those cases where not all orders were met represent local optima in the neighbourhood of the search, whereas missing an order or two would result in a less overall cost -during evaluation- than adding a new vehicle.

Scenario 3
Number of orders: 18;

Number of available vehicles: 10 differ in starting locations, end locations, operation times and capacities, but all have the same average speed.

Number of vehicles in the resulting solution: 2.

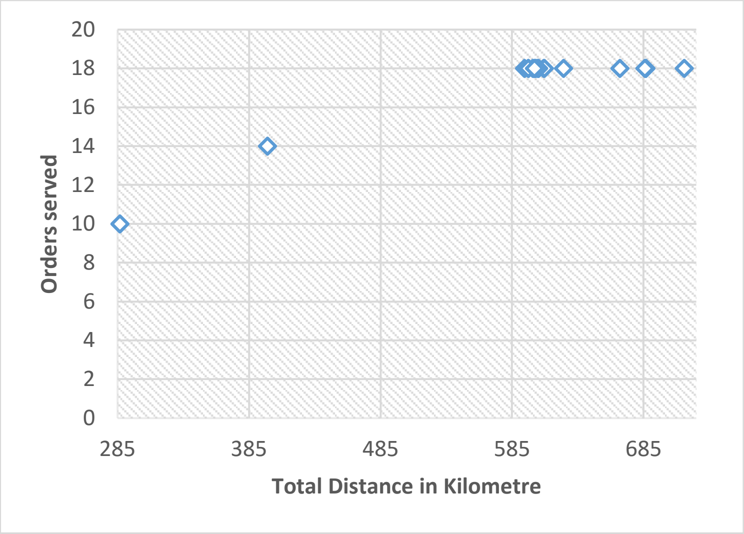

Figure 13 Scenario 3

A similar observation as in scenario two is witnessed here. The solver was able to compute solutions using two vehicles out of the available ten. In the two local optima one has 4 unserved orders, and the other missed 8.

It can be noted here that the increased number of orders in the problem led to an increase in the number of unserved orders in the computed solutions.

Scenario 4
Number of orders: 22;

Number of available vehicles: 10 differ in starting locations, end locations, operation times and capacities, but all have the same average speed.

Number of vehicles in the resulting solution: 2.

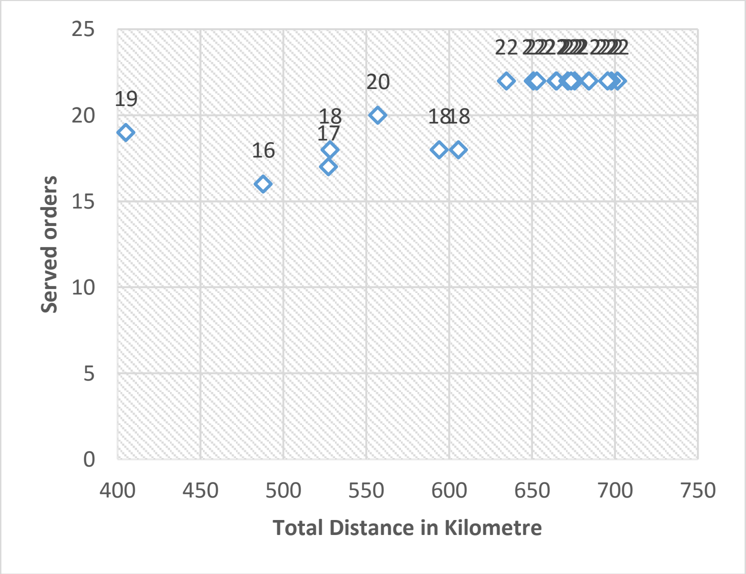

Figure 14 scenario 4

With 22 orders to serve, the differences between solutions meeting all the demanded orders decrease as they all concentrate in a relatively small area(range) regarding the total distance of the tour. In the same time, more orders to be transported, meant a decreased possibility to have all orders met with more occurrences of local optima missing between six (maximum) to two (minimum) orders.

Scenario 5
Number of orders: 305;

Number of available vehicles: 34 differ in starting locations, end locations, operation times and capacities, but all have the same average speed.

This scenario was run to test the ability of the developed algorithm to solve large problem sets. However, the implemented solver could not find any valid solution to the given problem within the number of allowed iterations.

One direct criticism to the tests performed would be that they do not include tours with orders in the range between 50 to 200 orders for example. Orders in this range of sizes would be necessary to better understand the performance of the algorithm in midsized problems.

5.3 Evaluation with Benchmarks

Usually, solvers to problems of transport logistics like the TSP and VRP are assessed through standardized benchmarks like the ones mentioned by (Solomon, 1987) for VRPTW. However, and due to the nature of the problems this solver is intended to solve, in addition to the lack of comprehensive benchmarks that evaluate the performance of multiple variants of the VRP problem. Together, and in the end also due to the shortage of time, the evaluation using this method has not been adopted in this work.

It is indeed possible to run standardized benchmarks using the developed approach. However, each of the irrelevant constraints has to be removed from the implemented algorithm to save computational time. For example, parts handling capacity constraints have to be omitted in non-capacity constrained problems; another example is the need of conversion from pickup and delivery notation to a simpler notation where all vehicles are assumed loaded already in the depot, and their route consisting of only delivery stops.

Also, a notable change in the data structure implemented in the algorithm has to be considered to be more efficient running the simple data format that these benchmarks use.

A popular standard benchmark for VRPTW is the Solomon benchmark, which can be found in http://web.cba.neu.edu/~msolomon/problems.htm

5.4 Research Outlook

This research has outlined the usage of Monte Carlo Search and in particular, the Nested Monte Carlo Search with Policy adaptation to solve the VRP. Still there exist many aspects where this research could be improved in the future, and this section will mention few.

First, the research has paved the way to future work on solvers that adapt more comprehensive approaches towards solving VRP with multiple constraints. Examples of solvers that can be relatively easily adapted to handle such complexities are the solvers based on genetic algorithms. Doing so will allow comparing different approaches contributing in finding a better solver for highly constrained problems. Moreover, the implementation of solvers using different algorithms will push towards developing standardized benchmark test for hybrid problems just like the ones available in the literature.

The research conducted in this thesis can be further extended by extensively studying the different parameters of the NRPA algorithm and its impact on the overall solution quality and evaluate the best parameters configuration.

Considering that Monte Carlo methods can be improved by providing more processing power, future work might look into the parallelization of the algorithm execution to compute better results, since a major drawback of the developed approach is its poor performance with a large dataset.

A more thorough study would be required to develop a complete application for the transportation logistics sector dealing with the dispatching problem from the early planning stage to scheduling to live-time tracking and dynamic adaptation to the changes in the environment during operation. Such a comprehensive approach requires studies in domains like Internet of Things (IoT), dynamic scheduling, and communication protocols suitable for such implementation.

Throughout this research, it is assumed while finding a solution to the dispatching problem that Euclidian distances between stops are fixed; that is a limitation to an accurate real-world simulation, in which sometimes vehicles have to alter their pre-defined route due to different reasons like traffic jams, or closed roads to name but a few. Future work on this problem should consider the variability of these distances, and this leads to the consideration of dynamic problems.

6 SUMMARY

The goal of the of this research was to develop an approach that handles complex real-world processes in the industry, while at the same time contributing to an enhanced efficiency of the transport sector and providing a contribution to sustainability.

Chapter 1 explained the background and the motivation for this research and introduced the research question and limitations.
Chapter 2 has briefly introduced common dispatching problems faced by the transportation logistics. Chapter 3 focused on TSP and VRP with their common variations and methods to solve them. It concluded that filling the identified gap in literature could be done by building on previous research in the field and implementing an NRPA approach to solving VRP.
Considering the previous works utilizing NRPA methods for complex TSP, a new approach was proposed in Chapter 4 which led to implementing a new solver for the vehicle routing problem. The proposed solver handles multiple constraints and considers vehicle choice.
To evaluate the validity of the implementation presented in Chapter 4, several tests using data from the industry have been conducted. Chapter 5 presents the results of these tests and discusses them providing conclusions and prospects for further research related to the field.

REFERENCES

BALDACCI, R., TOTH, P., and VIGO, D., 2010. Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, Vol. 175, No. 1, p. 213–245.

BROWNE, C.B., POWLEY, E., WHITEHOUSE, D., LUCAS, S.M., COWLING, P.I., ROHLFSHAGEN, P., TAVENER, S., PEREZ, D., SAMOTHRAKIS, S., and COLTON, S., 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 4, No. 1, p. 1–43.

CAZENAVE, T., 2009. Nested Monte-Carlo Search. *International Joint Conference on Artificial Intelligence*, p. 456–461.

CAZENAVE, T. and TEYTAUD, F., 2012. Application of the nested rollout policy adaptation algorithm to the traveling salesman problem with time windows. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 7219 LNCS, p. 42–54.

COOK, W., 2012. *In pursuit of the traveling salesman mathematics at the limits of computation*. Princeton University Press.

EDELKAMP, S. and GATH, M., 2013. Solving Single Vehicle Pickup and Delivery Problems with Time Windows and Capacity Constraints using Nested Monte-Carlo Search. *5th International Conference on Agents and Artificial Intelligence (ICAART)*, p. 324 – 329.

EDELKAMP, S., GATH, M., CAZENAVE, T., and TEYTAUD, F., 2013. Algorithm and knowledge engineering for the TSPTW problem. *Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Scheduling, CISched 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, p. 44–51.

EDELKAMP, S., GATH, M., GREULICH, C., HUMANN, M., HERZOG, O., and LAWO, M., 2016. Commercial Transport: Proceedings of the 2nd Interdiciplinary Conference on Production Logistics and Traffic 2015. U. Clausen, H. Friedrich, C. Thaller, and C. Geiger, eds., p. 427–440.

GATH, M., 2015. Optimizing Transport Logistics Processes with Multiagent-based Planning and Control, Vol. 3.

GAUR, D.R., MUDGAL, A., and SINGH, R.R., 2013. Routing vehicles to minimize fuel consumption. *Operations Research Letters*, Vol. 41, No. 6, p. 576–580.

GUTIN, G. and PUNNEN, A.P., eds., 2007. *The Traveling Salesman Problem and Its Variations*. Boston, MA: Springer US.

LAPORTE, G., 1992a. The Traveling Salesman Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, Vol. 59, p. 231–47–231–47.

LAPORTE, G., 1992b. The Vehicle Routing Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, Vol. 59, No. 2, p. 231–247.

LENSTRA, J.K. and KAN, A.H.G.R., 1981. Complexity of vehicle routing and scheduling problems. *Networks*, Vol. 11, No. 2, p. 221–227.

LIPOWSKI, A. and LIPOWSKA, D., 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, Vol. 391, No. 6, p. 2193–2196.

PAPADIMITRIOU, C.H., 1977. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, Vol. 4, No. 3, p. 237–244.

PISINGER, D. and ROPKE, S., 2007. A general heuristic for vehicle routing problems. *Computers and Operations Research*, Vol. 34, No. 8, p. 2403–2435.

PSARAFTIS, H.N., 1995. Dynamic vehicle routing: Status and prospects. *Annals of Operations Research*, Vol. 61, No. 1, p. 143–164.

RAZALI, N.M. and GERAGHTY, J., 2011. Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. *International Conference of Computational Intelligence and Intelligent System*.

Reducing emissions from transport - European Commission [online], 2016. Available: http://ec.europa.eu/clima/policies/transport/index_en.htm [Accessed 2016-3-15].

RENAUD, J., LAPORTE, G., and BOCTOR, F.F., 1996. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, Vol. 23, No. 3, p. 229–235.

RIMMEL, A., TEYTAUD, F., and CAZENAVE, T., 2011. Optimization of the nested Monte-Carlo algorithm on the traveling salesman problem with time windows. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 6625 LNCS, No. PART 2, p. 501–510.

ROSIN, C.D., 2011. Nested rollout policy adaptation for Monte Carlo tree search. *IJCAI International Joint Conference on Artificial Intelligence*, No. line 22, p. 649–654.

SILVER, D., HUANG, A., MADDISON, C.J., GUEZ, A., SIFRE, L., VAN DEN DRIESSCHE, G., SCHRITTWIESER, J., ANTONOGLOU, I., PANNEERSHELVAM, V., LANCTOT, M., DIELEMAN, S., GREWE, D., NHAM, J., KALCHBRENNER, N., SUTSKEVER, I., LILLICRAP, T., LEACH, M., KAVUKCUOGLU, K., GRAEPEL, T., and HASSABIS, D., 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, Vol. 529, No. 7587, p. 484–489.

SNYDER, L. V. and DASKIN, M.S., 2006. A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, Vol. 174, No. 1, p. 38–53.

SOLOMON, M.M., 1987. Algorithms for and Scheduling Problems the Vehicle Routing With Time Window Constraints. *Operations Research*, Vol. 35, No. 2, p. 254–265.

SOLOMON, M.M., APR, N.M., and SOLOMON, M.M., 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints Published by : INFORMS Stable URL : http://www.jstor.org/stable/170697 REFERENCES Linked references are available on JSTOR for this article : You may need to log in to, Vol. 35, No. 2, p. 254–265.

SOLOMON, M.M., DUMAS, Y., DESROSIERS, J., and GELINAS, E., 1995. An Optimal Algorithm for the Traveling Salesman Problem with Time Windows. *Operations Research*, Vol. 43, No. 2, p. 367–371.

SOONPRACHA, K., MUNGWATTANA, A., JANSSENS, G.K., and MANISIRI, T., 2014. Heterogeneous VRP review and conceptual framework. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Vol. II.

STEFAN EDELKAMP, MAX GATH, CHRISTOPH GREULICH, MALTE HUMMAN, OTTHEIN HERZOG, M.L., 2016. Monte-Carlo Tree Search for Logistics.

THE CLIMATE GROUP, 2008. SMART 2020 : Enabling the low carbon economy in the information age. *Group*, Vol. 30, No. 2, p. 1–87.

TOTH, P. and VIGO, D., 2014. *Vehicle Routing Problem, Methods, and Application*. Society for Industrial and Applied Mathematics.

APPENDIX 1. NDA AGREEMENT

A copy of the Non-Disclosure Agreement with (XTL Kommunikationssysteme GmbH) to gain access to the test data.

INTELLECTUAL PROPERTY ASSIGNMENT AGREEMENT

IN FAVOR OF XTL Kommunikationssysteme GmbH

This Intellectual Property Assignment Agreement (the "Agreement") is made and entered into as 01.02.2015, by and between XTL Kommunikationssysteme GmbH (the "Company") and Ashraf Abdo (the "Recipient") (collectively, the "Parties").

The Parties hereby agree as follows:

1. The Recipient agrees to assign to the Company, or its designee, all right, title, and interest in and to any and all inventions, original works of authorship, developments, concepts, improvements, designs, drawings, discoveries, algorithms, formulas, computer code, ideas, trademarks, or trade secrets, whether or not patentable or registrable under patent, copyright or similar laws, related to the Company's business, which the Intern may solely or jointly conceive or develop or reduce to practice, or cause to be conceived or developed or reduced to practice, with the use of Company's equipment, supplies, facilities, assets, or Company Confidential Information (*see* NONDISCLOSURE AGREEMENT), or which may arise out of any research or other activity conducted under the direction of the Company (collectively referred to as "Intellectual Property").

2. The Recipient understands and agrees that (i) all original works for authorship which are made by the Recipient (solely or jointly with others) within the scope of the Company's business which are protectable by copyright are "works made for hire," and (ii) the decision whether or not to commercialize or market any Intellectual Property is within the Company's sole discretion and for the Company's sole benefit and that no royalty or other consideration will be due to the Recipient as a result of the Company's efforts to commercialize or market any such Intellectual Property.

3. The validity, construction and enforceability of this Agreement shall be governed in all respects by the law of the country of Germany, specifically the city of Bremen. This Agreement may not be amended except in writing signed by a duly authorized representative of the respective Parties. This Agreement shall control in the event of a conflict with any other agreement between the Parties with respect to the subject matter hereof. The failure of either party to enforce its rights under this Agreement at any time for any period shall not be construed as a waiver of such rights.

NONDISCLOSURE AGREEMENT

Recitals

A. The Recipient understands that the Company has disclosed or may disclose information, which to the extent previously, presently, or subsequently disclosed to the Recipient is hereinafter referred to as "Proprietary Information" of the Company.

Operative Provisions

1. In consideration of the disclosure of Proprietary Information by the Disclosing Party, the Recipient hereby agrees: (i) to hold the Proprietary Information in strict confidence and to take all reasonable precautions to protect such Proprietary Information (including, without limitation, all precautions the Recipient employs with respect to its own confidential materials), (ii) not to disclose any such Proprietary Information or any information derived therefrom to any third person, (iii) not to make any use whatsoever at any time of such Proprietary Information except to evaluate internally its relationship with the Disclosing Party, and (iv) not to copy or reverse engineer any such Proprietary Information. The Recipient shall procure that its employees, agents and sub-contractors to whom Proprietary Information is disclosed or who have access to Proprietary Information sign a nondisclosure or similar agreement in content substantially similar to this Agreement

2. Without granting any right or license, the Company agrees that the foregoing shall not apply with respect to any information after five years following the disclosure thereof or any information that the Recipient can document (i) is or becomes (through no improper action or inaction by the Recipient or any affiliate, agent, consultant or employee) generally available to the public, or (ii) was in its possession or known by it prior to receipt from the Company as evidenced in writing, except to the extent that such information was unlawfully appropriated, or (iii) was rightfully disclosed to it by a third party, or (iv) was independently developed without use of any Proprietary Information of the Disclosing Party. The Recipient may make disclosures required by law or court order provided the Recipient uses diligent reasonable efforts to limit disclosure and has allowed the Company to seek a protective order.

3. Immediately upon the written request by the Company at any time, the Recipient will return to the Company all Proprietary Information and all documents or media containing any such Proprietary Information and any and all copies or extracts thereof, save that where such Proprietary Information is a form incapable of return or has been copied or transcribed into another document, it shall be destroyed or erased, as appropriate.

4. The Recipient understands that nothing herein (i) requires the disclosure of any Proprietary Information or (ii) requires the Company to proceed with any transaction or relationship.

5. The Recipient further acknowledges and agrees that no representation or warranty, express or implied, is or will be made, and no responsibility or liability is or will be accepted by the Disclosing Party, or by any of its respective directors, officers, employees, agents or advisers, as to, or in relation to, the accuracy of completeness of any Proprietary Information made available to the Recipient or its advisers; it is responsible for making its own evaluation of such Proprietary Information.

6. The failure of either party to enforce its rights under this Agreement at any time for any period shall not be construed as a waiver of such rights. If any part, term or provision of this Agreement is held to be illegal or unenforceable neither the validity, nor enforceability of the remainder of this Agreement shall be affected. Neither Party shall assign or transfer all or any part of its rights under this Agreement without the consent of the other Party. This Agreement may not be amended for any other reason without the prior written agreement of both Parties. This Agreement constitutes the entire understanding between the Parties relating to the subject matter hereof unless any representation or warranty made about this Agreement was made fraudulently and, save as may be expressly referred to or referenced herein, supersedes all prior representations, writings, negotiations or understandings with respect hereto.

7. This Agreement shall be governed by the laws of the jurisdiction in which the Company is located (or if the Company is based in more than one country, the country in which its headquarters are located) (the "Territory") and the parties agree to submit disputes arising out of or in connection with this Agreement to the non-exclusive of the courts in the Territory.

IN WITNESS WHEREOF, the Parties have executed this Agreement as of the date first above written.

_________, Ashraf Abdo
Signature, First and Last Name

_________, Max Gath
Signature, First and Last Name

_________, Ashraf Abdo
Signature, First and Last Name

_________, Max Gath
Signature, First and Last Name

APPENDIX 2. PSEUDOCODE OF THE ROLLOUT FUNCTION

```
procedure rollout()
        RESET: tour, tourSize,Visited, checked, vehiclesUsed
        RESET: Makespan = currentload = violations = cost = 0

        Vehicle CurrentVehicle = ChooseVehicle()
        Node CurrentNode = StartOf (CurrentVehicle)
        Node PrevNode = CurrentNode

        Visited[CurrentNode] = true
        tour[0] = CurrentNode
        tourSize = 1

        TourLoop: while (true)

            Successors = Find_Successor_Candidates ()

            if (No Successors) // No successors nodes available for
                                // Tour CurrentVehicle indicates
                                // The search of this vehicle subtour
                                // has ended
                if capacity constraints violated of (CurrentVehicle)
                violations ++

                tour[validtourSize++] = EndOf(CurrentVehicle)

                if (CountRemainingVehicles > 0)

                    CurrentVehicle = ChooseVehicle()
                    CurrentNode = vehiclesstart[CurrentVehicle]
                    visited[CurrentNode] = true
                    checked[CurrentNode] = true
                    tour[validtourSize++] = CurrentNode
                    tourlengthOfV[CurrentVehicle]++
                    tourSize++
                    Prev = vehiclesstart[CurrentVehicle]
                    Makespan = EarliestVisitingTime[CurrentVehicle]

                        for all checked nodes && not visited  do
                            Checked[] = false

                    RESET: violations, CurrentLoad

                    // After picking up new vehicle
                    // Proceed to the main search loop
                    continue TourLoop

                else //No More vehicles available
                    for all nodes not checked  do
                        violations++ // all non checked nodes are missed orders!
                    Violations + = N - TourSize
                    break TourLoop
                end if

            //  selecting next node from candidates
            CurrentNode = Select_Next_Move(Successors)
            checked[CurrentNode] = true
            tourSize++

            define: oldviolations = violations

            if capacity constraints violated of (CurrentVehicle)
                violations ++
            end if
            if  CurrentNode time window violated
                violations ++
            end if
            if  CurrentVehicle window violated
                violations ++
            end if
            // Add the checked node to Tour
            // only if it does not violate
            if (oldviolations == violations)

                extend the tour with node
                prev = node
                visited[node] = true
                visitingVehicle[node] = CurrentVehicle
                tour[validtourSize++] = node

            else if pickup node
                    checked[node + 1] = true
                    violations++
            end if

            else if delivery node
                tour = remove_correspondent_pickup(node, tour)
            end if
            end if
        end while
        NonUsedvehicles = CountRemainingVehicles()

return  ( - Vehicle violation penalty * (NonUsedVehicles - V)
          + Order violation penalty * violations + cost)
```

APPENDIX 3. TABLES OF RESULTS

Scenario 2

Number of orders: 13

Number of available Vehicles: 10 differ in starting locations, end locations, operation times and capacities, but all have the same average speed.

Orders Served	Unserved orders	Vehicles Used	stops	Total Distance (Kilometre)
13	0	2	17	515.48
13	0	2	17	464.58
13	0	2	17	469.82
11	2	2	14	405.05
13	0	2	17	502.01
12	1	2	15	371.46
13	0	2	17	510.53
13	0	2	17	515.48
13	0	2	17	551.28
13	0	2	17	510.53
13	0	2	17	464.58
13	0	2	17	507.06
13	0	2	17	512.27
11	2	2	14	412.04
13	0	2	17	511.84
11	2	2	14	361.22
13	0	2	17	510.47
13	0	2	17	480.38
13	0	2	17	515.48
13	0	2	17	464.58

Table 1- Twenty random samples from results obtained by solving problem of scenario #1

Scenario 3

Number of orders: 18

Number of available Vehicles: 10 differ in starting locations, end locations, operation times and capacities, but all have the same average speed.

Orders Served	Unserved orders	Vehicles Used	stops	Total Distance (Kilometre)
18	0	2	22	609.05
18	0	2	22	597.41
18	0	2	22	602.89
18	0	2	22	594.28
18	0	2	23	666.82
18	0	2	23	715.77
18	0	2	22	686.74
14	4	2	17	398.91
18	0	2	22	609.41
18	0	2	22	686.74
18	0	2	22	604.65
18	0	2	22	605.09
18	0	2	22	624.08
18	0	2	23	691.41
18	0	2	22	602.89
18	0	2	22	604.65
18	0	2	22	685.95
18	0	2	22	600.62
18	0	2	22	601.94
18	0	2	22	597.41

Table 2 Twenty random samples from results obtained by solving problem of scenario #3

Scenario 4
Number of orders: 22
Number of available Vehicles: 10 differ in starting locations, end locations, operation times and capacities, but all have the same average speed.

Orders Served	Unserved orders	Vehicles Used	Number of Unique stops	Total Distance (Kilometre)
22	0	3	29	701.52
22	0	3	28	634.54
22	0	3	27	664.41
18	4	3	23	605.5
18	4	3	23	594.08
20	2	3	25	556.97
22	0	3	28	675.67
16	6	3	21	487.87
18	4	3	23	528.14
22	0	3	28	684.24
17	5	3	22	527.18
22	0	3	27	650.81
22	0	3	28	664.79
22	0	3	28	652.82
22	0	3	28	684.31
19	3	3	24	405.01
22	0	3	28	697.9
22	0	3	28	671.6
22	0	3	28	695.61
22	0	3	28	673.57

Table 3 Twenty random samples from results obtained by solving problem of scenario #4