1 INRODUCTION

1.1 Overview

Supply Chain Management could be defined as the practice of analyzing all aspects of acquiring, storing, moving, and delivering materials from the time they are acquired through any conversion or production processes through to the time final products are used or sold. A company's supply chain may consists of geographically dispersed facilities where raw materials, intermediate products, or finished products are acquired, transformed, stored, or sold, and transportation links connecting the facilities along which products flow.

Logistics management is that part of the supply chain which plans, implements and controls the efficient, effective forward and reverse flow and storage of goods, services and related information between the point of origin and the point of consumption in order to meet customers' requirements. Depending on the industry sector, supply chain logistics costs account from 5% to 50% of a product's total landed cost. The Vehicle Routing Problem (VRP) is an important problem occurring in many distribution systems.

Many companies are faced with problems regarding the transportation of people, goods or information. This is commonly denoted as routing problems. Indeed they not only model the problems of collection and delivery of goods, but, more generally, appear as a key ingredient in many transportation systems, such as those for solid waste collection, street cleaning, bus routing, dial-a-ride systems, routing of maintenance units, transports for handicapped. Another area in which very similar problems play a relevant role is modern telecommunication networks, even if here we find "routing" and not "vehicle routing" problems. As the world economy turns more and more global, transportation is becoming more important. And with the current energy and economic crisis, conserving resources is at the utmost priority.

Environmental Accounts published by the Office for National Statistics in the UK show that, on a UK resident's basis, greenhouse gas emissions fell 1.4 per cent between 2005 and 2006 to 724.5 million tonnes of CO2 equivalent(Office of National Statistics, News Release, 2006). Between 2005 and 2006 greenhouse gas emissions from the non-household sector decreased

by 1.1 per cent to 572.8 million tonnes of CO2 equivalent. This was largely driven by a fall in emissions from the transport and communications sector due to changes in the structure of the UK shipping industry. If shipping industry emissions are removed from the data the year on year change in emissions from the non-household sector rose 0.2 per cent.

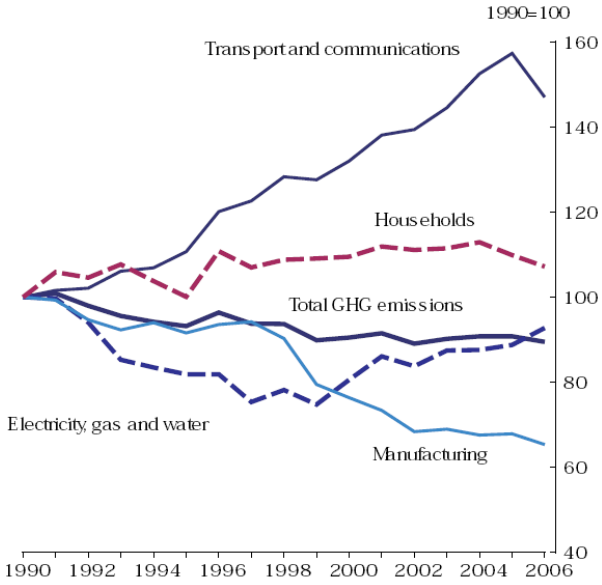

Figure 1 Greenhouse gas emissions in UK

Greenhouse gas emissions from the non-household sector accounted for 79.1 percent of all emissions in 2006. The transport and communications industries were one of the most significant non-household contributors to greenhouse gas emissions in 2006, responsible for 15.7 per cent (113.8 million tonnes of CO2 equivalent) and 13.3 per cent (96.3 million tonnes of CO2 equivalent) respectively. Emissions from the road transport industry show a small year on year increase of 0.4 per cent but at 190.9 million tonnes of CO2 equivalent this is 17.9 per cent above the 1990 level.

In practice, vehicle routing may be the single biggest success story in operations research. For example, each day 103,500 drivers at UPS follow computer-generated routes. The drivers visit 7.9 million customers and handle an average of 15.6 million packages.

Solving different kinds of the Vehicle Routing Problem is an important area of Operations Research. Achieving improvement of only a small percentage may result in large savings and reduce the strain on the environment caused by pollution and noise. The cost of

implementing a solution requires analysts and developers who understand the specific problem for the company. This initial cost of implementation is still a major drawback for companies to take the step towards implementing a solution. It is also viewed that the business is dynamic and parameters might change, which would render the current implementation inefficient. This will cause the company to reinvest into a new solution again.

Smaller companies are not even in the position to consider such an investment. Smaller companies also tend to be more flexible in their service, which let to their problems not being, tied a specific type of method for the solution. The approach of *good-enough* results in better solutions than currently implemented for these types of problems.

We consider the Vehicle Routing Problem in which a fleet of vehicles must service known customer demands for a single commodity from a common depot at minimum cost. This difficult combinatorial problem contains the Travelling Salesman Problem as special case. The problem can consist of multiple constraints such as heterogeneous fleet, multiple time windows and peak and off-peak travel times. In a *pure* routing problem there is only a geographic component, which can be represented by a graph on a two dimensional space. In more realistic routing problems, scheduling plays a major role. Scheduling is represented by the time component.

This study will create a solution for the VRP and some variants with the help of evolutionary algorithms implemented in parallel and build on an adaptive object model approach. The challenge is to combine these methods into one method to find a *good-enough* solution in *acceptable* time. The problems in research are often more simplistic than real-life problems. This research model the approach of flexible methods which would let to an adaptive implementation. A number of important basic models exist and is used as representative problem to assist in the investigation.

The rest of this chapter will discuss the some background of these methods in more detail.

1.2 Background on VRP

One of the most significant problems of supply chain management is the distribution of products between locations, most known as the Vehicle Routing Problem (VRP). The vehicle routing problem is one of the most challenging problems in the field of combinatorial

optimization. Dantzig and Ramser first introduced the VRP in 1959. In a short paper published in Management Science in October 1959 they wrote:

This paper is concerned with the optimum routing of a fleet of gasoline delivery trucks between a bulk terminal and a large number of service stations supplied by the terminal. (Omitted) A procedure based on a linear programming formulation is given for obtaining a near optimal solution. The calculations may be readily performed by hand or by an automatic digital computing machine. No practical applications of the method have been made as yet. A number of trial problems have been calculated, however. (Dantzig and Ramser, 1959)

They proposed the first mathematical programming formulation.

There has been since then a steady evolution in the design of solution methodologies, both exact and approximate, for this problem. In 1964 Clarke and Wright proposed an effective greedy heuristic that improved Dantzig and Ramser approach. Since then, hundreds of models and algorithms were proposed for the optimal and approximate solution of the different versions of the VRP. Vehicle Routing Problems are amongst the most important Combinatorial Optimization problems, because of their difficulty as well as their practical relevance. Yet, no known exact algorithm is capable of consistently solving to optimality instances involving more than 50 customers and often requires relative few side constraints.

Since the Dantzig and Ramser paper appeared, work in the field has exploded dramatically. Today, Google Scholar search of the words 'Vehicle Routing Problem' yields more than 24,800 entries. The June 2006 issue of OR/MS Today provided a survey of 17 vendors of commercial routing software, whose packages are capable of solving average-size problems with 1,000 stops, 50 routes, and two-hour hard-time windows in an execution time of 2 to 10 minutes. (Golden, Raghavan and Wasil, 2008)

While much has been documented about the VRP in major studies that have appeared from 1971 (starting with *Distribution Management* by Eilon, Watson-Gandy, and Christofides) to 2002 (ending with The Vehicle Routing Problem by Toth and Vigo), there are important advances and new challenges that has been raised in the last 5 years or so due to technological innovations such as Global Positioning Systems (GPS), Radio Frequency Identification (RF ID), and parallel computing. The portfolio of techniques for modelling and solving the standard, capacitated VRP and its many variants has advanced significantly. Researchers and

practitioners have developed faster, more accurate solution algorithms and better models that give them the ability to solve large-scale problems.

There are several main survey papers on the subject of VRP's (Toth and Vigo, 2001). A classification scheme was given by Desrochers, Lenstra and Savelsbergh (1990). Valle et al (2009) implements an Integer Programming Formulation, a Branch-and-cut method and a Local Branching to solve a non-capacitated Vehicle Routing Problem. Kallehauge (2008) review the exact algorithms proposed in the last three decades for the solution of the vehicle routing problem with time windows (VRPTW). The exact algorithms for the VRPTW are in many aspects inherited from work on the travelling salesman problem (TSP).

Yeun et al (2008) provides an overview of the methods used in solving the classical VRP, the Capacitated VRP, the VRP with Time Windows and the VRP with Pickup and Delivery. From his study, it is clear the methods used as old as 1995 is still valid in solving the problem today.

Besides being one of the most important problems of operations research in practical terms, the vehicle routing problem is also one of the most difficult problems to solve. The problem is to design routes for the vehicles so as to meet the given constraints and objectives minimizing a given objective function. VRP is a generalization of the travelling salesman problem (TSP), therefore is NP-Hard. The VRP has a finite number of feasible solutions. The VRP solution space increase exponentially as the number of customers increases. The travelling salesman problem is the VRP with one vehicle with no limits, no depot (or any depot), customers with no demand.

In the m-TSP problem, m salesmen have to cover the cities given. Each city must be visited by exactly one salesman. All salesmen start from the same city (the depot) and must end their journey in this city again. We now want to minimize the sum of distances of the routes. The VRP is the m-TSP where a demand is associated with each city, and each salesmen/vehicles has a certain capacity (not necessarily identical). The sum of demands on a route cannot exceed the capacity of the vehicle assigned to this route. As in the m-TSP we want to minimize the sum of distances of the routes. Note that the VRP is not purely geographic since the demand may be constraining.

1.2.1 VRP variants

There exist a number of VRP generalizations.

- CVRP – Capacitated Vehicle Routing Problem: The vehicles have limited carrying capacity of the goods that must be delivered.
- VRPTW - Vehicle Routing Problem with Time Windows: The delivery locations have time windows within which the deliveries (or visits) must be made.
- VRPLC - Vehicle Routing Problem Length Constraint: The routes are limited to a specific length.
- VRPBTW - Vehicle Routing Problem with Backhauling and Time Windows: Backhauling is part of the problem.
- MCVRPTW – Multi Compartment Vehicle Routing Problem with Time Windows: Multiple commodities allowed on the vehicle.
- MDVRPTW – Multi Depot Vehicle Routing Problem with Time Windows: Stops served from more than one depot.
- VRPPD - Vehicle Routing Problem with Pickup and Delivery: A number of goods need to be moved from certain pickup locations to other delivery locations. The goal is to find optimal routes for a fleet of vehicles to visit the pickup and drop-off locations.
- Vehicle Routing Problem with LIFO: In this context LIFO stands for Last In First Out. Similar to the VRPPD, except an additional restriction is placed on the loading of the vehicles: at any delivery location, the item being delivered must be the item most recently picked up. This scheme reduces the loading and unloading times at delivery locations because there is never any need to temporarily unload items to get to the items needing to be dropped off.

This study will focus on handling several variations of the vehicle routing problem through the implementation of constraint functions in the adaptive object model framework. The purpose is to provide an adaptive solution that is *good enough* for most variations. The current computer processor ability allows the research to implement new methods that was previously deemed as too slow.

1.3 Input Data

Solving the vehicle routing problem is a complex task which results in time consuming algorithms. Knowledge of the problem environment can assist in developing more effective algorithms. The problem environment consists of the constraints imposed on the problem, the input data and the objective function to minimize with. In the field of information systems it is customary to distinguish between data, information, and knowledge.

- Data. The term data refers to numeric (or alphanumeric) strings that by themselves do not have a meaning. They can be facts of figures to be processed.
- Information. Information is data organized so that it is meaningful to the person receiving it.
- Knowledge. Knowledge has several definitions. For example, according to the Webster's New Dictionary of the American Language, Knowledge is: *a clear and certain perception of something, understanding, learning, all that has been perceived or grasped by the mind, practical experience, skill, acquaintance or familiarity, organized information applicable to problem solving.*

Data, information, and knowledge can be classified by their degree of abstraction and by their quantity. Knowledge is the most abstract and exists in smallest quantity.

Another definition of knowledge is that given by John F Sowa (Sowa, 2000): "Knowledge encompasses the implicit and explicit restrictions, placed upon objects (entities), operations, and relationships along with general and specific heuristics and inference procedures involved in the situation being modelled."

The Solomon datasets has been used as benchmark for algorithms since it has been published in 1987. Solomon generated six sets of problems. Their design highlights several factors that affect the behaviour of routing and scheduling algorithms. They are:

- Cost relation between customers represented by the geographical location;
- the number of customers serviced by a vehicle;
- percent of time-constrained customers;
- tightness and positioning of the time windows.

Solomon's geographical data are

- randomly generated
- clustered
- mix of random and clustered structures.

Some problem sets have a short scheduling horizon and allow only a few customers per route (approximately 5 to 10). In contrast, other sets have long scheduling horizons permitting many customers (more than 30) to be serviced by the same vehicle.

Current solutions for the VRP use the Solomon datasets for benchmark only. It then concludes that the algorithm was effective on certain problem types. These solutions do not attempt to utilize knowledge of the problem environment to improve results, either for speed of convergence or quality of the answer.

The other approach is to develop an algorithm to solve a specific problem type only. This method can be seen as utilizing knowledge of the problem environment in a static way. These algorithms are generally quick and effective. We are looking for the same efficiency, without the restriction of knowledge on the type of problem build into the algorithm. Methods have been proposed to identify the environment and then select the appropriate algorithm to solve the specific problem. Fuzzy clustering is a preferred way of identifying the problem space. This research will use a similar approach and will utilize the knowledge extensively.

The research will also monitor the behaviour of the operations on the data environment to implement more efficient improvement move combinations. This must be done dynamically, as the aim is still to provide one algorithm that can adapt to the problem environment.

1.4 Algorithms

The optimization of VRP type problems requires us to obtain the least of some measure, namely cost. The problem is so complex that exact algorithms do not have the power to provide high quality solutions to these types of problems. Heuristic methods will be used to assist in solving the problem.

1.4.1 Artificial Intelligence

This past few years have witnessed an increased interest in applied AI. The topic is enjoying tremendous publicity under multiple titles. Many major periodicals have published cover stories on AI or have dedicated special issues to it. Dozens of books on AI have appeared on the market. Many AI newsletters are being published regularly, and conferences and conventions on this topic are being held worldwide. To a certain extent, AI has become a sensation.

The commercial applications of AI are projected to reach several billion dollars annually. Major management consulting firms are deeply involved in applied AI. Many contribute in AI research projects.

These developments may have a significant impact on many organizations, both private and public, and on the manner in which organizations are being managed.

Artificial intelligence is a term that encompassed many definitions. Most experts agree that AI is concerned with two basic ideas. First, it involves studying the thought processes of humans (to understand what intelligence is); second, it deals with representing those processes via machines (computers, robots, etc.)

One well-publicized definition of AI is as follows: Artificial intelligence is behaviour by a machine that, if performed by a human being, would be called intelligent. A thought-provoking definition is provided by Elaine Rich (1983): "Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better." Mark Fox of Carnegie-Mellon University often says that AI is basically a theory of how the human mind works (Turban and Aronson, 2001). Winston and Prendergast (1984) list three objectives of artificial intelligence:

- Make machines smarter (primary goal)
- Understand what intelligence is (the Noble laureate purpose)
- Make machines more useful (the entrepreneurial purpose)

Let us explore the meaning of the term intelligent behaviour. Several abilities are considered signs of intelligence:

- Learn or understand from experience
- Make sense out of ambiguous or contradictory messages
- Respond quickly and successfully to a new situation (different responses, flexibility)
- Use reason in solving problems and directing conduct effectively
- Deal with perplexing situations
- Understand and infer in ordinary, rational ways
- Apply knowledge to manipulate the environment
- Acquire and apply knowledge
- Think and reason
- Recognize the relative importance of different elements in a situation

Although AI's ultimate goal is to build machines that will mimic human intelligence, the capabilities of current commercial AI products are far from exhibiting any significant success when compared with the abilities just listed. Nevertheless, AI programs are getting better all the time, any they are currently useful in conducting several tasks that require some human intelligence.

An interesting test designed to determine if a computer exhibits intelligent behaviour was designed by Alan Turing (1950) and is called the Turing Test. According to this test, a computer could be considered to be smart only when a human interviewer, conversing with both an unseen human being and an unseen computer, could not determine which is which.

The potential value of artificial intelligence can be better understood by contrasting it with natural, or human, intelligence. According to Kaplan (1984), AI has several important commercial advantages:

- AI is more permanent. Natural intelligence is perishable from a commercial standpoint in that workers can change their place of employment or forget information. AI, however, is permanent as long as the computer systems and programs remain unchanged.

- AI offers ease of duplication and dissemination. Transferring a body of knowledge from one person to another usually requires a lengthy process of apprenticeship; even so, expertise can never be duplicated completely. However, when knowledge is embodied in a computer system, it can be copied from that computer and easily moved to another computer, sometimes across the globe.
- AI can be less expensive than natural intelligence. There are many circumstances in which buying computer services costs less than having corresponding human power carry out the same tasks (over the long run)
- AI being a computer technology is consistent and thorough. Natural intelligence is erratic because people are erratic; they do not perform consistently.
- AI can be documented. Decisions made by a computer can be easily documented by tracing the activities of the system. Natural intelligence is difficult to reproduce; for example, a person may reach a conclusion but at some later date may be unable to re-create the reasoning process that led to that conclusion or to even recall the assumptions that were a part of the decision.

Natural intelligence does have several advantages over AI:

- Natural intelligence is creative, whereas AI is rather uninspired. The ability to acquire knowledge is inherent in human beings, but with AI, tailored knowledge must be built into a carefully constructed system.
- Natural intelligence enables people to benefit from and use sensory experience directly, whereas most AI systems must work with symbolic input.
- Perhaps most important, human reasoning is able to make use at all times of a wide context of experience and bring that to bear on individual problems; in contrast, AI systems typically gain their power by having a very narrow focus.

The advantages of natural intelligence over AI result in the many limitations of expert systems.

The definitions of AI presented to this point concentrated on the notion of intelligence. The following definitions and characteristics of AI focus on decision making and problem solving.

1.4.1.1 Symbolic Processing

When human experts solve problems, particularly the type that are considered appropriate for AI, they do not do it by solving sets of equations or performing other laborious mathematical computations. Instead, they choose symbols to represent the problem concepts and apply various strategies and rules to manipulate these concepts. According to Waterman, the AI approach represents knowledge as sets of symbols that stand for problem concepts. In AI jargon a symbol is a string of characters that stands for some real-world concept.

1.4.1.2 Heuristics

Heuristics (rules of thumb) are included as a key element of AI in the following definition: "Artificial intelligence is the branch of computer science that deals with ways of representing knowledge using symbols rather than numbers and with rules-of-thumb, or heuristics, methods for processing information" (Encyclopaedia Britannica)

People frequently use heuristics, consciously or otherwise, to make decisions. By using heuristics one does not have to rethink completely what to do every time a similar problem is encountered. The topic of heuristics will be revisited.

1.4.1.3 Inferencing

Artificial intelligence involves an attempt by machines to exhibit reasoning capabilities. The reasoning consists of inferencing from facts and rules using heuristics of other search approaches. Artificial intelligence is unique in that it makes inferences by employing the pattern-matching (or recognition) approach.

1.4.1.4 Pattern Matching

The following definition of AI focuses on pattern-matching techniques: Artificial intelligence works with pattern-matching methods which attempt to describe objects, events, or processes in terms of their qualitative features and logical and computational relationships.

1.4.2 Heuristic

A heuristic is a replicable method or approach for directing one's attention in learning, discovery, or problem-solving. It is originally derived from the Greek "heurisko" (εὑρίσκω), which means "I find". (A form of the same verb is found in Archimedes' famous exclamation

"eureka!" – "I have found [it]!") The term was introduced in the 4th century AD by Pappus of Alexandria.

Heuristics is the development of methods and rules for the construction of theories and theorems on a non-deductive basis (as opposed to algorithms which provide deductive foundations for such constructions). The heuristic method may be understood as a special case of the trial and error method, i.e. random attempts until a solution is found. There is no secure way. When a solution is found it may, however, be tested with scientific rigor and its truth or falsity may be established.

The heuristic method is different from the deductive method in its application of assumptions, analogies, working hypothesis, and different kinds of models. Heuristics is different from "trial and error" by not using arbitrary assumptions but apply a qualified basis from concepts, models and hypotheses.

Meta-heuristics provided a way of considerably improving the performance of simple heuristic procedures, such as those based on hill climbing. The search strategies proposed by meta-heuristic methodologies result in iterative procedures with the ability to escape local optimal points. Meta-heuristics have been developed to solve complex optimization problems in many areas, with combinatorial optimization being one of the most fruitful. Generally, the best procedures achieve their efficiencies by relying on context information. The solution method can be viewed as the result of adapting meta-heuristic strategies to specific optimization problems.

The term meta-heuristic (also written metaheuristic) was coined by Fred Glover in 1986 (Glover, 1986) and has come to be widely applied in the literature, both in the titles of comparative studies and in the titles of volumes of collected research papers. Meta (from Greek: μετά = "after", "beyond", "with", "adjacent"), is a prefix used in English in order to indicate a concept which is an abstraction from another concept, used to complete or add to the latter. In epistemology, the prefix meta- is used to mean *about* (its own category). For example, metadata are data about data (who has produced them, when, what format the data are in and so on).

A meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality.

The heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule.

The evolution of meta-heuristics during the past ten years has taken an explosive upturn. Meta-heuristics in their modern forms are based on a variety of interpretations of what constitutes "intelligent" search. These interpretations lead to design choices that in turn can be used for classification purposes. However, a rigorous classification of different meta-heuristics is a difficult and risky enterprise, because the leading advocates of alternative methods often differ among themselves about the essential nature of the methods they espouse.

This thesis will implement a specific design meta-heuristic algorithm to improve solutions in the current solution space. It will be adapted to perform efficiently in a parallel environment.

1.5 Parallel Algorithms

In computer science, a parallel algorithm, as opposed to a traditional serial algorithm, is one which can be executed a piece at a time on many different processing devices, and then put back together again at the end to get the correct result. Parallel programming was once the sole concern of extreme programmers worried about huge supercomputing problems. With the emergence of multi-core processors for mainstream applications, however, parallel programming is well poised to become a technique every professional software developer must know and master.

Parallel algorithms are valuable because it is faster to perform large computing tasks via a parallel algorithm than it is via a serial (non-parallel) algorithm, because of the way modern processors work. It is far more difficult to construct a computer with a single fast processor than one with many slow processors with the same throughput. There are also certain theoretical limits to the potential speed of serial processors. On the other hand, every parallel algorithm has a serial part and so parallel algorithms have a saturation point (see Amdahl's law). After that point adding more processors does not yield any more throughput but only increases the overhead and cost.

The cost or complexity of serial algorithms is estimated in terms of the space (memory) and time (processor cycles) that they take. Parallel algorithms need to optimize one more resource,

the communication between different processors. Two ways parallel processors communicate is shared memory or message passing.

A multi-agent system (MAS) is a system composed of multiple interacting *intelligent* agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. An intelligent agent (IA) is an autonomous entity which observes and acts upon an environment (i.e. it is an agent) and directs its activity towards achieving goals (i.e. it is rational). Intelligent agents may also learn or use knowledge to achieve their goals. The evolutionary approach through Ant Systems relate to a multi-agent system.

The agents in a multi-agent system have several important characteristics:

- Autonomy: the agents are at least partially autonomous
- Local views: no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge
- Decentralization: there is no designated controlling agent (or the system is effectively reduced to a monolithic system)

Multi-agent systems can manifest self-organization and complex behaviours even when the individual strategies of all their agents are simple. The relation of this multi-agent system approach to the parallel computing resides in the inter-communication between the agents.

Parallel algorithm design is an interesting and challenging area of computer science which requires a combination of creative and analytical skills. It is important to add the discussion to the study, to assure that there exists at least a limited design for parallel implementation. This ease the adaption of the solution for more advanced implementations in the future.

1.6 Adaptive Object Modelling

An adaptive object model is effectively a software factory with an interpretive run time, instead of code generation. It is an object model where the domain representation is interpreted at runtime and can be altered or changed with immediate effect. The adaptive model defines mechanisms to describe entities, attributes and relationships, as well as mechanisms to interpret the domain model and execute business rules. In an ever-changing

business environment, business models and rules have migrated from compiled source code to external metadata. This paradigm empowers domain experts to take control over application implementations, and allows them to change an application's business model as the business evolves.

Data themselves become more universal and reusable when they are accompanied by descriptions of themselves that let other programs make sense of them. They can become even more independent when they are accompanied in their travels by code.

As this evolutionary process unfolds, and the architecture of a system matures, knowledge about the domain becomes embodied more and more by the relationships among the objects that model the domain, and less and less by logic hardwired into the code. Objects in such an active object-model are subject to runtime configuration and manipulation like any other data. Changes to this runtime constellation of objects constitute changes to the model, and to the operations that traverse or interpret it.

Data that describe other data, rather than aspects of the application domain itself, are called metadata. Metadata is when you know something is going to vary in a predictable way and you store the descriptions of the variation in a database so that it is easy to change. The key is to define the problem domain, and then developing both design time and run time variables for that domain to facilitate the development, deployment, execution, operation and maintenance of solutions.

We will implement the concept of adaptive object modelling in describing the problem objects in metadata, as well as interpreting the object model at run-time for assisting the heuristic algorithm in having domain knowledge. We implement the solution through the Expandable Software Infrastructure (ESI) developed by E-Logics (Pty) Ltd: The ESI's goal is to realize runtime configurability, adaptability, extendibility and intuitive configuration requirements through the use of metadata and can be briefly described as a metadata-driven component based framework.

1.7 Summary

Researching the VRP provides an excellent basis to implement new methodologies on various levels. The purpose of this research is to extend the circle of research objectives by proposing a not so traditional objective in the VRP. The problem definition includes additional intentions to be solved.

The design of a solution for the defined problem environment requires a combination of different approaches on different levels. The methods discussed in this thesis are by no mean exhausted, but define the selected methods which this study will utilize. Current solutions rely mostly on heuristics to solve the complex Vehicle Routing Problem and only a few instances utilized the power of parallel algorithms.

Building the solution on an adaptive object model emphasize the importance to structure the problem in more than one dimension. This study focus mainly in providing a solution which is capable of handling adaptive objects. The remaining parts of the thesis formulate the problem, discuss some of the history of the problem and design an approach required to solve the proposed problem.

2 VRP: AN OVERVIEW

2.1 Introduction

This chapter outlines some of the related work that was done on the VRP during the past years. It will also focus on the specific methods used to solve the VRP in its different formats and data environments.

The vehicle routing problem is one of the most challenging problems in the field of combinatorial optimization. Dantzig and Ramser first introduced the VRP in 1959 (Dantzig and Ramser, 1959). The VRP originated from the Travelling Salesman Problem (TSP). The majority of OR oriented minds had been presented with the TSP or variations of, for a very considerable time (Cummings, 2000).

Both the VRP and the TSP are concerned with determination of routes in a graph such that a certain cost associated between nodes is minimized. In fact, if there is only one vehicle with infinite capacity, the consequent VRP can be seen as a simple TSP.

2.2 Travelling Salesman Problem

Mathematical problems related to the travelling salesman problem were treated in the 1800s by the Irish mathematician Sir William Rowan Hamilton and by the British mathematician Thomas Penyngton Kirkman (History of the TSP, 2007). Hamilton created the Icosian game in which the player must find paths and circuits on the dodecahedral graph, satisfying certain conditions. e.g., adjacency conditions, etc. The rights were sold for £25 to a wholesaler dealer in games and puzzles.

The general form of the TSP appears to be have been first studied by mathematicians starting in the 1930s by Karl Menger in Vienna and Harvard. In 1932 Menger published "Das botenproblem", in Ergebnisse eines Mathematischen Kolloquiums (Cummings, 2000). Menger called it the "Messenger Problem" a problem encountered by postal messengers, as well as by many travellers. He went on to define the problem as: "the task of finding, for a finite number of points whose pair wise distances are known, the shortest path connecting

the points. The rule, that one should first go from the starting point to the point nearest, etc., does not in general result in the shortest path."

During the 1950s a number of solutions appeared from the likes of George B. Dantzig, Fulkerson, and Johnson (1954). Their approach remains the only known tool for solving nontrivial TSP instances with more than several hundred cities; over the years, it has evolved further through the work of M.Grtschel, S. Hong, M. Jnger, P. Miliotis, D. Naddef, M. Padberg, W.R. Pulleyblank, G. Reinelt, and G. Rinaldi. George B. Dantzig is generally regarded as one of the three founders of linear programming, along with von Neumann and Kantorovich. (Cummings, 2000)

In 1959 George.B. Dantzig, D.R. Fulkerson, and S.M. Johnson published "On a linear-programming, combinatorial approach to the travelling-salesman problem", Operations Research 7, 59-66 (Dantzig, Fulkerson and Johnson, 1959). This provided a step-by-step application of the Dantzig-Fulkerson-Johnson for a ten city example. This was the same year that Dantzig and Ramser published their first article about the VRP.

2.3 Background on VRP

In the past four decades, a tremendous amount of work in the field of vehicle routing and scheduling problems has been published. They are summarized in recent books and surveys, see Bramel and Simchi-Levi (1997), Braysy et al. (2004), Choi and Tcha (2007), Crainic and Laporte (1998), Desrosiers et al. (1995), Fisher (1995), Laporte (1992) and Nagy and Salhi (2007) (Yeun et al., 2008). Some research efforts were oriented towards the development and analysis of approximate heuristic techniques capable of solving real-size VRP problems. Bowerman, Calamiand and Brent Hall (1994) classified the heuristic approaches to the VRP into five classes:

1. cluster-first/route-second,
2. route-first/cluster-second,
3. savings/insertion,
4. improvement/exchange and

simpler mathematical programming representations through relaxing some constraints.

From the two clustering procedures, the cluster-first/route-second looks more effective. This algorithm first groups the nodes into clusters, assigns each cluster to a different vehicle and, finally, finds the vehicle tour by solving the corresponding travelling salesman problem (TSP). Heuristic methods 3 and 4 permit to construct an initial solution or improve the current set of tours by either inserting customers or exchanging arcs. Some approximate approaches called meta-heuristics, including simulated annealing, tabu search and genetic algorithms, have recently become very popular (Gendrau, Laporte and Potvin, 1997).

On the other hand, effective optimal approaches for VRPTW problems of smaller size have also been reported. Exact approaches can be categorized according to the underlying methodology into: (a) dynamic programming techniques (Kolen, Rinnooy and Trienekens, 1987), which are extensions of the state-space relaxation method of Christofides, Mingozzi and Toth (1981); (b) Lagrangian relaxation methods which are currently capable of optimally solving some 100-customer VRPTW problems, (Desrosiers, Sauve and Soumis, 1988) (Halse, 1992) (Jornsten, Madsen and Sorensen, 1986); (c) column generation algorithms that are based on a combination of linear programming relaxed set covering and column generation (Desrochers, Desrosiers and Solomon, 1992), and (d) K-tree approaches that extended the classical 1-tree method for the TSP to the case with vehicle capacity and time window constraints (Fisher, 1994)(Fisher, Jornsten and Madsen, 1997). The first three exact approaches rely on the solution of a shortest path problem with time windows and vehicle capacity constraints either as part of a Lagrangian relaxation or to generate new columns. (Dondo and Cerdá, 2006)

The past years, quite good results have been achieved for the Vehicle Routing Problem with Time Windows (VRPTW), in both the classes of exact methods and meta-heuristics. Surveys can be found in Toth and Vigo (2001: Chapter 7) and Yeun et al. (2008). Bräysy and Gendreau (2005) give an excellent overview over meta-heuristics for the VRPTW. All of these considered traditionally vehicle routing for which each customer is visited exactly once.

The VRP is an important combinatorial optimization problem. It also occupies a central place in distribution management. Toth and Vigo (2001) report that the use of computerized methods in distribution processes often results in savings ranging from 5% to 20% in transportation costs. Baker and Ayechew (2003) and Laporte (2007) describe several case studies where the application of VRP algorithms has led to substantial cost savings.

The VRP was introduced by Dantzig and Ramser (1959) more than five decades ago. There has been since then a steady evolution in the design of solution methodologies, both exact and approximate, for this problem. Yet, no known exact algorithm is capable of consistently solving to optimality instances involving more than 50 customers (Golden et al., 1998).

Heuristics are usually used in practice. Heuristics include constructive heuristics e.g., Clarke and Wright (1964), which gradually build a feasible solution while attempting to keep solution cost as low as possible, two-phase heuristics on which customers are first clustered into feasible routes and actual routes are then constructed e.g. Fisher and Jaikumar (1981); Gillett and Miller (1974), and improvement methods which either act on single routes by application of a Travelling Salesman Problem (TSP) heuristic, or on several routes by performing customer reallocations or exchanges e.g. Kinderwater and Savelsbergh (1997), Thompson and Psaraftis (1993).

As reported by Laporte and Semet (2002), classical heuristics usually have a low execution speed but often produce solutions having a large gap with respect to the best known (typically between 3% and 7%). In the last fifteen years, several meta-heuristics have been put forward for the solution of the VRP. These typically perform a thorough exploration of the solution space, allowing deteriorating and even infeasible intermediate solutions.

A number of methods maintain a pool of good solutions which are recombined to produce even better ones. There exist many families of meta-heuristics for the VRP: simulated annealing, deterministic annealing, tabu search, genetic algorithms, ant systems, and neural networks. While the success of any particular method is related to its implementation features, it is fair to say that tabu search (TS) is most favoured in the approaches. Extensive computational experiments independently conducted by several researchers corroborate that Tabu Search outperforms most of the competitors on a regular basis. For comparative computational results over the Christofides, Mingozzi, and Toth (CMT) fourteen benchmark instances, see Gendreau, Laporte, and Potvin (Cordeau and Laporte, 2002).

Due to its wide applicability in practical settings, the VRPTW has been an area of intense research during the last ten years. Generally speaking, the methodologies for solving this problem can be classified as:

- Exact algorithms.
- Route construction heuristics.
- Route improvement heuristics.
- Composite heuristics that include both route construction and route improvement procedures.
- Meta-heuristics - like tabu search, simulated annealing, genetic algorithms, evolutionary algorithms and hybrids
- Hyper-heuristics
- Memetic Algorithms

2.3.1 Evolutionary algorithms

Genetic algorithms are adaptive heuristic search methods that mimic evolution through natural selection. They work by combining selection, recombination and mutation operations. The selection pressure drives the population toward better solutions while recombination uses genes of selected parents to produce offspring that will form the next generation. Mutation is used to escape from local minima.

Blanton and Wainwright (1993) were the first to apply a genetic algorithm to VRPTW. They hybridized a genetic algorithm with a greedy heuristic. Under this scheme, the genetic algorithm searches for a good ordering of customers, while the construction of the feasible solution is handled by the greedy heuristic.

Thangiah et al. (1994) test the same approach to solve vehicle routing problems with time deadlines. In the algorithm proposed by Potvin and Bengio (1996), new offspring are created by connecting two route segments from two parent solutions or by replacing the route of the second parent-solution by the route of the first parent-solution. Mutation is then used to reduce the number of routes and to locally optimize the solution. Berger, Salois and Begin

(1998) present a hybrid genetic algorithm based on removing certain customers from their routes and then rescheduling them with well-known route-construction heuristics.

The mutation operators are aimed at reducing the number of routes by rescheduling some customers and at locally reordering customers. Further studies built on the word of Berger through creating new crossover and mutation operators. Berger and Barkaoui (2003) continue to use the genetic algorithm as a base and a scheme was proposed that relies on the concept of simultaneous evolution of two populations pursuing different objectives subject to partial constraint relaxation. The first population evolves individuals to minimize total travelled distance while the second focuses on minimizing temporal constraint violation to generate a feasible solution, both subject to a fixed number of tours.

Homberger and Gehring (1999) propose two evolutionary meta-heuristics based on the class of evolutionary algorithms called Evolution Strategies and three well-known route improvement procedures Or-opt (Or, 1976), λ-interchanges (Osman, 1993) and 2-opt (Potvin and Rousseau, 1995). Gehring and Homberger(2000) use a similar approach with parallel tabu search implementation.

Bräysy, Berger and Barkaoui (2000) hybridize a genetic algorithm with an evolutionary algorithm consisting of several route construction and improvement heuristics. The genetic algorithm by Tan, Lee and Ou (2001) is based on Solomon's (Solomon, 1987) insertion heuristic, λ-interchanges and the well-known PMX-crossover operator. Other studies on various meta-heuristics for VRPTW can be found in Bianchessi and Righini (2007), Berger, Barkaoui and Bräysy (2002) and Yeun et al. (2008)

2.3.2 Initial Solutions

Finding a feasible and integrated initial solution to a hard problem is that the first step in addressing the scheduling issue. Heuristics typically use a greedy approach to obtain a good initial solution in an efficient manner and then incrementally improve the solution by neighbourhood exchange or local searches.

Research illustrates that high quality initial solutions allow meta-heuristics to achieve *higher quality* solutions more quickly. Marius Solomon was one of the first researchers to consider the VRPTW. He designed and analysed a number of algorithms to find initial feasible

solutions for the VRPTW (Solomon, 1987). His sequential insertion heuristic (SIH) gave very good results in most environments, and most current heuristic methods make use of this heuristic (or a variation thereof) to effectively find a feasible starting solution. It has been used frequently: e.g. VRPTW with backhauls, routing heterogeneous vehicles, routing with multiple time windows per customer, dynamic VRPTW, on-line routing, dynamic routing for airport shuttles. (Dullaert and Bräysy, 2003)

Dullaert and Bräysy (2003) introduced a push backward modification on Solomon's SIH. This method recalculates the departure time at the depot when a stop is inserted between the depot and the first stop.

Bianchi and Mastrolilli (2004) argue that because of the close relationship between the VRP and TSP, the fast algorithms for the TSP can be applied in the same manner to solve the VRP. However, the addition of a constraint on the capacity of a vehicle avoids its direct usage. Therefore, when the vehicle's capacity is too low, an optimal TSP tour cannot be anymore optimal for this problem. If the capacity is not so extremely low when compared with the existing demand, an optimal TSP tour can be a good starting solution for applying a heuristic-based algorithm for solving the VRP.

In the case of the VRP with stochastic demand, they observed a higher performance of the meta-heuristics using search strategies typical for solving the TSP, i.e. minimizing total travel distance. This could mean that, due to the stochasticity of the demand, the minimization of constraint violations considering the capacity of the vehicle becomes less important than minimizing its total distance.

Joubert and Claasen (2006) address the shortcomings of Solomon's SIH in that it considers all un-routed nodes when calculating the insertion and selection criteria for each iteration. This method makes it computationally expensive. They introduce a compatibility matrix for identifying and eliminating the obvious infeasible nodes. This results in a more effective and robust route construction heuristic.

2.3.3 VRP and Clustering

The philosophy of cluster first route second has been implemented in various shapes and formats. Clusters of nodes are first defined, then such clusters are assigned to vehicles and sequenced on the related tours and finally the routing and scheduling for each individual tour in terms of the original nodes is separately found.

Dondo and Cerda (2006) published a paper on a cluster-based optimisation approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. They presented a three phased heuristic algorithm approach: phase 1 identified a set of cost-effective feasible clusters, phase two assigns clusters to vehicles and sequences them on each tour by using the cluster-based formulation, phase 3 orders nodes within clusters and scheduling vehicle arrival times at customer locations for each tour.

The solution implements exact elimination rules, which are used to reduce the problem size and thus enhancing the efficiency of the solution algorithm. As in most solutions, these elimination rules are subject to the specific problem environment. Finding a good set of clusters, each one comprising of several customer sites, without relying on routing information is quite a difficult task. This paper introduced a time window-based heuristic algorithm that efficiently assembles customer nodes into a rather low number of feasible clusters.

The heuristic clustering procedures leads to a compact version of the VRPTW. The number of binary variables for a problem with 200 nodes, 10 vehicles and 15 clusters drops from 21 910 to 265, i.e. almost a two order of magnitude reduction. Numerical results indicate that the cluster based optimisation method proved to be quite successful on a variety of Solomon's single depot homogeneous fleet benchmark problems.

2.3.4 Tabu Search

Tabu Search is a memory-based search strategy, originally proposed by Glover (1986), to guide the local search method to continue its search beyond a local optimum. One way of achieving this is to keep track of recent moves or solutions made in the past. Several survey papers and books have been written on Tabu Search (Cordeau and Laporte, 2002).

It was Osman (1993) who proposed the concept of λ-interchanges. In his implementation he uses $\lambda = 2$, thus allowing a mix of single and double vertex moves, and single and double vertex swaps between vehicle routes. Osman tested two strategies for selecting a neighbour solution. In the first, called best admissible (BA), the best non-tabu solution is selected. In the second, called first best admissible (FBA), the first admissible improving solution is selected if one exists; otherwise the best admissible solution is retained. Osman shows through empirical testing that with the same stopping criterion FBA produces slightly better solutions, but this variant is much slower than BA. Osman's Tabu Search implementation uses fixed tabu tenures, no long-term memory mechanism and no intensification schemes.

In Taburoute neighbour solutions are obtained by moving a vertex from its current route *r* to another route *s* containing one of its closest neighbours. Insertion into route *s* is performed concurrently with a local reoptimization. This may result in creating a new route or deleting one. To limit the neighbourhood size, only a randomly selected subset of vertices is considered for reinsertion in other routes.

The Adaptive Memory Procedure (AMP) of Rochat and Taillard (1995) was presented under the title "Probabilistic Diversification and Intensification". If applied periodically during the search process, it provides a diversification process by allowing new high quality solutions to emerge. If applied as a post-optimizer, it is best seen as an intensification procedure. The method should not be regarded as a VRP heuristic per se, but rather as a general procedure applicable to several contexts and in conjunction with several heuristic schemes. For example, it was applied by Bozkaya, Erkut and Laporte (2003) to post optimize political districts obtained by means of a Tabu Search heuristic.

Xu and Kelly (1996) introduced a network flow model as a general local search strategy to solve the VRP. They used a straightforward model by relaxing the hard side constraints and introducing a dynamic penalty system, and efficiently update and frequently solve the network flow model to find the best customers to insert into new routes without the use of the generalized assignment problem. The penalty parameters are changed such that the feasibility of the search is controlled.

Garcia, Potvin and Rousseau (1994) describe a tabu search heuristic where the neighbourhood is restricted to the exchange of arcs that are close in distance. The initial solution is created using Solomon's I1 insertion heuristic, and the algorithm oscillates between

2-opt (Potvin and Rousseau, 1995) and Or-opt (Or, 1976) exchanges. When one has not made any improvement for a certain number of iterations, the other improvement operator is used and vice versa.

In order to minimize the number of routes, the algorithm tries to move customers from routes with a few customers into other routes using Or-opt exchanges. The parallel implementation is performed by partitioning the neighbourhood among slave processors. The master processor is then used to guide the tabu search. After exploration of the neighbourhood, the best move from each processor is sent to the master.

The granularity concept proposed by Toth and Vigo (1998) does not only apply to the VRP or to Tabu Search algorithms, but to discrete optimization on the whole. Like AMP, it is a highly portable mechanism. The idea is to permanently remove from consideration long edges that have only a small likelihood of belonging to an optimal solution.

More specifically, a threshold is defined and the search is performed on the restricted edge set $E(v) = \{(v_i, v_j) \in E : c_{ij} \leq v\} \cup I$, where I is a set of *important* edges defined as that incident to the depot. The value of v is set equal to $\beta\bar{c}$, where β is called a sparsification parameter, and $\bar{c}$ is the average edge cost in a good feasible solution quickly obtained, for example, by the Clarke and Wright (1964) algorithm. In practice, selecting β in the interval [1.0; 2.0] results in the elimination of 80% to 90% of all edges. Toth and Vigo have applied this idea in conjunction with Taburoute (Gendreau, Hertz and Laporte, 1994). These approaches is viewed as forced learning and relate closely to the probability matrix used in this study.

Badeau et al. (1997) study the problem using a 2-level parallel implementation that combines the so-called master-slave scheme with an allocation of each sub problem to a different processor. In this master-slave scheme, the master process manages the adaptive memory and generates solutions from it; these solutions are then transmitted to slave processes that improve them by performing tabu search and return the best solutions found to the master. Results on benchmark problems show that parallelization of the original sequential approach does not degrade solution quality, for the same amount of computation, while providing substantial speed-ups. This parallel implementation creates independent operating agents and utilise the processing power available. It does not contribute to the meta-heuristic, or memory control of the Tabu Search.

Carlton (1995) describes a reactive tabu search that dynamically adjusts its parameter values based on the current search status. More precisely, the size of the tabu list is managed by increasing the tabu list size if identical solutions occur too often and reducing it if no feasible solution can be found. This approach is applied to several types of problems with time windows. Its robustness comes from a simple neighbourhood structure, which can be easily adapted to different problems. Namely, each customer is removed and reinserted at some other location in the current solution.

Schulze and Fahle (1999) propose a tabu search performing several search threads in parallel. Each thread is started with a different initial solution and a neighbouring solution is generated by performing a sequence of simple customer shifts (ejection chain). All routes generated by the tabu search heuristic are collected in a pool. At the termination of local optimization steps, the worst solution is replaced by a new one created by solving the set covering problem on the routes in the pool with Lagrangian relaxation based heuristic.

With this new set of solutions, the whole process is restarted until a certain stopping criterion is fulfilled. In addition, the proposed method tries to eliminate routes having at most three customers by trying to move these customers into other routes. The routing of customers supplied by the same vehicle is improved by performing Or-opt exchanges within the route and the search is diversified by penalizing frequently performed customer shifts.

To generate an appropriate number of initial solutions, three different heuristics are used, namely Solomon's (Solomon, 1987) I1 insertion heuristic, the parallel route building heuristic of Potvin and Rousseau (1993) and a modified version of the Savings heuristic of Clarke and Wright (1964). In the parallel implementation, each processor handles a set of solutions instead of just one and solves also the set covering problem separately on these solutions to avoid idle times. Each time a processor terminates its local optimization process, the routes of the optimized solutions are sent to all other processors to enable sharing of knowledge.

Gehring and Homberger (2000) study a two-phase approach, where the tabu search is combined with evolutionary algorithm ES1. In this evolutionary algorithm the search is mainly driven by mutation based on Or-opt, 2-opt* and λ–interchange moves with $\lambda = 1$. In addition a special Or-opt based operator is used to reduce the number of routes.

The individuals of a starting population are generated by means of a stochastic approach that is based on the savings algorithm of Clarke and Wright (1964). The evolutionary algorithm is used in the first phase to minimize the number of routes. In the second phase, the total distance is minimized using a tabu search algorithm utilizing the same local search operators. The approach is parallelized using the concept of cooperative autonomy, i.e., several autonomous sequential solution procedures cooperate through the exchange of solutions. The cooperating slave processes are configured in different ways using different seeds for random number generators to create diversity in the search. (Gendreau and Bräysy, 2001)

Potvin and Naud (2009) presented a variant on the classical vehicle routing problem, where a customer request for a transportation company can be serviced either by its private fleet of vehicles or assigned to an external common carrier. A tabu search heuristic with a neighbourhood structure based on ejection chains is used to solve the problem. The implementation is based on the computation of a least-cost ejection path in a graph structure. This method allows multiple displacements of customers on vehicles of different types which make it effective on large heterogeneous instances.

Moccia, Cordeau and Laporte (2010) describe an incremental neighbourhood tabu search (ITS) heuristic for the generalized vehicle routing problem with time windows. The purpose of this work is to offer a general tool that can successfully be applied to a large variety of specific problems. The algorithm builds upon a previously developed tabu search heuristic by replacing its neighbourhood structure. The new neighbourhood is exponential in size, but the proposed evaluation procedure has polynomial complexity. It uses a shortest path calculation which can be computed by the so-called reaching algorithm in an acyclic graph. The computations are speed up by taking advantage of the shortest paths computed at previous steps of the algorithm.

2.3.5 Ant Optimisation

The Ant System approach, originally proposed by Colorni, Dorigo and Maniezzo (1991) is based on the behaviour of real ants searching for food. Real ants communicate with each other using an aromatic essence called pheromone, which they leave on the paths they traverse. In the absence of pheromone trails ants more or less perform a random walk.

Bullnheimer, Hartl and Strauss (1997) use the Ant System to solve the VRP in its basic form, i.e. with capacity and distance restrictions, one central depot and identical vehicles. A 'hybrid' Ant System is used with specific information for improvement. To solve the VRP, the artificial ants construct vehicle routes by successively choosing cities to visit, until each city has been visited.

Whenever the choice of a city would lead to infeasible solution for reasons of the constraints, the depot is chosen and a new tour started. For the selection of a yet not visited city, two aspects are taken into account: how good was the choice of that city, information stored in the pheromone trial, and how promising is the choice of the city, a measure of desirability.

It was found that the number of ants to start with should be the same as the number of cities in the problem space to allow each ant to start from another city. After initializing the basic ant system algorithm, the two steps, construction of vehicle routes and trial update, are repeated for a given number of iterations.

The 2-opt heuristic for the TSP is used to ensure that each tour is a 2-optimal tour, i.e. there is no possibility to shorten the tour by exchanging 2 arcs. A savings value measures the favourability of combining two cities and calculates their relative location to each other as well as to the depot. Capacity utilization is also measured as a factor to determine the probability of the next city. Although good results were obtained, a tabu search heuristic still outperforms their approach.

Mailleux, Deneubourg and Detrain (2000) compared the behaviour of Lasius niger scouts at sucrose droplets of different volumes, and empirically identified the criterion used by each scout to assess the amount of food available as well as the rules governing its decision to lay a recruitment trail. When scouts discovered food volumes exceeding the capacity of their crop, 90% immediately returned to the nest laying a recruitment trail.

In contrast, when smaller food droplets were offered, several scouts stayed on the foraging area, presumably exploring it for additional food. If unsuccessful, they returned to the nest without laying a trail. The droplet volume determined the percentage of trail-laying ants but had no influence on the intensity of marking when this was initiated. The key criterion that regulated the recruiting behaviour of scouts was their ability to ingest their own desired volume.

This volume acted as a threshold triggering the trail-laying response of foragers. Collective regulation of foraging according to food size resulted from the interplay between the distribution of these desired volume thresholds among colony members and the food volume available. We borrow from the behaviour and note that even the ants have mechanisms to balance between diversification and intensification of the improvement part.

Gambardella, Taillard and Agazzi (1999) presented an Ant Colony Optimization which is organized with a hierarchy of artificial ant colonies designed to successively optimize a multiple objective function: the first colony minimizes the number of vehicles while the second colony minimizes the travelled distances. Cooperation between colonies is performed by exchanging information through pheromone updating. The basic ACO idea is that a large number of simple artificial agents are able to build good solutions to hard combinatorial optimization problems via low-level based communications.

Montemanni et al. (2003) developed a new algorithm for dynamic VRP base on Ant Colony System. The ACS-DVRP algorithm they propose for the DVRP is based on three main elements. First, there is an event manager, which collects new orders and keeps trace of the already served orders and of the current position of each vehicle. The event manager uses this information to construct a sequence of static VRP-like instances, which are solved heuristically by an ACS (Ant Colony System) algorithm, the second element of our architecture.

The third element, the pheromone conservation procedure, is strictly connected with the ACS algorithm. It is used to pass information about characteristics of good solutions from a static VRP to the following one. The Ant Colony System (ACS) algorithm is an element of the Ant Colony Optimization (ACO) family of algorithms. The main underlying idea was to parallelize search over several constructive computational threads. A dynamic memory structure, which incorporates information on the effectiveness of previously obtained results, guides the construction process of each thread. The behaviour of each single agent is inspired by the behaviour of real ants.

Gambardella et al. (2003) present a modular approach to ALS (advanced logistics systems) design and implementation, driven by the user needs. They show how different algorithms and modules can be implemented in an ALS and how tailor-made solutions can be integrated into traditional supply chain management software. An always increasing number of large and

medium-large distribution companies have already adopted ALS to manage their whole supply chain.

The basic information processing infrastructure is in place and many supply chain management suites already provide optimization modules for some components. The objective of their AntRoute software component is to integrate a state-of-the-art optimization algorithm within an existing supply chain management structure. AntRoute has been implemented in C++ and it has been deployed as windows DLL, but its code can be recompiled under most operating systems. The algorithm has been modelled after MACS-VRPTW (Gambardella, Taillard and Agazzi, 1999)

Reimann, Doemer and Hartl (2003) have developed a generalized Ant System. Generally, the Ant System algorithm consists of the iteration of three steps: Generation of solutions by ants according to private and pheromone information, application of a local search to the ants' solutions and update of the pheromone information. They use an Insertion algorithm derived from the I1 insertion algorithm proposed by Solomon for the VRPTW. The algorithm is adapted to allow for a probabilistic choice in each decision step. This is done by choosing seed customers probabilistically according to their distance from the depot.

Inserting further customers on the current tour is done using a roulette wheel selection overall un-routed customers with positive evaluation function κ_i. The chosen customer i is then inserted into the current route at its best feasible insertion position. To compute the evaluation function κ_i for inserting an unrouted customer i at its best insertion position on the current tour we first determine for each un-routed customer i the attractiveness of insertion at any feasible insertion position on the current tour.

Given the attractiveness they then compute the evaluation function of the best insertion position for each customer i on the current tour. After all ants have constructed their solutions, the pheromone trails are updated on the basis of the solutions found by the ants.

Bianchi and Mastrolilli (2004) carried out research which focuses on the Vehicle Routing Problem with stochastic demand, a variation of deterministic classical routing problems, where each customer demand is assumed to follow a given probability distribution, instead of having a single known value. The ACO meta-heuristic is implemented for the deterministic problem in the research.

The stigmergic information is stored in the form of a matrix and at the beginning of the algorithm these values are initialized to a parameter τ_0, except for those elements that belong to the starting solution, generated by the farthest insertion heuristic, who receive a 'reinforcement' equal to *r* iterations of global update rule. After each construction step, a local update rule is applied to the element of the matrix corresponding to the chosen customer pairs.

For the VRP with stochastic demand, the ACO was implemented as two algorithms as follows: ACS-0 where after the a-priori solution is constructed, the OrOpt-0 local search is applied, and ACS-tsp where the OrOpt-tsp local search is used instead of OrOpt-0. The OrOpt-0 and OrOpt-tsp differ in the way the neighbouring solutions are evaluated. The first version exploits the VRP with stochastic demand's objective function, while the second version exploits the TSP objective function. Results show not a significant difference from using the Ant Colony algorithms from other methods for the VRP with stochastic demand.

Despite the common principles of intensification and diversification, meta-heuristics can be profoundly different and also their applicability to a given problem can produce varied results (Rizzoli et al., 2004). Moreover, it was also remarked by Martin, Otto, and Felten that meta-heuristics tend to perform very well when *hybridized* with local search methods, combining specific problem knowledge in the improvement of the solutions. ACO is particularly apt for hybridization, since it is rarely able to build a solution that is good enough, but on the other hand it produces good candidate solutions which can be further improved using various local search techniques.

In their conclusion, after more than ten years of research, ACO has proven to be one of the most successful meta-heuristics and its application to real world problems demonstrates that it has now become a fundamental tool in applied Operational Research and Management Science.

2.3.6 Hyper-Heuristics

A hyper-heuristic is a heuristic search method that seeks to automate, often by the incorporation of machine learning techniques, the process of selecting, combining, generating or adapting several simpler heuristics (or components of such heuristics) to efficiently solve computational search problems. Hyper-heuristics can be thought of as "heuristics to choose

heuristics". One of the motivations for studying hyper-heuristics is to build systems which can handle classes of problems rather than solving just one problem.

The fundamental difference between meta-heuristics and hyper-heuristics is that most implementations of meta-heuristics search within a search space of problem solutions, whereas hyper-heuristics always search within a search space of heuristics. One of the motivations of hyper-heuristic research is to investigate the development of adaptive decision support systems that can be applied to a range of different problems and different problem instances. One possible approach is to dynamically adjust the preferences of a set of simple low-level heuristics (or neighbourhood operators) during the search.

Bai et al. (2007) research a simulated annealing hyper-heuristic technique in order to investigate how the algorithm can intelligently choose between different neighbourhood operators (heuristics) according to the different problems. They consider two of the most popular variants of vehicle routing problems (capacitated VRP and VRP with time windows) and investigate the adaptation of the heuristic-selection mechanism across these variants and at different stages of the search.

Specifically, they investigate: 1). How the hyper-heuristic adapts to these two types of vehicle routing problems by changing preferences of low-level heuristics and whether the hyper-heuristic can automatically identify heuristics that are particularly good for a given type of vehicle routing problem? 2). How the hyper-heuristic adapts its selection decision during different stages of the search when solving a particular problem instance? Their hypothesis is that the hyper-heuristic algorithm will produce good quality solutions across a range of problem instances, without having to resort to tuning parameters for each instance.

Cuesta-Cañada, Garrido and Terashima-Marín (2005) use the idea behind hyper-heuristics which is to find some combination of simple heuristics to solve a problem instead than solving it directly. In this paper they introduce the first attempt to combine hyper-heuristics with an ACO algorithm. The resulting algorithm was applied to the two-dimensional bin packing problem, and encouraging results were obtained when solving classic instances taken from the literature. The performance of our approach is always equal or better than that of any of the simple heuristics studied, and comparable to the best meta-heuristics known.

2.3.7 Memetic Algorithms

Memetic Algorithms (MA) is used as a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem search. Cultural evolution, including the evolution of knowledge, can be modelled through the same basic principles of variation and selection that underlie biological evolution. This implies a shift from genes as units of biological information to a new type of units of cultural information: memes.

A meme is a cognitive or behavioural pattern that can be transmitted from one individual to another one. Since the individual who transmitted the meme will continue to carry it, the transmission can be interpreted as a replication: a copy of the meme is made in the memory of another individual, making him or her into a carrier of the meme. (Dawkins, 1976)

Berger and Barkaoui (2002) involves parallel co-evolution of two populations to solve the VRPTW. The first population evolves individuals to minimize total travelled distance while the second focuses on minimizing temporal constraint violation to generate a feasible solution. New genetic operators have been designed to incorporate key concepts emerging from promising techniques such as insertion heuristics, large neighbourhood search and ant colony systems to further diversify and intensify the search.

The parallel version of the method is based on a master-slave message-passing paradigm. The master controls the execution of the algorithm, synchronizes atomic genetic operations and handles parent selection while the slaves concurrently execute genetic operations. Results from a computational experiment show that the serial version of the proposed technique matches or outperforms the best-known heuristic routing procedures. Alternatively, simulation results obtained for the parallel version show a significant improvement over the serial algorithm, matching or even improving solution quality. The parallel algorithm shows a speed-up of five in computing solution having near similar quality.

Prins and Bouchenoua (2002) present a memetic algorithm for solving a new vehicle routing problem that generalizes two classics, the VRP and the CARP. An extended model is introduced: the NEARP (Node, Edge and Arc Routing Problem). It is defined on a mixed graph with required nodes, edges and arcs and contains the VRP and the CARP as particular cases. A common data structure shared by all algorithms is proposed for coding NEARP

instances. The first algorithms developed are three simple heuristics that are used to initialize the initial population of the MA. The third heuristic, a tour splitting method, plays also a key-role in chromosome evaluation.

A memetic algorithm for the NEARP is developed. It manipulates chromosomes corresponding to sequences of tasks, without trip delimiters, allowing adaptations of simple crossovers like OX or LOX. The tour splitting technique designed for the third heuristic is used to split the chromosomes into trips

Mendoza et al. (2010) presents a Multi-Compartment Vehicle Routing Problem (MC-VRP) which consists of designing transportation routes to satisfy the demands of a set of consumers for several products that because of incompatibility constraints must be loaded on independent vehicle compartments. Memetic algorithms are evolutionary algorithms that use local search procedures to intensify the genetic search. The proposed MA shares some of the elements that have been proven effective on the distance-constrained VRP. Starting from an initial population $\mathcal{P}(0)$ comprised of P individuals, the algorithm runs for T generations.

At every generation *t*, crossover, mutation, and local search operators are applied with probabilities p_c, p_m and p_{ls}, respectively. The offspring produced by the operators join the current population to form an expanded population $\mathcal{E}(t)$, from which the best P individuals are selected to become part of the next generation, namely $\mathcal{P}(t+1)$. Clones, which are individuals sharing the same value of the objective function, are completely forbidden in the population to foster diversification in the objective space.

2.4 Problem Overview

There exist ample research and methods to solve the VRP. Solutions can be summarized on two levels: the methodology to follow, e.g. meta-heuristic methods and the strategy implemented e.g. local knowledge of the problem. Almost all research focus on solving a specific case of the VRP in a known environment. The problem in this thesis is unique and existing research provides a toolset to work out an answer for the problem.

It is argued that in order to tackle a complex problem domain, the first thing to do is to construct a well-structured problem formulation, i.e. a "representation". This requires identifying a problem by specifying the undesirable and problematic state currently occupied,

the resources currently available to move away from that problematic state, particularly the available courses of actions, the combinatorial constraints on using them, etc., and the criteria that need to be satisfied to say that a problem no longer exists or is solved.

Problem formulation is the creative and probably the more important step towards overcoming a problematic state than problem-solving. A good definition of what the problem is, is believed to be more than half of the way towards its eventual elimination. (Krippendorff)

The remainder of this chapter identifies and formulates the problem domain as well as the resources used in solving the problem. It first presents the basic interpretation of the Vehicle Routing Problem, followed by an overview of some of the variants. The final sections of the chapter are devoted to methods and scenarios that influence the ease of solving the complex problem. The chapter conclude with the actual problem aimed to be solved.

2.5 Formulation: Vehicle Routing Problem

Logistics can be defined as the provision of goods and services from a supply point to various demand points. The transportation of raw materials from the suppliers to the factory, from the factory to the depots, and the distribution to customers can be described as a complete logistic system. With an effective logistic system, cost can be reduced due to fewer penalties for late delivery, lowered trucking cost, shorter distances and effective use of capacity of the vehicle. One of the most significant measures of a logistic system is effective vehicle routing. Optimizing of routes is the basis of vehicle routing problems.

The VRP originated from the Travelling Salesmen Problem (TSP). According to Winston (Winston, 1994) the TSP can be define as a problem where a salesperson must visit each of ten cities once before returning to his home. The cities need to be selected to minimize the total distance the salesmen travels.

According to Barbarosoglu and Ozgur (1999) the VRP can be described as the problem of designing optimal delivery or collection of routes from one or several depots to a number of customers subject to side constraints. Thus, the basic VRP can be described as vehicles that depart from the depot, visit one or more customers and return to the depot.

The VRP has a finite number of feasible solutions. The solution space increase exponentially as the number of customers increases. Thus the VRP is known as a non-polynomial hard (NP-hard) problem.

The basic VRP is today no more than a classical problem. The advance of science has prompted the industry to ask for more real life solutions. The basic VRP is given by a set of identical vehicles, a depot, a set of customers to be visited and a directed network connecting the depot and customers. Let us assume there are K vehicles, V = {0,1,2,3,…K-1}, and N+1 customers, C = {0,1,2,3,…N}. We denote the depot as customer 0, or C_0. Each arc in the network corresponds to a connection between two nodes. A route is defined as starting from the depot, going through a number of customers and ending at the depot. A cost c_{ij} and a travel time t_{ij} are associated with each arc of the network.

The problem is to find tours for the vehicles in such a way that:

- The objective function is minimized. The objective function can be the total travel distance, the number of vehicles used, or any cost related function.

Several constraints must be applied on the **basic** VRP:

- Only one vehicle handles the deliveries for a given customer. We will not split deliveries across multiple vehicles. A customer can only be visited once a day.
- The number of vehicles is equal to the number of routes, meaning that a vehicle can only complete one route per day.
- The demand of the customers on every route is known with certainty. The demand of the customers in total on one route cannot exceed the capacity of the specific vehicle that will cover that route.
- The travelling distance between customer *i* and *j* are the same as the travel distance between *j* and *i*.
- The vehicles have the same capacity with the same fixed and variable cost, thus a homogeneous fleet is assumed.
- The vehicles must complete their route within a maximum length of time, usually the time the depot is open.

- The vehicle returns to the depot at the end of the route.

The VRP can be formulated as follows:

- A set of identical vehicles V
- A special node called the depot,
- A set of customers C to be visited
- A directed network connecting the depot and the customers

Let us assume there are

K vehicles, V = {0, 1, 2,…, K – 1}, and N + 1 customers, C = {0, 1, 2, … , N}.

For simplicity, we denote the depot as customer 0.

- Each arc in the network corresponds to a connection between two nodes.
- A route is defined as starting from the depot, going to any number of customers and ending at the depot.
- The number of routes in the traffic network is equal to the number of vehicles used, K. Therefore, exactly K directed arcs leave the depot and K arcs return to the depot.
- A cost c_{ij} and a travel time t_{ij} are associated with each arc of the network.
- Every customer in the network must be visited only once by one of the vehicles.
- Since each vehicle has a limited capacity q_k, and each customer has a varying demand m_i, q_k must be greater than or equal to the summation of all demands on the route travelled by vehicle k.
- Vehicles are also supposed to complete their individual routes within a total route time, which is essentially the time window of the depot.

There are two types of decision variables in a VRP.

- The decision variable $x_{ijk,}(i, j = 0,1,2..N; k = 0,1,2..K; i \neq j)$ is 1 if vehicle k travels from node i to node j, and 0 otherwise.

- The decision variable t_i denotes the time a vehicle starts service at node i. The triangular inequality, i.e. $c_{ij} < c_{ih} + c_{hj}$ and $t_{ij} \leq t_{ih} + t_{hj} \forall h, i, j \in N$ need not apply.

The objective is to design a set of cost-minimizing routes that service all the customers while all the constraints stated above are satisfied. The model can be mathematically stated as follows:

Notation:

K = total number of vehicles.	N = total number of customers.
c_i = customer i, where i = 1, 2, … ,N.	c_0 = the depot.
c_{ij} = cost incurred on arc from node i to j.	t_{ij} = travel time between node i and j.
m_i = demand at node i.	q_k = capacity of vehicle k.
e_i = open time at node i.	l_i = close time at node i
t_i = arrival time at node i.	f_i = service time at node i.
r_k = maximum route time allowed for vehicle k.	p_i = polar coordinate angle of c_i.
R_k = vehicle route k.	O_k = total overload for vehicle k.
T_k = total tardiness for vehicle k.	D_k = total travel distance for vehicle k.
W_k = total travel time for vehicle k.	$C(R_k)$ = cost of the route R_k based on cost function.
$C(S)$ = sum total cost of individual routes $C(R_k)$.	

Table 1: Notation

Principle decision variable: $x_{ijk} = \{0,1\}$: 0 if there is no arc between node *i* and *j* and 1 otherwise.

$$Min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=0}^{K-1} c_{ij} x_{ijk} \tag{1}$$

Subject to:

$$\sum_{k=0}^{K-1} \sum_{j=1}^{N} x_{ijk} = K \quad for\ i = 0 \tag{2}$$

$$\sum_{j=1}^{N} x_{ijk} = \sum_{j=1}^{N} x_{jik} \leq 1 \quad for\ i = 0; k \in [0, K-1] \tag{3}$$

$$\sum_{k=0}^{K-1} \sum_{j=1}^{N} x_{ijk} = 1 \quad for\ i = 1, 2..N \tag{4}$$

$$\sum_{i=0, i \neq h}^{N} x_{ihk} - \sum_{j=1, j \neq h}^{N} x_{hjk} = 0 \quad \forall h \in [1, N]; k \in [0, K-1] \tag{5}$$

$$u_i - u_j + N x_{ij} \leq N - 1 \quad for\ i \in [1, N]; j \in [1, N]; i \neq j \tag{6}$$

$$\sum_{i=0}^{N} m_i \sum_{j=0, j \neq i}^{N} x_{ijk} \leq q_k \quad \forall k \in [0, K-1] \tag{7}$$

$$\sum_{i=0}^{N} \sum_{j=0, j \neq i}^{N} x_{ijk} (t_{ij} + f_i + w_i) \leq r_k \quad \forall k \in [0, K-1] \tag{8}$$

- The objective function of the problem is given in (1).
- Constraint (2) specifies that there are exactly K routes going out of the depot.
- The third constraint (3) makes sure that each route leaves the depot and return to the depot
- Constraints (4) and (5) make sure exactly one vehicle goes to and leaves a customer.

- Constraint (6) ensures that there are no sub-tours in the solution. A sub-tour is a route that does not pass through the depot.
- (7) is the capacity constraint.
- Maximum travel time for each vehicle is assured in Eq. (8).

This paragraph describes the elements of the **basic** VRP. These basic principles will be altered to fit into the generic framework designed. The next paragraph investigates some variants on the VRP and the impact on the basic VRP.

2.6 VRP variants

The model described in this section is a standard mathematical model for a basic VRP problem. When additional constraints are needed, they must be added to the existing constraints in the model or some of the existing constraints must be relaxed.

The industry requires additional constraints on the basic VRP. Additional constraints that can be included consist of the following:

- The limitation of the length, duration or cost of each individual tour. This restricts a route for running too long, which can result in overtime costs, insufficient fuel, etc.
- The addition of a variable service time for each customer. The volume of the stock to be delivered can have an influence on the service time at a customer. The delivery time will have an influence on the total route time and must be taken into account.
- The addition of time windows during which the customers have to be visited. The problem we will discuss is the use of multiple time windows, i.e. the customer can specify more than one time period available for delivery.
- The vehicle can return to the depot and have enough time for another route before the maximum allowed time is up. This will allow double scheduling, which will result in a cost saving, as the second route utilize the same vehicle and reduce the number of vehicles required to service all the customers.
- The travel time can vary between customers depending on the time of day. This implies peak and off-peak travel times.

- The fleet is not necessarily homogeneous, i.e. vehicles can differ in capacity and cost. This might result in a good solution to use the vehicles with a large capacity to pick up customers that is far away from the depot.
- A vehicle can have a specified available time. This allows for certain vehicles to be out in the field longer to cater for long routes. The implementation will add time window constraints to a vehicle.

Implementation of these variants requires a redefinition of the mathematical model for our problem, for example if we allow double scheduling:

- Constraint (2) is now invalid and will be replace by

$$\sum_{j=1}^{N} x_{ijk} \le p_k \quad for\ i = 0; k \in [0, K-1] \tag{2}$$

where p_k is the maximum number of routes allowed for vehicle k.

The number of routes going out of the depot for a specific vehicle is constrained to a maximum of p_k, which implies that a vehicle can now have multiple routes done in a day.

- If we impose time windows at a stop

$$t_0 = 0 \tag{9}$$

$$t_i + x_{ijk}(t_{ij} + f_i + w_i) \le t_j \quad i, j \in [1, N]; i \ne j; k \in [0, K-1] \tag{10}$$

$$e_i \le t_i \le l_i \tag{11}$$

- If we redefine the service time at each stop as

f_i = Fixed Time + (Variable Time * m_i)

- If we redefine the meaning of travel time

t_{ij} = Travel Time at $(t_i + f_i + w_i)$

which calculates the travel time from i to j depending on the departure time at i.

- We just make a note that q_k is not necessarily the same for each vehicle.
- The monetary cost of a route can be calculated as follows

$$C(R_{ki}) = (F_k / \sum_{j=1}^{N} x_{ijk}) + (D_k * V_k) \quad for\ i = 0; k \in [0, K-1]$$

where the first term is the fixed cost of the vehicle divided into the number of routes and the second term is the distance of the route multiplied by the running cost of the vehicle.

2.6.1 Multiple Depots

Solving the Vehicle Routing Problem is a complex task. Allowing multiple depots into the problem formulation increase the number of solutions in the problem space exponentially. For this reason, VRP problems are generally viewed form a single depot problem.

The multiple depot concept is important to consider in the context of this study. It provides information on the construction of the problem from the raw input data. The method of handling multiple depots contributes to the approach to handle different groups of customers. The simplistic methodology used, as well as the variables considered, assists with designing the complete solution in this problem environment.

The study follows a divide and conquers approach. The generally accepted method for solving the multiple depots problem utilise the same principle. The first step towards solving the multiple depots VRP is to break it up in solvable problem spaces. Stops are allocated to depots using the following procedure:

- Create a formula that results in a discrete value per stop per depot, e.g. the distance or cost between the stop and the depot.
- Apply this formula on a stop for each depot.
- Get the minimum result and allocate the stop to the depot associated.

This study classifies this allocation method as inefficient if the discrete function does not contain other solution parameters such as neighbouring stops and vehicles. The solution is not dependent on discrete stop information, but is the result of stops interacting with others

stops and vehicles, etc. The formula should result in a discrete value for a stop, but should not only depend on the explicit relations to a depot.

The allocation of a stop to a depot can influence the overall optimization, and careful consideration should be applied in the selection of the formula. The figure below depicts a simple scenario where the allocation of stops has a major impact on the solution.

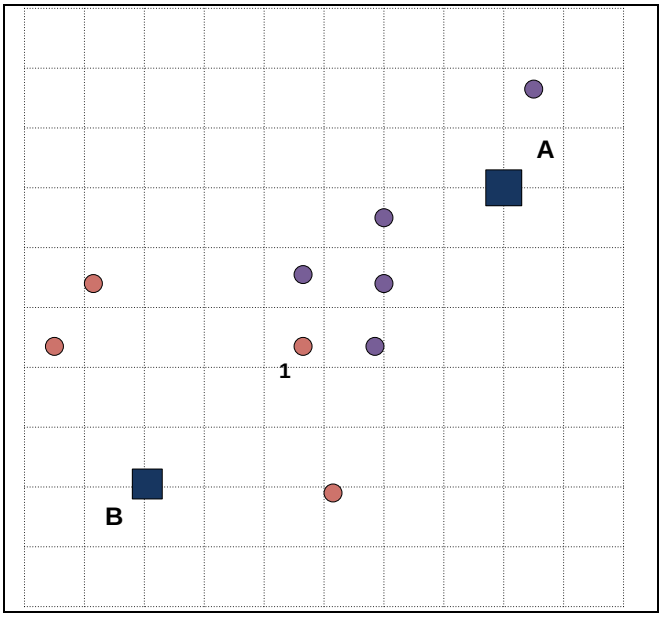

Figure 2: Stop allocation to depot

In the figure above we can see that:

- Stop number 1 is closer to depot B than depot A, and thus allocated to depot B with the simple distance based allocation formula.
- All the neighbouring stops will be serviced by depot A, because of their distance to A.
- The relation of stop 1 with other stops allocated to depot B is much weaker than its relation with the stops from A.

This solution will result in stop 1 being serviced by a vehicle that has to travel an unnecessary additional distance, just to service stop 1. The problem is that the optimization will not be able to do something about it, and the stop might even end as an orphan. The influence of neighbouring stops should be considered.

To comply with these requirements, we implement a clustering algorithm to assist in the allocation logic. This leads to a new problem; which clustering algorithm to use. There are many clustering algorithms because the notion of a 'cluster' cannot be precisely defined. What constitutes a cluster, or a good clustering, has biases because of the application. The domain knowledge is what constitutes the belief that there are subgroups among a bulk of data about objects. Such beliefs mould the structures used to represent those groups.

Clustering generates concepts, provides generalization, data summarization and is an inductive process. Given a data set, any clustering (produced by an algorithm or a human) is a hypothesis to suggest (or explain) groupings in the data.

What discriminates one hypothesis over another given the same data set? This criterion is what we refer to as the mathematical formulation of the inductive principle. It has also received the name clustering criterion. The logic of applying clustering to the multiple depot problem, instigates the advantage of using clustering on other levels of the problem.

We have formulated a sub problem in our problem space, which will assist in solving the complex problem. The next chapter will discuss the implementation of the clustering algorithm for use of classification of stops to improve knowledge of the problem environment.

2.7 Problem Statement

This study will create a solution for the VRP and some variants with the help of evolutionary algorithms implemented in parallel and build on an adaptive object model approach.

The Ant Colony Optimization technique will be the base algorithm used in this study. The aim is to solve 'any' VRP without utilising previous knowledge of the data environment, constraints and size.

The approach utilise adaptive objects to allow flexibility of implementing problem specific constraints, clustering to assist in quicker analysis of data environment and evolutionary algorithms to adapt during run-time.

2.8 Adaptive Objects

A number of forces shape the way in which software evolves. One is a desire to make programs as reusable as possible. Another is to push configuration decisions out into the data. Yet another is to push such decisions out onto the users. Still another is to defer such these decisions until runtime.

Achieving such an implementation is an evolutionary process. First, the problem is solved for a specific instance of a problem. Later, we may broaden the utility by adding options and parameters. After a while, the domain or business objects come to constitute a program of sorts, which can be dynamically constructed and manipulated by users themselves. During this evolutionary process, descriptions of the data, such as maps of the layouts of data objects, and references to methods or code, are needed to permit these heretofore anonymous capabilities to be accessible during runtime.

Our problem or goal is to design an application that has adaptability. As an object-oriented application evolves, the elements of an object-oriented framework emerge. Where raw, undifferentiated, white-box code once was dynamically pluggable black-box components begin to appear. Internal structure, which was once haphazard becomes better differentiated, and more refined. As such a framework evolves, these elements themselves, together with the protocols and interfaces they expose, come to constitute a domain specific language for the framework's target domain.

The following aspects should be considered:

- Efficiency: Highly dynamic systems can be inimical to efficiency. However, efficiency is often a false idol. For instance, the cost of referencing an object in a remote database may be several orders of magnitude more expensive than accessing a local object, and such overhead may overwhelm secondary concerns, such as the cost of assessors vs. direct variable references.
- Complexity: Complex data structures and code are hard to debug and comprehend. Alas, many programmers are better at creating complexity than simplicity.
- Resources: Dynamic strategies can be costly in terms of space, processing time, secondary storage, etc.

- Flexibility: A program should be versatile, and usable in a variety of contexts. This, in turn enhances:
- Reusability: A versatile, flexible application, or, for that matter, a code-level artefact, should be as reusable as possible. The reuse of such code avoids duplicated effort, eases the learning and comprehension burden of new programmers, and makes maintenance easier, since multiple, redundant copies of essentially the same code need not be maintained.
- Adaptability: It is essential that an artefact be flexible enough so as to confront and address changing requirements. We distinguish several "shades" of adaptability.
- Maintainability: It is important that an artefact be maintainable enough to as to confront and address changing requirements. Code that can't be worked on will lapse into stagnation.
- Tailorability: One size does not fit all. Often, an artefact will not fit the needs of a particular user "off the rack", but can be tailored to do so when certain "alterations" can be made.
- Customizability: Just an artefact can be tailored to a particular user or users; it can be customized to adapt it better for a particular task. This may seem at first to be a lot like tailorability, but we find that distinguishing between forces for change than emanate from individual users and those that arise from taking on different tasks useful.

The problem will be solved on the adaptive object model principle. The advantage of this model should be clear, without compromising the efficiency of the algorithm too much. The framework will be clearly defined for the end user to understand. We approach the implementation of this model in the following areas:

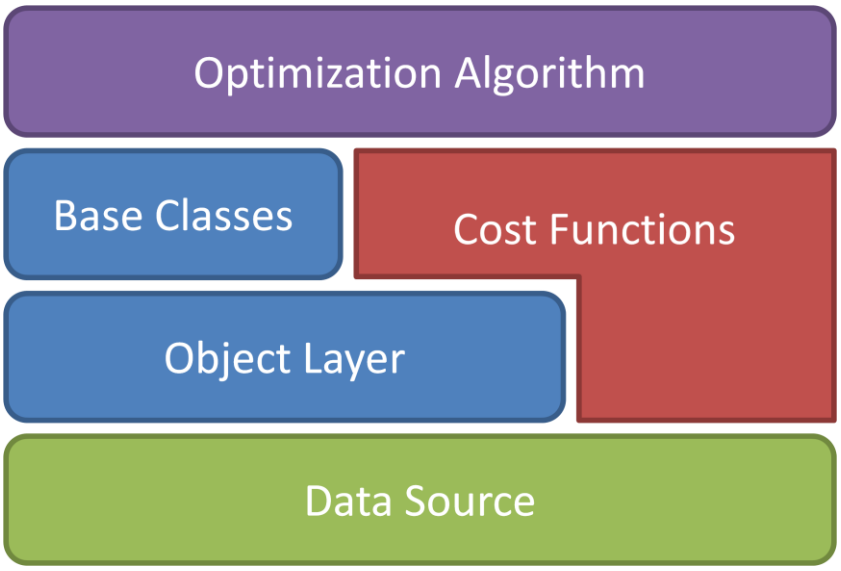

Figure 3: Object Layers

2.8.1 Data source

This layer provides access to the *raw* data and is the responsibility of the implementer. The method of access is not be part of the research and does not influence the solution. It is important that the data is available to load into the objects and can be accessed for any function. The data should also be complete regarding all entities required by the object layer and the cost layer.

The generic component used in this study, the ESI (Expandable Software Infrastructure) utilizes metadata to map the physical data to the data source, which ease the implementation on top of different databases.

2.8.2 Object layer

The object layer represents the source data in a managed and structured way, i.e. the data can be accesses as an object with properties and methods. All the objects are stored in the problem space to be available for usage through the solution. The object layer implements the business entities of the problem instance. A typical problem space in the VRPTW representative problem consist of the following objects and properties, depending on the constraints applied on the problem type:

- Stop – Volume, location, time windows.

- Depot – Time windows, location
- Vehicle – capacity

The object layer can differ from solution to solution and we therefore require a more dependable interface to the algorithm.

2.8.3 Base classes

The input data is loaded into objects and used as the problem space consists of static data with all properties available that is used to solve the problem. The optimization algorithm requires a known interface to work with. From the representative problem formulation, we define the base objects that are required as a list of stops, a depot, a list of vehicles (resources), a list of routes and a solution object. Detail description and explanation follows later in the thesis.

Vehicles do not form part of the original description according to Barbarosoglu and Ozgur (1999), but was identified as a critical enough side constraint input to be part of the base object model. This study re-evaluates the classification of a vehicle as a base class and proposes the classification as an object required to feed a constraint or cost function. The concept formulated stipulates the notion that a route cannot exist without a resource available. This study will utilise a vehicle as the resource, but in an abstraction, a resource can be defined as an entity that must exist before a route can exist.

The problem is to define an interface for the objects that is complete and sufficient enough to be used in the algorithm without limiting the implementer to a fix structure. Any module using this VRP solution that implements the defined interfaces can provide data to the resulting algorithm.

2.8.4 Cost functions

This layer represents the user defined objective cost functions as well as constraint functions. The implementation view a constraint function as checking if a solution is valid, while the cost function is the driver toward a good solution. Combined with a meta-heuristic approach, the cost function is used by the heuristic to determine the quality of the solution and the constraint function to determine the validity of the solution. In the adaptive object model

approach, these two types of functions are the main drivers that support the meta guidance. It is important to remember that these functions are not known beforehand.

2.8.5 Optimization algorithm

The objective function of the problem is to minimize cost while adhering to all constraints. An adaptive object implementation allows for a higher level of abstraction of these functions, i.e. the 'user' of the algorithm has the ability to provide the cost and constraint functions. The design of the algorithm makes use of these external functions to guide the solution to a minimum.

The optimization algorithm has the base classes and cost functions as input. The base class structures are well defined, but the cost function behaviours are unknown for the purpose of this study. The goal is to design an algorithm that can solve the problem with this limited knowledge. The algorithm must build up a knowledge base while executing.

This paper will not discuss the multi-objective cost function, but the resulting solution should be easy to adapt to incorporate such cost drivers. All the underlying layers result in known structures as input for the algorithm.

2.9 Algorithm

In mathematics, computing, linguistics and related subjects, an algorithm is a sequence of finite instructions, often used for calculation and data processing. It is formally a type of effective method in which a list of well-defined instructions for completing a task will, when given an initial state, proceed through a well-defined series of successive states, eventually terminating in an end-state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as probabilistic algorithms, incorporate randomness.

A randomized algorithm or probabilistic algorithm is an algorithm which employs a degree of randomness as part of its logic. In common practice, this means that the machine implementing the algorithm has access to a pseudorandom number generator. The algorithm typically uses the random bits as an auxiliary input to guide its behaviour, in the hope of achieving good performance in the "average case". Formally, the algorithm's performance will be a random variable determined by the random bits, with (hopefully) good expected value;

this expected value is called the expected running time. The "worst case" is typically so unlikely to occur that it can be ignored.

In computer science, a heuristic algorithm or simply a heuristic is an algorithm that ignores whether the solution to the problem can be proven to be correct, but which usually produces a good solution or solves a simpler problem that contains or intersects with the solution of the more complex problem. Heuristics are typically used when there is no known way to find an optimal solution, or when it is desirable to give up finding the optimal solution for an improvement in run time.

Two fundamental goals in computer science are finding algorithms with provably good run times and with provably good or optimal solution quality. A heuristic is an algorithm that abandons one or both of these goals; for example, it usually finds pretty good solutions, but there is no proof the solutions could not get arbitrarily bad; or it usually runs reasonably quickly, but there is no argument that this will always be the case.

Many problems in AI can be solved in theory by intelligently searching through many possible solutions:(Artificial Intelligence, 2008) Reasoning can be reduced to performing a search. For example, logical proof can be viewed as searching for a path that leads from premises to conclusions, where each step is the application of an inference rule (Albus, 2002). Planning algorithms search through trees of goals and sub goals, attempting to find a path to a target goal, a process called means-ends analysis. Robotics algorithms for moving limbs and grasping objects use local searches in configuration space. Many learning algorithms use search algorithms based on optimization.

Simple exhaustive searches are rarely sufficient for most real world problems: the search space (the number of places to search) quickly grows to astronomical numbers. The result is a search that is too slow or never completes. The solution, for many problems, is to use "heuristics" or "rules of thumb" that eliminate choices that are unlikely to lead to the goal (called "pruning the search tree"). Heuristics supply the program with a "best guess" for what path the solution lies on.

A very different kind of search came to prominence in the 1990s, based on the mathematical theory of optimization. For many problems, it is possible to begin the search with some form of a guess and then refine the guess incrementally until no more refinements can be made.

These algorithms can be visualized as blind hill climbing: we begin the search at a random point on the landscape, and then, by jumps or steps, we keep moving our guess uphill, until we reach the top. Other optimization algorithms are simulated annealing, beam search and random optimization

Evolutionary computation uses a form of optimization search. For example, they may begin with a population of organisms (the guesses) and then allow them to mutate and recombine, selecting only the fittest to survive each generation (refining the guesses). Forms of evolutionary computation include swarm intelligence algorithms (such as ant colony or particle swarm optimization) and evolutionary algorithms (such as genetic algorithms and genetic programming).

This thesis will focus on ant colony optimization techniques. The ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs and is a member of swarm intelligence methods, and it constitutes some meta-heuristic optimizations.

2.9.1 Initial Solution

The general two-phased approach in solving a problem through heuristic methods consists of an initial solution, followed by an improvement stage. High quality initial heuristics often allow local searches and meta-heuristics to achieve better solutions more quickly. The applied solution uses a similar approach and this paragraph formulates the initial solution part of the problem.

Current initial solutions focus on providing quality through minimizing the cost function. Marius Solomon was one of the first researchers to consider the VRPTW. He designed and analyzed a number of algorithms to find initial feasible solutions for the VRPTW (Solomon, 1987). His sequential insertion heuristic (SIH) gave very good results in most environments, and most current heuristic methods make use of this heuristic (or a variation thereof) to effectively find a feasible starting solution.

There exist a number of route construction heuristics. Joubert and Claasen (2006) discuss some of the most prominent improvements on the SIH. All these improvements is related to a specific aspect of the VRP type solved for that instance of the problem, for example, using

the Time Window Compatibility (TWC) depends on the problem to include time windows as a constraint. This research interprets the improvement of the initial solution algorithm as the calculated move of a constraint to this part of the algorithm. Implementing the TWC improvement on the initial solution is born out of the time window constraint.

This research implements an initial solution that utilizes the defined constraints as improvement on the initial solution. The aim is to reduce the number of unnecessary calculations while generating a high quality solution.

The purpose of the initial solution is to provide the improvement stage with a start. We consider the definition of a high quality solution as a solution that is aligned with the improvement stage. This is identified as having a solution that allows quicker convergence. We argue that using the knowledge gained during the initial stage can benefit the improvement stage.

The approach is now to define the outcome of an initial solution as the solution, as well as additional information gained. This interface to the improvement stage is flexible to include scenarios where the user just needs the improvement stage, for example, when a legacy system contains existing routes. The legacy solution can be implemented as the initial solution without any additional information.

2.9.2 Meta-Heuristics

The implementation of an algorithm that can efficiently and in reasonable time solve the aforementioned problem has not been successfully implemented before. To embark on a journey to find a sufficient algorithm requires investigation of existing problems and solutions as well as inventing new methods. Several papers have been presented that solve the VRP with additional side constraints. They mainly focus on solving the basic VRP with one or two additional side constraints. Some of the most popular problems include the VRP with time windows and the VRP with pickup and delivery.

Heuristic methods play an important role in solving problems with this complexity. Most solutions include a heuristic method, or a hybrid of heuristic methods at the heart of the solution. In the next section, we will discuss some of the more popular heuristic methods.

Meta-heuristics, or global optimization heuristics, have a common feature: they guide a subordinate heuristic in accordance with a concept derived from artificial intelligence, biology, nature or physics to improve their performance.

Meta-heuristics succeed in leaving the local optimum by temporarily accepting exchanges that decrease the objective function value. Meta-heuristics use information of the problem environment and the nature of the objective function to direct the search process to regions that promise better solutions. It is possible that the meta-heuristic will return to the local optimum without finding a better solution. This is called cycling and can be avoided by adjusting the heuristic's settings to allow more degrading moves for longer.

The concept of a heuristic being trapped at a local optimum can be demonstrated in Figure 4. If a heuristic finds a solution S, with objective function F(S), where S is close to point C, then it will only improve until it gets to local optimum C. No further improvements in the objective function will be achievable, because all moves will reduce the objective function. However, if a meta-heuristic finds a solution close to point B, degrading moves will be allowed that may direct the search to the global optimum, point A.

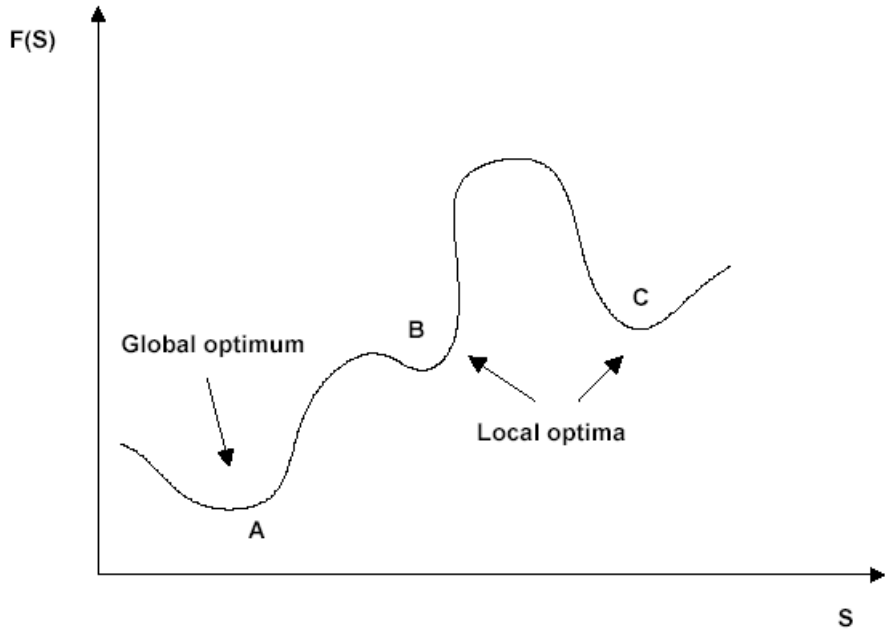

Figure 4: Global and Local Optima

Meta-heuristics will be successful on a given optimization problem if it can provide a balance between the exploitation of the accumulated search experience and the exploration of the search space to identify regions with high quality solutions in a problem specific, near optimal way. The various meta-heuristics are classified according to the following criteria:

- **Trajectory methods vs. discontinuous methods**: Trajectory methods like Simulated Annealing (SA) and Tabu Search (TS) follow one single search trajectory corresponding to a closed walk on the neighbourhood graph. Discontinuous methods allows larger jump in the neighbourhood graph.
- **Populated-based vs. single-point search**: In single-point search only one single solution is manipulated at each iteration of the algorithm. TS and SA are single-point search methods. Genetic and ant colony algorithms are population-based.
- **Memory usage vs. memoryless methods**: Meta-heuristics with memory are the TS, GA and ant systems. According to Taillard et al.(2001) these meta-heuristics with memory can be viewed as adaptive memory programming (AMP) heuristics. The term "memory" was used explicitly for TS, but other meta-heuristics use mechanisms that can be considered as memories. There are meta-heuristics that cannot be entered into the AMP methods, such as SA. However they may be included in the improvement procedure of AMP.
- **One vs. various neighbourhood structures**: SA and TS algorithms are based on one single neighbourhood structure. Other algorithms such as Iterated Local Search typically use at least two different neighbourhood structures.
- **Dynamic vs. static objective function**: Some algorithms modify the evaluation of the single search states during the run of the algorithm. In the use of a dynamic objective function penalties for the inclusion of certain solution attributes that modify the objective function are introduced. TS may be interpreted as using dynamic objective function, as some point in the search is forbidden, corresponding to infinitely high objective function values. The other algorithms use static objective functions.

Evaluation of heuristic methods consists of comparing criteria such as running time, quality of solution, ease of implementation, flexibility and robustness. For the purpose of our algorithm, flexibility is an important consideration. The algorithm should be able to handle

changes in the data patterns, side constraints and objective function, as each client has his own specific requirement. We are not working on a predetermined set of data with a specified objective function. Working in such an environment makes it possible to find a method that is effective for that specific environment by making use of the knowledge about the problem.

Because the heuristic methods are non deterministic, i.e. we cannot predict the result even if we apply the same algorithm on the same data with the same number of iterations, the algorithm should not perform poorly on any instance, as well as being able to produce a good solution each time it is applied to the same instance.

We will also try to validate the applicability of the method on our problem by discussing the design of the method is well as what we see as its advantages and disadvantages. With this approach we will filter out certain methods. Comparisons discussed in this paper are from existing papers, which mainly present the best results found for the method. Comparison is also made difficult because solutions were not all implemented on the same computer (running time), and have not all use the same number of iterations. Existing methods is also not designed for our specific problem and thus we cannot really compare methods outright to decide on a method to implement for our problem.

Using only the best results of a non-deterministic heuristic, as is often done in the literature, may create a false picture of its real performance. We considered average results based on multiple executions on each problem an important basis for the comparison of non-deterministic methods. Furthermore, it would also be important to report the worst-case performance.

Moreover, an algorithm should be able to produce good solutions every time it is applied to a given instance. This is to be highlighted since any heuristics are non-deterministic, and contain some random components such as randomly chosen parameter values. The output of separate executions of these non-deterministic methods on the same problem is in practice never the same. This makes it difficult to analyze and compare results.

The meta-heuristic method has a guidance procedure of some sort to help it traversing through the solution space. A meta-heuristic is the implementation of a heuristic method with a guidance procedure.

2.9.3 Tabu Search

Tabu Search was proposed by Glover (1986) and has quickly become one of the best and most widespread local search methods for combinatorial optimization. The method performs an exploration of the solution space by moving from a solution x_t identified at iteration t to the best solution x_{t+1} in a subset of the neighbourhood $N(x_t)$ of x_t. Since x_{t+1} does not necessarily improve upon x_t, a tabu mechanism is put in place to prevent the process from cycling over a sequence of solutions.

A naïve way to prevent cycles is to forbid the process from going back to previously encountered solutions, but doing so would typically require excessive bookkeeping. Instead, some attributes of past solutions are registered and any solution possessing these attributes may not be considered for μ iterations. This mechanism is often referred to as short term memory. Other features such as diversification and intensification are often implemented. The purpose of diversification is to ensure that the search process will not be restricted to a limited portion of the solution space. It keeps track of past solutions and penalizes frequently performed moves. This is often called long term memory. Intensification consists of performing an accentuated search around the best known solutions.

There are two main ways to define neighbourhood structures in Tabu Search algorithms for the VRP. The first, termed λ-interchanges by Osman (Osman, 1993) consist of exchanging up to λ customers between two routes. The second, called ejection chains typically acts on more than two routes at the same time.

To prevent cycling, it is common to prohibit reverse moves for μ iterations. Thus a customer moved from route r to route s at iteration t may be prohibited from being reinserted in route r until iteration $t + \mu$, or it may not leave route s until iteration $t + \mu$. Taillard (1999) suggests randomly selecting μ in an interval $[\mu; \mu]$, according to a discrete uniform distribution, but some algorithms use a fixed value of μ. It is also common to use an aspiration criterion to revoke the tabu status of a move if this causes no risk of cycling, for example if this yields a better overall incumbent feasible solution, or a better incumbent among the set of solutions possessing a certain attribute.

To diversify the search, most Tabu Search implementations penalize frequently performed moves. This is done by adding to the routing cost $c(x_{t+1})$ of x_{t+1} a penalty term equal to the product of three factors:

1. a factor measuring the past frequency of the move;
2. a factor measuring instance size (such as $\sqrt{n}$);
3. a user-controlled scaling factor.

In practice, implementing such a mechanism is both effective and computationally inexpensive.

Intensification consists of accentuating the search in promising regions. Periodic route improvements by means of TSP algorithms may be classified as intensification techniques. Another way of performing intensification is to conduct a search using a wider neighbourhood structure around some of the best known solutions.

An adaptive memory is a population of good solutions encountered during the search procedure. Similar to what is done in genetic algorithms; the idea is to combine solution elements from the population to construct new solutions. If their value is less than those of some solutions in the population, they are retained and the worst solutions are discarded from the population.

In the VRP context, a new solution is initialized from a number of non-overlapping routes extracted from good solutions. Typically these routes do not cover all vertices for otherwise there would be overlaps. The search is then initiated from the selected routes and the un-routed vertices.

2.9.4 Ant Algorithms.

Ants appeared on earth some 100 million years ago, and have a current total population estimated at 10^{16} individuals. Most of these ants are social insects, living in colonies of 30 to millions of individuals. The complex behaviours that emerge from colonies of ants have intrigued humans, and there have been many studies of ant colonies aimed at a better understanding of these collective behaviours. Collective ant behaviours that have been

studied include the foraging behaviour, division of labour cemetery organization and brood care, and construction of nests. (Engelbrecht, 2007)

The South African, Eugene Marais (1871-1936) was one of the first to study termite colonies. In 1910, Marais abandoned his law practice and retreated to the remote Waterberge – the mountain area north-west of Pretoria. Here he studied two creatures - termites and baboons which, on the face of it, had nothing in common. Both fascinated him, as did all wild creatures. (Marais, n.d.)

He published his conclusions about termites as a series of speculative articles, written entirely in Afrikaans and appearing only in local newspapers, as The Soul of the White Ant. While observing the natural behaviour of these creatures, he noticed that firstly, the whole termitarium had to be considered as a single organism whose organs work like those of a human being. The queen was the brain and the womb; the workers were mouthparts and tissue builders; the soldiers' white blood cells and the humus gardens were the stomach. And secondly that the actions within the termitarium were completely instinctive.

Marais began writing Soul of the Ape in 1916, but never finished it. It was published posthumously years later. His theory was that, unlike termites, baboons – and by extension all primates – had the ability to memorize the relationship between cause and effect. They could therefore vary their behaviour voluntarily. While termites were instinctive, the mind of baboons was based on "causal memory".

The reason for this difference, according to Marais, was natural selection. According to him, natural selection was not, like Darwin had insisted, the survival of the fittest, but rather the line of least resistance. Those species best able to adapt to their specific environment survived, while those not able to, would become extinct. Natural selection, therefore, had the tendency to both localize and specialize species.

These conclusions to which he came were new and radical and might well have had an influence in Europe. But Marais was half a hemisphere away, half a century too soon and writing in a language no one could understand. The Soul of the White Ant was brought under the attention of the world only by being seemingly plagiarized by a Belgian Nobel prize winner, Maurice Maeterlinck. The Soul of the Ape was incomplete and originally only published in South Africa.

While most research effort concentrated on developing algorithmic models of foraging behaviour, models have been developed for other behaviours, including division of labour, cooperative support, self-assembly, cemetery organization and even traffic flow. These complex behaviours emerge from the collective behaviour of much unsophisticated individuals.

In the context of collective behaviour, social insects are basically stimulus-response agents. Based on information perceived from the local environment, an individual performs a simple, basic action. These simple actions appear to have a large random component. Despite this simplicity in individual behaviour, social insects form a highly structured social organism.

One of the most surprising behavioural patterns exhibited by ants is the ability of certain ant species to find the shortest path (Dorigo and Stutzle, 2004). Biologists have shown experimentally that this is possible by exploiting communication based only on pheromones. It is these behavioural patterns that inspire the development of optimization algorithms based on the ant behaviour. The first attempts in this direction appeared in the early '90s and can be considered as rather "toy" demonstrations. Since then these and similar ideas have attracted more research which let to Ant Colony Optimization (ACO) algorithms.

How do ants find the shortest path between their nest and food source, without any visible, central, active coordination mechanisms? Studies of the foraging behaviour of several species of real ants revealed an initial random or chaotic activity pattern in the search for food. As soon as a food source is located, activity patterns become more organized with more and more ants following the same path to the food source. "Auto-magically", soon all ants follow the same shortest path.

This emergent behaviour is a result of a recruitment mechanism whereby ants that have located a food source influence other ants towards the food source. The recruitment mechanism differs for different species, and can either be in the form of direct contact, or indirect "communication". Most ant species use the latter form of recruitment, where communication is via pheromone trails. When an ant locates a food source, is carries a food item to the nest and lays pheromone along the trail.

Forager ants decide which path to follow based on the pheromone concentrations on the different paths. Paths with a lager pheromone concentration have a higher probability of

being selected. As more ants follow a specific trail, the desirability of that path is reinforced by more pheromone being deposited by the foragers, which attracts more ants to follow that path. The collective behaviour that results is a form of autocatalytic behaviour, where positive feedback about a food path causes that path to be followed by more and more ants. The indirect communication where ants modify their environment (by laying of pheromones) to influence the behaviour of other ants is referred to as stigmergy.

Deneubourg et al (1990) studied the foraging behaviour of the Argentine ant species *Iridomyrmex humilis* in order to develop a formal model to describe its behaviour. In the double bridge experiment, a nest of a colony of Argentine ants is connected to a food source by two bridges. The ants can reach the food source and get back to the nest using any of the two bridges. The goal of the experiment is to observe the resulting behaviour of the colony. What is observed is that if the two bridges have the same length, the ants tend to converge towards the use of one of the two bridges.

If the experiment is repeated a number of times, it is observed that each of the two bridges is used in about 50% of the cases. These results can be explained by the fact that, while moving, ants deposit pheromone on the ground; and whenever they must choose which path to follow, their choice is biased by pheromone: the higher the pheromone concentration found on a particular path, the higher is the probability to follow that path.

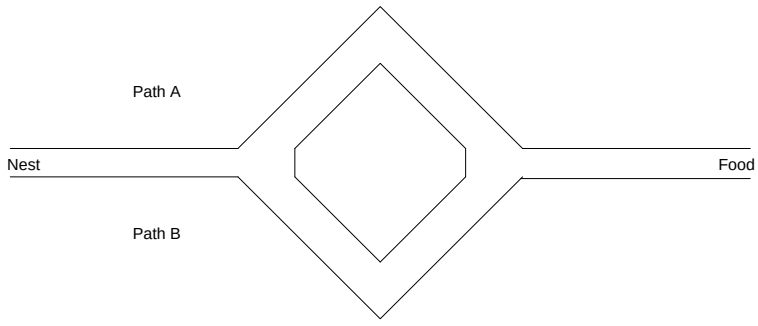

Figure 5: Binary Bridge Experiment

From this experiment, (illustrated in Figure 5), a simple formal model was developed to characterize the path section process. For this purpose, it is assumed that ants deposit the same amount of pheromone and that pheromone does not evaporate. Let $n_A(t)$ and $n_B(t)$

denote the number of ants on paths A and B respectively at time step *t*. Pasteels, Deneubourg and Goss (1987) found empirically that the probability of the next ant to choose path *A* at time step *t +1* is given as,

$$P_A(t+1) = \frac{(c + n_A(t))^\alpha}{(c + n_A(t))^\alpha + (c + n_B)^\alpha} = 1 - P_B(t+1)$$

Equation 1: Binary Bridge Probability Equation

Where *c* quantifies the degree of attraction of an unexplored branch, and α is the bias to using pheromone deposits in the decision process. The larger the value of α, the higher the probability that the next ant will follow the path with a higher pheromone concentration – even if that branch has only slightly more pheromone deposits. The larger the value of *c*, the more pheromone deposits is required to make the choice of path non-random. It was found empirically that $\alpha \approx 2$ and $c \approx 20$ provides a best fit to the experimentally observed behavior.

Using the probability defined in Equation 1, the decision rule of an ant that arrives at the binary bridge is expressed as follows: $if\ U(0,1) \leq P_A(t+1)\ then$ follow path A otherwise follow path B.

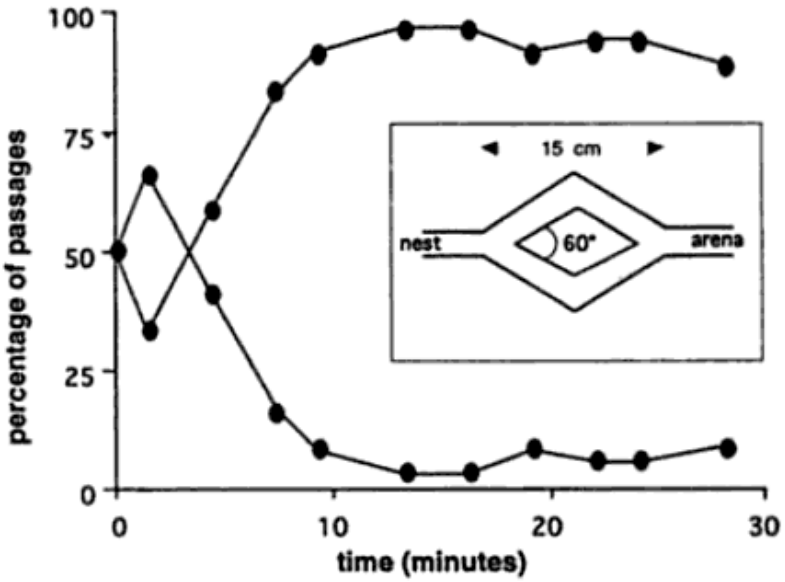

Figure 6: Percentage of all passages per unit time as a function of time.

The winning branch in time. Figure was not favoured by the initial fluctuations, which indicates that these fluctuations were not strong enough to promote exploitation of the other branch in the beginning (Bonabeau, Dorigo and Theraulaz, 1999) .

Goss et al (1989) extended the binary bridge experiment, where one of the branches of the bridge was longer than the other, as illustrated in Figure 7. Dots in this figure indicate ants. Initially, paths are chosen randomly with approximately the same number of ants following both paths (as illustrated in Figure 7 on the left). Over time, more and more ants follow the shorter path as illustrated in Figure 7 on the right. Selection is biased towards the shortest path, since ants that follow the shortest path returns to the earlier than ants on the longer path. The pheromone on the shorter path is therefore reinforced sooner than that on the longer path.

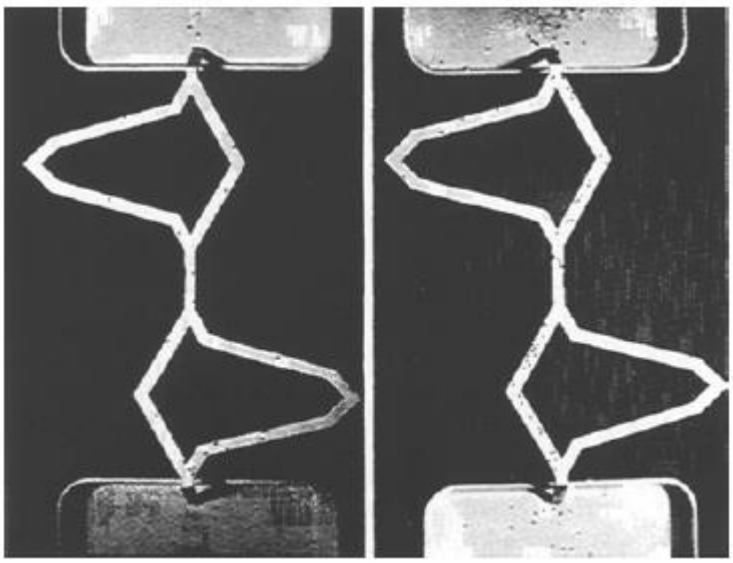

Figure 7: Shortest Path Selection by Forager Ants

Goss et al found that the probability of selecting the shorter path increases with the length ratio between the two paths.

Although an ant colony exhibits complex adaptive behaviour, a single ant follows very simple behaviours. An ant can be seen as a stimulus-response agent (Nilsson, 1998): the ant observes pheromone concentrations and produces an action based on the pheromone-stimulus. An ant can therefore abstractly be considered as a simple computational agent. An artificial ant algorithmically models this simple behaviour of real ants. The logic implemented is a simple production system with a set of production rules as illustrated in Algorithm 1. This algorithm is executed at each point where the ant needs to make a decision.

```
Let r ~ U(0,1);
For each potential path A do
      Calculate P_A using e.g. Equation 1;
      If r ≤ P_A then
            Follow path A;
            Break;
      End
End
```

Algorithm 1: Artificial Ant Decision Process

While Algorithm 1 implements a simple random selection mechanism, any other probabilistic selection mechanism can be used, for example, roulette wheel selection. The implementation of an Ant solution on the VRP necessitate a more complex method as the cost function that drives the probability depends on more combinations.

These experimental observations form a basic set of tasks any model needs to explore. The following results were obtained using a few different species of ants in the bridge experiment.

1. When ants are exposed to two paths of unequal length the ants will choose the shortest path.
2. If a shorter path is offered after the ants have chosen, they are unable to switch to the new path.
3. The ants will break symmetry and chose one path, even when both paths are equal.
4. If ants are offered two unequal food sources they will usually choose the richest source.
5. If a richer food source is offered after the ants have chosen, some species can switch to this new source, and others are unable to.

In line with the ideas proposed above, the most important modelling constraint in what follows is the principle of locality. In the models we will use, the behaviour of the individual organisms will be determined solely by local influences. This means that the individual organisms will not have any memory, non-local navigational skills, or any type of behaviour that involves storage of internal information. Any information flow must then be a product of the collective behaviour.

2.9.5 Memetic Algorithms

Memetic Algorithms (MA) are used as a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem search. Cultural evolution, including the evolution of knowledge, can be modelled through the same basic principles of variation and selection that underlie biological evolution. This implies a shift from genes as units of biological information to a new type of units of cultural information: memes.

A meme is a cognitive or behavioural pattern that can be transmitted from one individual to another one. Since the individual who transmitted the meme will continue to carry it, the transmission can be interpreted as a replication: a copy of the meme is made in the memory of another individual, making him or her into a carrier of the meme. (Dawkins, 1976)

Memetic Algorithms denote a family of meta-heuristics that have as central theme the hybridization of different algorithmic approaches for a given problem. Special emphasis was given to the use of a population-based approach in which a set of cooperating and competing agents were engaged in periods of individual improvement of the solutions while they sporadically interact. Another main theme was to introduce problem and instance-dependent knowledge as a way of speeding-up the search process. MAs exploit problem-knowledge by incorporating pre-existing heuristics, pre-processing data reduction rules, approximation and fixed-parameter tractable algorithms, local search techniques, specialized recombination operators, truncated exact methods, etc. Also, an important factor is the use of adequate representations of the problem being tackled. (Moscato and Cotta, 2005)

2.10 Parallel

Searching the solution space for the best solution can be improved through a parallel implementation. The algorithm goal is to increase the diversification of the probable solutions as well as increase efficiency. Local searches tend to get stuck in a local minimum. With a parallel method, this negative aspect can be utilized. We would like the solution to reach a local minimum as soon as possible, in as many as possible places.

The evaluation of different solutions provides some new challenges. The goal is to provide solutions that are diverse in the solution space. This requires a method to compare the diversity measure of solutions. If successfully applied, the method accomplishes a dual goal: assisting with guiding the algorithm into other areas to explore, and providing the end result with alternative solutions for the problem.

The diversification comparison assist the parallel algorithm not to pursue venues simultaneously that will ultimately end up in the same solution. It also provides the meta control mechanism powerful information regarding the tightness of the problem. If there exist a number of solutions that qualify as different from each other, it indicates that the solution space contains multiple possibilities of valid solutions. If not, the problem space is

tight and more emphasis should be put on the local search which finds the most optimal solution for that particular stream.

Providing the user with a diverse set of solutions that can be viewed as close to each other in the total cost, allows the user to execute a selection based on non-tangible variables that was not considered in the algorithm. These variables can range from a biting dog to a possible new customer to join a route. It can also assist in strategic planning by highlighting master routes through execution of different scenarios.

This thesis aim to design a parallel model for algorithm to solve various instances of the VRP on various data environment scenarios. This parallel design is an extension of the proposed adaptive algorithm. The focus is mainly on performance and efficiency, but discussion will include possible future implementation. The parallel solution assists in generating a number of diverse feasible solutions, which provide a choice for the user.

The programming style used is a synchronous master/workers paradigm on multiple levels. The master implements a central memory through which passes selective communication to share between the diverse solutions, and that captures the global knowledge acquired during the search. The worker implements the search process. The parallel algorithm continues from the initial setup which creates more than one initial solution through various methods.

The pheromone matrix and the best found solutions will be managed by the master. At specified iterations, the master consolidates the pheromone matrix from all the workers and then broadcast the pheromone matrix to all the workers. Each worker handles an ant process. He receives the pheromone matrix, constructs a complete solution, applies a tabu search for this solution and sends the solution found to the master. When the master receives all the solutions, he updates the pheromone matrix and the best solutions found, and then the process is iterated.

2.11 Summary

This thesis identifies the problem of solving the VRP with variants through applying an adaptive solution which has no predefined knowledge of the problem environment. The environment is defined as the geographical distribution of the stops, the constraints applied on the solution and the objective cost functions to minimize.

The current focus on solving VRP is to identify the type of problem beforehand and implement as solution for the specific type. Better computer processing power allows the introduction of alternative methods into the solution. This research utilizes concepts such as clustering, adaptive object modelling, meta-heuristics and adaptive object modelling to improve the quality of the solution. Parallel implementation is considered throughout to improve speed.

Success is measured if an answer is *good enough* in a *reasonable time* for a problem where the user define the *constraint* and *cost* model.