Chapter 1

Introduction

South Africa's level of urbanization closely follows international trends in developed countries, with the highest level of economic activity focused in a few metropolitan areas; attracting both people and investment. The good functioning of these metropolitan areas is of strategic importance to the country, as these areas are the main focus for economic and social development. The level of transport services provided impacts directly on the efficiency and the quality of the development in the metropolitan areas. South African metropolitan areas are experiencing rapid growth, and are having difficulties in controlling the physical urban expansion. Both public and freight transport costs are negatively impacted by these phenomena. As demand for transport increases faster than the supply of these services, commuting and freight transportation costs increase at a higher than inflation rate. The community at large experiences the demands for more extensive infrastructure and services.

Customers, both businesses and private consumers, demands products and services at the point of utilization. The geographically dispersed point of supply and point of utilization are bridged through transport. The majority of urban freight is carried by means of road transport, and the definition of the *Organization for Economic Co-operation and Development* (OECD) for urban freight transport applies:

> *"The delivery of consumer goods (not only retail, but also by other sectors such as manufacturing) in city and suburban areas, including the reverse flow of used goods in terms of clean waste."* — OECD (2003)

Goods transport has a major impact on the economic power, quality of life, accessibility and attractiveness of local communities, especially in city and metropolitan areas, but receives much less attention in comparison to passenger movement. According to the first *State of*

Logistics Survey for South Africa prepared by CSIR Transportek (2004), 83% of the total tonnage transport bill of ZAR 134 billion is transported via road, while 22% of the total tonnage is transported within metropolitan areas. Freight transport within metropolitan and urban areas have different characteristics from long haulage, and the main attributes include (Taniguchi et al., 2004):

- Frequent deliveries of smaller quantities
- Low utilisation of the capacity of trucks
- Time windows

Efficiently transporting goods within urban areas facilitates the establishment of sustainable cities. OECD (2003) acknowledges the contribution that freight vehicles make to traffic congestion, energy consumption and negative environmental impacts. Yamada and Taniguchi (2005) conclude that the majority of benefits for freight carriers can be achieved by implementing advanced vehicle routing and scheduling systems, hence addressing congestion, energy consumption, and indirectly environmental impacts. The problem concerned with allocating customer deliveries (or collections) to vehicles, and determining the visiting order of those customers on each vehicle route, is classified as the Vehicle Routing Problem (VRP), and has as its main objective to minimize some measurable function, such as distance traveled, time traveled, or total fleet cost.

1.1 *Modeling* as research motivation

South Africa provides a fascinating interface between the developed and the developing world. In a critical review, Leinbach and Stansfield (2002) have emphasized that Industrial Engineers should re-adopt a systematic view. They argue that the perception of Industrial Engineers has been negatively impacted by their ability to model the obvious, and in the over-simplification of their models, to the extent that reality is not represented comprehensively. Industrial engineers should therefor appreciate the complex and intertwined relationships between social, political, and economic factors influencing urban freight transport systems.

A systematic approach in addressing a problem is illustrated in the lower cycle of Figure 1.1 where a problem is modeled, the model is solved, and the solution is interpreted so as to change the original problem through decisions (Rardin, 1998). Identifying and scoping a problem is not a trivial matter, and is important in ensuring that the final solution that a decision is based upon, will in fact represent, and ultimately address the core problem. Taha

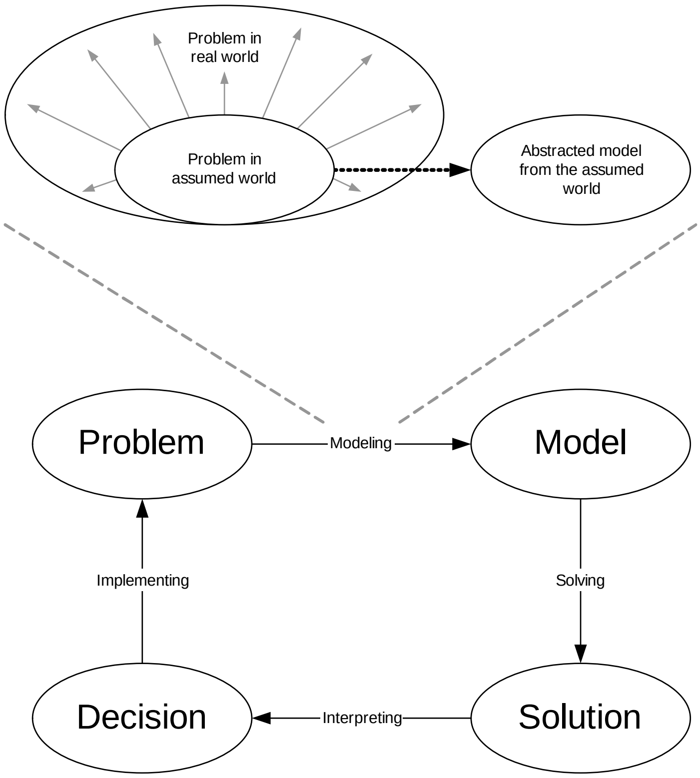


Figure 1.1: Operations Research cycle

(2003) expands the action of *modeling* in Figure 1.1 and illustrates how representations of the real world can easily be over-simplified. Interrelationships within the *real* world are so complex and abundant, that no one person can comprehend it in its entirety. We refer to the *problem in the real world* as the first level of abstraction. The human is a contextual being: the cultural, social and emotional context of an individual forms the individual's perception of the reality in which he or she exist. The second level of abstraction therefor represents the contextually sensitive view, referred here to as the *assumed reality*, that an individual has of the real problem. But even the abstract and fragmented view is often too complex to solve in its entirety. Through the actions of analyzing, and applying a methodology of *divide-and-conquer*, the individual scopes the problem in a structured way through simplifying assumptions. These assumptions may be justified in the absence of complete and accurate data about the assumed reality. The third level of abstraction is referred to as a *model*. The verb *modeling* therefor requires the problem solver to not only scope the problem, but also justify the endeavors to ensure that the assumed reality has been challenged to represent the real problem more comprehensively. This is illustrated through the arrows stretching the boundaries of the assumed reality towards the real world. Although the model can be any representation of a real problem, from scraps of paper with notes on them, a functional flow block diagram or process maps, in this thesis the term is used as a structured and mathematical model with an optimization intent.

Once the model is a true representation of the problem at hand, the decision maker can proceed to *solve* the model. It should be emphasized that only the model is solved, and not the problem itself. The availability and the ease of use of new generation optimization software have facilitated the process of solving models representing complex operational problems. The rapidly increasing processing power of computers brings the optimization opportunities right to the desk of the practitioner. The solution, however, is often but a list of numerical results.

The numerical solution, and its sensitivity to changes in parameters, requires careful consideration before recommendations and decisions are made, and is only considered as decision *support*. Implementation impacts, and possible change factors are considered before a final decision is made and implemented. The impact of the decision is then *assessed* so as to close the problem solving-cycle. Implemented changes may either address the original problem adequately, or may elicit new problems that require modeling, solving, and decision making.

1.2 *Intelligence* as the research driver

Freight carriers are sharing the road network with various modes of public transport. The use of private vehicles have rapidly increased. The increase can be attributed to both an increase in the number of trips undertaken, and increased journey lengths (Banister, 1995; Spence, 1998). Road network performance is negatively impacted by the higher usage of private vehicles and results in higher levels of congestion, and a significant reduction in operating speeds. Public transport performance is impacted negatively when operating speeds decrease, resulting in increased operating costs for the carriers, and thus impacting negatively on its attractiveness. As a result, the economically able part of the population turn to their private vehicles for a reliable source of transport, and unknowingly contributes to the hyper-congestion phenomenon.

Congestion does not only increase the stress levels of road users from a commuting point of view, but it also increases the complexity for vehicle and fleet managers overseeing the scheduling, routing, and optimization of their fleet concerns.

Carrier companies represent both public and private entities executing the logistic and distribution functions of freight. This thesis addresses the complexities of freight transport. Freight carriers are continuously expected to provide higher levels of service at lower rates, and therefor try to minimize their logistic costs, and maximize their profit. Sharing the road infrastructure with other vehicles such as private cars and public transport forces carriers to plan their freight routes more carefully. Enhanced vehicle routing and scheduling takes the congestion constraint into account and attempts to improve the vehicular utility through shorter routes and higher load factors. Software applications often do not provide adequate functionality by not being able to address complex business requirements such as companies having a fleet of vehicles that differ in capacity and/or running costs, and multiple scheduling where vehicles are allowed to complete a trip, return to the depot to renew it's capacity, i.e. offload goods collected, or loading goods to be delivered. The reason for software deficiencies are related to the extreme computational complexity when solving routing models. Human intervention is required to, for instance, split the fleet into vehicle categories that represent similar or the same capacity and/or costs. Each category is then solved independently, adjusting demand as customers are serviced by other categories. Human operators can also intervene by evaluating vehicular routes, and identifying vehicles that may be used for a second trip, and then schedule such vehicles accordingly. Although such interventions are mechanistic in nature, they require the time and effort of experienced individuals having a thorough understanding of vehicle routing so as to intervene wisely.

We refer to ourselves (in a more formal way) as *homo sapiens* — man the wise — and value our mental abilities to *think* and *reason* to assist us in improving our surroundings. We require our *thought processes* and *intelligence* to make decisions that will maximize the utility that we obtain from logistics — moving goods from points of manufacture to points of consumption that are geographically dispersed.

> *"What is mind? What is the relationship between mind and the brain? What is thought? What are the mechanisms that give rise to imagination? What is perception and how is it related to the object perceived? What are emotions and why do we have them? What is will and how do we choose what we intend to do? How do we convert intentions into action? How do we plan and how do we know what to expect from the future?"*—Albus (1999)

It seems clear from the quote by Albus (1999) that before one toss terms such as *thinking* and *planning* around, one should carefully consider how such actions take place, and how one intends to employ such actions to improve, for example, urban freight congestion.

1.2.1 Intelligence

In their leading text, Russell and Norvig (2003) introduces Artificial Intelligence (AI) as not only understanding the human intellect, but also building entities (or agents) that *are* intelligent. Although it encompasses a huge variety of subfields of study, with many varying definitions, the authors have categorized AI approaches in a two-dimensional framework represented in Figure 1.2.

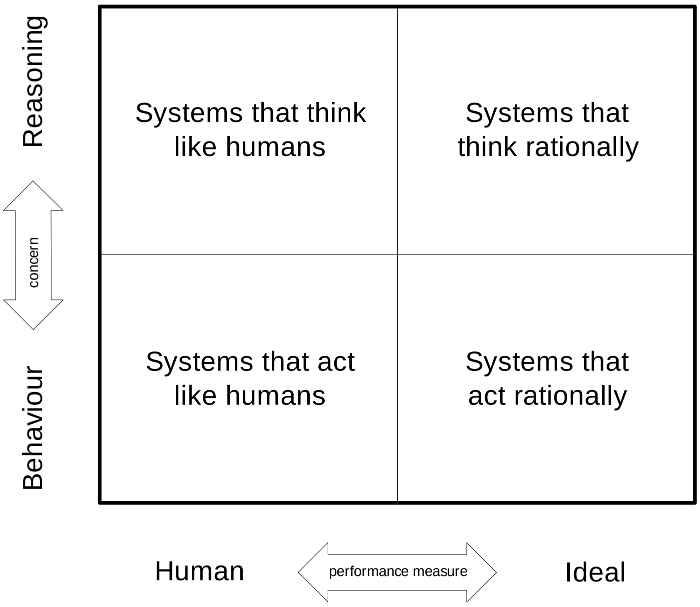


Figure 1.2: Categories of artificial intelligence (Adapted from **?**)

The top half of the framework is concerned with thought processes and reasoning, as opposed to the lower half that is concerned with the behavioral element of intelligence. The left side of the framework measures the success of an agent's intelligence against the fidelity of human performance. The right half establishes an *ideal* concept of intelligence as a benchmark, referred to as *rationality*. This is analogous to effectiveness — *doing the right things*. However, the *right* within *rationality* is only relative to what is known at the time of the *doing*.

An *agent* is something that acts. This thesis is concerned with the development of a computer agent that could intelligently intervene in the routing and scheduling of distribution vehicles. But how is it to be distinguished from mere *programming*? It should be able to operate autonomously, perceive the environment, persist over a period of time, and be able to adopt the goals and objectives of another entity. As an improvement on a basic agent, this thesis propose a *rational agent* that has a strategy to achieve the best possible outcome for a given objective, either known, or the expected outcome should some of the parameters be uncertain. The focus of the thesis is therefor not on understanding the human thought processes, but on creating a system that can think, and act rationally.

1.2.2 Complexity

Perfect rationality in modeling is often too difficult to attain due to too high computational demands when looking for exact solutions. Problems such as the routing and scheduling of vehicles can often not be solved exactly, and require the use of solution algorithms that provided approximate solutions where the optimality of the solution can neither be proved in advance, nor confirmed once a solution is found. The different opinions with regards to either finding an exact optimal solution versus settling for a *good enough* solution given a specific environment have led to the split that occurred between *Decision Theory* and *Artificial Intelligence* in the latter half of the twentieth century.

Decision Theory is the field of study where probability theory and utility theory are combined to present a formal framework for decision making under uncertainty. The field of operations research addresses complex management decisions rationally. The intention of the pure branch of decision theory is to obtain a rational decision, or a global optimum.

On the contrary, the complexity in finding a single optimum value led the pioneers of AI such as Herbert Simon (1916–2001) to prove that being able to find a *good enough* answer describes human behavior more accurately — and earned him the Nobel prize in economics in 1978. And although the computational ability of computers have increased dramatically over the past decade, the intention is still to assist mere mortal logistics decision makers to improve their ability to manage distribution fleets.

1.3 Formulating the research question

The primary research question that this thesis intends to answer is *whether it is feasible to develop a rational and intelligent agent to schedule a predefined variant of the VRP.* In order to answer the question, a number of secondary research questions will be stated in terms of the concept of an *intelligent agent.*

In his paper on the engineering of mind, Albus (1999) identifies four functional elements of an intelligent system.

Sensory perception — accepting input data from both outside and from within the system. The data is then transformed through classification and clustering into meaningful representations of the real world. The first secondary research question addresses the analysis of input data and is stated as follows:

> *How should customer parameters be clustered so that meaningful classification can be done prior to executing the solution process?*

Behavior generation — planning and controlling actions so that goals are achieved. An intelligent agent accepts task with goals, objects and priorities. The tasks are then broken up into jobs and, along with resources, are assigned to agents. Hypothetical plans are created and simulated to predict the outcome of the plans. The simulated results are evaluated, and the agent selects the best expected hypothesized plan. In terms of this thesis an agent refers to computational elements that plan and control the execution of a routing algorithm, correcting for errors and perturbations along the way. The planning processes of the agent are heuristics and metaheuristics that attempt to converge to optimal vehicle routes and schedules. This lead to another secondary research question:

> *How can heuristics and metaheuristics be used to establish vehicle routes and schedules in a complex and constrained environment?*

Value judgement — the computation of a predefined set of costs, risks, benefits, and or penalties related to the vehicle routes. In operations research terms these computational expressions are referred to as the objective function(s). The third secondary research question is derived from value judgement:

> *What should constitute the objective function of the model so that the real problem is adequately represented?*

World modeling — an overall strategy that uses input parameters and variables to update a knowledge database. Data is used to query the behavior generation of plans regarding current routes and schedules. The strategy further simulates possible results of future plans after analyzing the current plans. Simulated results are evaluated, using the value judgement, so the best expected plan for execution can be selected. After execution, the strategy allows for sensory expectations to be created regarding future actions — analogous to bumping your feet against an obstacle in the dark. After stumbling, and reacting to the pain, you lift your feet unnaturally high so as to avoid the next obstacle. The fourth and fifth, probably the most challenging secondary research questions addresses the agents ability to learn from the past and improve in future:

What critical parameters influence the agent's learning, and should therefor be included in creating future expectations?

How are future expectations created from the past performance?

1.4 Research design and methodology

The process diagram in Figure 1.3 provides an overview of an intelligent agent's decision process. The agent in this thesis will be a hybrid computerized solution algorithm that has

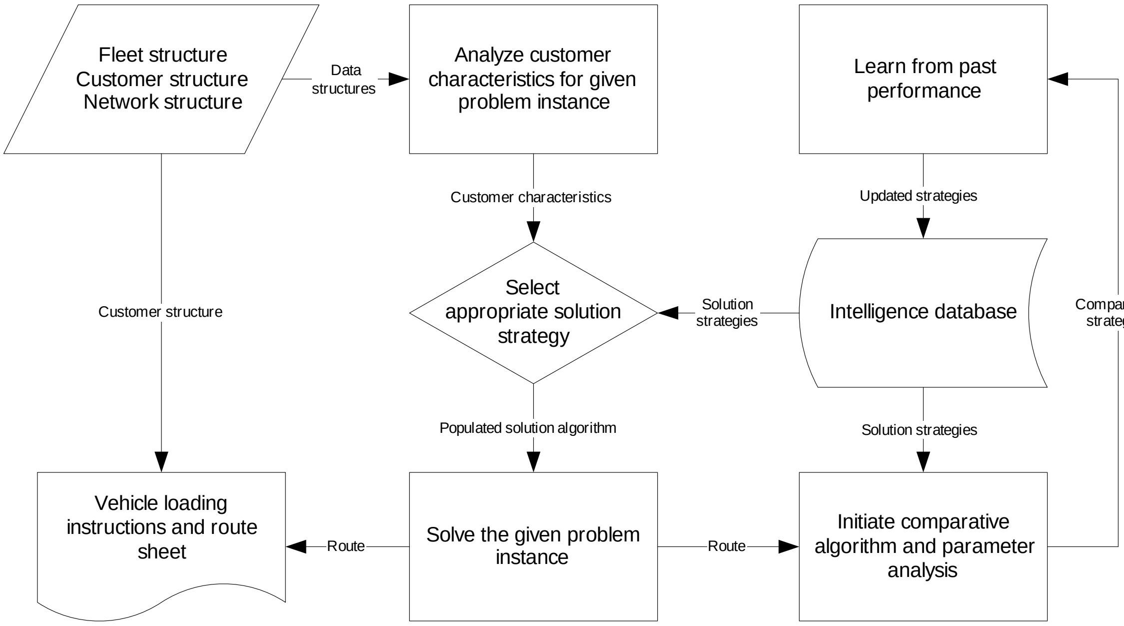


Figure 1.3: Overview of the intelligent agent's decision process

the following inputs:

- Fleet structure
- Customer structure, i.e. demand quantity, geographical location, time windows
- Network structure, derived from customers' geographical locations

The algorithm will analyze the clustering characteristics of the geographical distribution of customers. Based on the randomness (or clusteredness) of the distribution and the time window characteristics of the customers, the algorithm will select an appropriate solution strategy — a combination of a metaheuristic solution algorithm, along with its appropriate parameter values. The problem instance is solved, and the solution is interpreted and presented in a useful loading instruction and route sheet. Behind the scenes the algorithm will initiate comparative analysis of the proposed solution strategy by solving the provided problem instance with various metaheuristics and various parameter values for each metaheuristic. The algorithm will then learn from these analysis through a neural network, and update the intelligence database by recommending new solution strategies for the given problem instance, or reiterating current solution strategies.

The algorithm will be coded using the *MATLAB*® development environment. The analysis and solution components, and the comparative analysis components will run on separate computer processors to optimize for speed and in doing so, address the computational complexity of the hybrid algorithm.

1.5 The structure of the thesis

To elaborate on the exact nature of the research problem, Chapter 2 reviews literature on the VRP and its variants. The chapter concludes with the mathematical formulation of the *Capacitated Heterogeneous Fleet Vehicle Routing Problem with Multiple Soft Time Windows and Probabilistic Travel and Service Time* as addressed in this thesis. The review of solution algorithms, both exact and approximate, are conducted in Chapter 3, concluding with the recommendation of two metaheuristic solution algorithms, each covered in more detail in later chapters. The analysis of the customer structure is reviewed in Chapter 7, and the chapter proposes an algorithm to determine the level of clusteredness of a customer network. The algorithm is tested by analyzing benchmark data sets provided for pre-defined problem instances in literature.

Chapters 4 through 6 is dedicated to the development of various metaheuristic solution algorithms. Chapter 4 develops an improved initial solution algorithm to enhance the computational performance of the Tabu Search solution algorithm, developed in Chapter 5. The Genetic Algorithm is less sensitive to the quality of an initial solution, and is treated independently in Chapter 6. For each metaheuristic the various parameters are discussed, and default values proposed. The respective algorithms are discussed at high level, followed by detailed discussions of algorithmic particularities, and concluded by testing and validating the algorithm through benchmark data sets.

The integration of the algorithms, as well as the agent's ability to learn from repetitive decision making is covered in Chapter 8. The thesis is concluded in Chapter 9 with a critical analysis of the research contribution, and setting a research agenda.

Chapter 2

The *Vehicle Routing Problem*: origins and variants

Rardin (1998) states that the organizing of a collection of customer locations, jobs, cities, or points, into sequences and routes are among the most common discrete optimization problems. The first of the two review chapters focus on the origins and the mathematical formulation of the VRP and its variants.

2.1 The origins of the basic VRP

2.1.1 The Traveling Salesman Problem (TSP)

The simplest, and probably most famous of routing problems known to researchers is the TSP that seeks a minimum-total-length route visiting every one of N points in a given set $V = \{1, 2, \ldots, N\}$ exactly once across an arc set A. The distance between all point combinations in A, (i, j), where $(i, j) \in V | i \neq j$, is known. In the notation introduced by Rardin (1998), the symbol '$\triangleq$' denotes *defined to be*. With the decision variable x_{ij} defined as:

$$x_{ij} \triangleq \begin{cases} 1 & \text{if a salesman travels from node } i \text{ to node } j \text{, where } i, j = \{1, 2, \ldots, N\} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

we formulate the problem as

$$\min z = \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{2.2}$$

subject to

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall j \in \{2, \ldots, N\} \tag{2.3}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall i \in \{2, \ldots, N\} \tag{2.4}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad \forall S \subset V \tag{2.5}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \{2, \ldots, N\} \tag{2.6}$$

The objective of the problem minimizes the total distance traveled in (2.2). Each node must be visited exactly once according to (2.3) and (2.4), also referred to as *degree constraints*. Subtours are eliminated through the introduction of (2.5). The $|S|$ denotes the number of elements in the subset S. Schrage (2002) states that there are of the order 2^n constraints of type (2.5), as opposed to the alternative in (2.7)

$$u_j \geq u_i + 1 - (1 - x_{ij})n \qquad \forall j \in \{2, \ldots, N\} | j \neq i \tag{2.7}$$

of which there are of the order $N - 1$ constraints. Only a few of the former type constraints will be binding in the optimum. Padberg and Rinaldi (1987) therefor propose an efficient and effective iterative process of adding violated constraints of type (2.5) as needed.

Although a number of TSP variations exist, our interest is in the variant where multiple salesmen are routed simultaneously.

2.1.2 The Multiple Traveling Salesman Problem (MTSP)

The MTSP is similar to the notoriously difficult TSP that seeks an optimal tour of N cities, visiting each city exactly once with no sub-tours. In the MTSP, the N cities must be partitioned into M tours, with each tour resulting in a TSP for one salesperson. The MTSP is more difficult than the TSP because it requires determining which cites to assign to each salesperson, as well as the optimal ordering of the cities within each salesperson's tour (Carter and Ragsdale, 2005; Kara and Bektas, 2005). Consider a complete directed graph $G = (V, A)$ where V is the set of N nodes (or cities to be visited), A is the set of arcs and $C = (c_{ij})$ is the cost (distance) matrix associated with each arc $(i, j) \in A$. The cost matrix can be symmetric, asymmetric, or Euclidean. The latter refers to the straight-line distance measured between the two geographically dispersed nodes. There are M salesmen based at the depot, denoted as node 1. The single depot MTSP consists of finding tours for the M salesmen subject to each salesman starting and ending at the depot, each node is located in exactly one tour, and the number of nodes visited by a salesman lies within a predetermined time (or distance) interval. The objective is to minimize the cost of visiting all the nodes. We define the decision variable, x_{ij}, in (2.1). For any salesman, u_i denotes the number of nodes visited on that salesman's route up to node i, with corresponding parameters K and L denoting the minimum and maximum number of nodes visited by any one salesman, respectively. We can therefor state that $1 \leq u_i \leq L$ when $i \geq 2$, and when $x_{i1} = 1$, then $K \leq u_i \leq L$. The following Integer Linear Program (ILP) formulation is proposed by Kara and Bektas (2005).

$$\min z = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.8}$$

subject to

$$\sum_{j=2}^{N} x_{1j} = M \tag{2.9}$$

$$\sum_{i=2}^{N} x_{i1} = M \tag{2.10}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall j \in \{2,\ldots,N\} \tag{2.11}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall i \in \{2,\ldots,N\} \tag{2.12}$$

$$u_i + (L-2)x_{1i} - x_{i1} \leq L-1 \qquad \forall i \in \{2,\ldots,N\} \tag{2.13}$$

$$u_i + x_{1i} + (2-K)x_{i1} \geq 2 \qquad \forall i \in \{2,\ldots,N\} \tag{2.14}$$

$$x_{1i} + x_{i1} \leq 1 \qquad \forall i \in \{2,\ldots,N\} \tag{2.15}$$

$$u_i - u_j + Lx_{ij} + (L-2)x_{ji} \leq L-1 \qquad \forall i,j \in \{2,\ldots,N\} | i \neq j \tag{2.16}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in \{2,\ldots,N\} \tag{2.17}$$

The objective in (2.8) minimizes the total cost of traveling to all nodes, while constraints (2.9) and (2.10) ensures that all M salesmen are allocated routes. Degree constraints are imposed by (2.11) and (2.12). The MTSP-specific constraints (2.13) and (2.14) are referred to as *bounding constraints* and Kara and Bektas (2005) introduce these as the upper and lower bound constraints on the number of nodes visited by each salesman. The value of u_i is initialized to 1 if and only if node i is the first node on the tour of any salesman. Inequality (2.15) forbids a salesman to only visit a single node on its tour. The formation of subtours between all nodes in $V \setminus \{1\}$ (all nodes except the depot) are eliminated by (2.16) as it ensures that $u_j = u_i + 1$ if and only if $x_{ij} = 1$. They are also referred to as Subtour Elimination Constraints (SEC).

Next we consider a variant where each of the M salespeople has a predefined, yet similar, capacity. An analogy is having salespeople traveling with samples in their vehicles. Not only do their cars have limited space for the samples, but each customer visited may require a different number of the samples. As a variant of the MTSP it is referred to as the Capacitated Multiple Traveling Salesman Problem (CMTSP), but in the context of this thesis the vehicular related name, Vehicle Routing Problem (VRP), is preferred.

2.1.3 The Vehicle Routing Problem (VRP)

The distribution problem in which vehicles based at a central facility (depot) are required to visit — during a given time period — geographically dispersed customers in order to fulfill known customer requirements are referred to as the VRP (Christofides, 1985). The main objective of the VRP is to minimize the distribution costs for individual carriers, and can be described as the problem of assigning optimal delivery or collection routes from a depot to a number of geographically distributed customers, subject to constraints (**?**). The most basic version of the VRP have also been called *vehicle scheduling*, *truck dispatching*, or simply the *delivery problem*. A number of different formulations appear in the authoritative work of Christofides (1985). The basic problem can be defined with $G = (V, A)$ being a directed graph where $V = \{v_1, \ldots, v_N\}$ is a set of vertices representing N customers, and with v_1 representing the depot where M identical vehicles, each with capacity Q, are located (**?**). $E = \{(v_i, v_j)|v_i, v_j \in V, i \neq j\}$ is the edge set connecting the vertices. Each vertex, except for the depot $(V\backslash\{v_1\})$, has a non-negative demand q_i and a non-negative service time s_i. A matrix $C = (c_{ij})$ is defined on A. In some contexts, c_{ij} can be interpreted as travel cost, travel time, or travel distance for any of the identical vehicles. Hence, the terms cost, time, and distance are used interchangeably, although t_{ij} denotes the travel time between nodes i and j in the formulation provided below. The basic VRP is to route the vehicles one route per vehicle, each starting and finishing at the depot, so that all customers are supplied with their demands and the total travel cost is minimized. Although Christofides (1985) presents three different formulations from the early 1980s, the following mathematical formulation of the VRP is adapted from Bodin et al. (1983) and Filipec et al. (1998). During this period little changes were made to the formulation of the problem. The decision variable, x_{ij}^k is defined as

$$x_{ij}^k \triangleq \begin{cases} 1 & \text{if vehicle } k \text{ travels from node } i \text{ to } j\text{, where} \\ & i,j \in \{1,2,\ldots,N\} | i \neq j, \text{ and } k \in \{1,2,\ldots,K\} \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

$$\min z = \sum_{i=0}^{N} \sum_{\substack{j=0 \\ j\neq i}}^{N} \sum_{k=1}^{K} c_{ij} x_{ij}^k \tag{2.19}$$

subject to

$$\sum_{i=0}^{N} \sum_{k=1}^{K} x_{ij}^k = 1 \qquad \forall j \in \{1,\ldots,N\} \tag{2.20}$$

$$\sum_{j=0}^{N} \sum_{k=1}^{K} x_{ij}^k = 1 \qquad \forall i \in \{1,\ldots,N\} \tag{2.21}$$

$$\sum_{i=0}^{N} x_{ip}^k - \sum_{j=0}^{N} x_{pj}^k = 0 \qquad \forall p \in \{1,\ldots,N\}, k \in \{1,\ldots,K\} \tag{2.22}$$

$$\sum_{j=0}^{N} q_j \left(\sum_{i=0}^{N} x_{ij}^k \right) \leq Q \qquad \forall k \in \{1,\ldots,K\} \tag{2.23}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} t_{ij} x_{ij}^k \leq D \qquad \forall k \in \{1,\ldots,K\} \tag{2.24}$$

$$\sum_{j=1}^{N} x_{0j}^k \leq 1 \qquad \forall k \in \{1,\ldots,K\} \tag{2.25}$$

$$\sum_{i=1}^{N} x_{i0}^k \leq 1 \qquad \forall k \in \{1,\ldots,K\} \tag{2.26}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i,j \in \{1,\ldots,N\}, k \in \{1,\ldots,K\} \tag{2.27}$$

The degree constraints are represented by (2.20) and (2.21). Route continuity is enforced by (2.22) as once a vehicle arrived at a node, it must also leave that node. No one vehicle can service customer demands that exceeds the vehicle capacity in (2.23). A maximum route length is limited by (2.24). Constraints (2.25) and (2.26) ensures that each vehicle is scheduled no more than once.

2.2 Variants of the VRP

The basic VRP makes a number of assumptions, including utilizing a homogeneous fleet, a single depot, one route per vehicle, etc. These assumptions can be eliminated by introducing additional constraints to the problem. This implies increasing the complexity of the problem, and, by restriction, classifies the extended problem as an *np*-hard problem. It should be noted that most of these additional constraints are often implemented in isolation, without integration, due to the increased complexity of solving such problems. In the next few sections, these variants are introduced in isolation, before proposing an integrated formulation in Section 2.3.

2.2.1 The concept of time windows

A *time window* can be described as a window of opportunity for deliveries. It is an extension of the VRP that has been researched extensively (Ibaraki et al., 2005; Taillard, 1999; Taillard et al., 1997; Tan et al., 2001c). A time window is the period of time during which deliveries can be made to a specific customer i, and has three main characteristics:

- Earliest allowed arrival time, e_i, also referred to as the *opening time*
- Latest allowed arrival time, l_i, also referred to as the *closing time*
- Whether the time window is considered *soft* or *hard*

Consider the example, illustrated in Figure 2.1, where customer i requests delivery between 07:30 and 17:00. To distinguish between the actual and the specified times of arrival, the

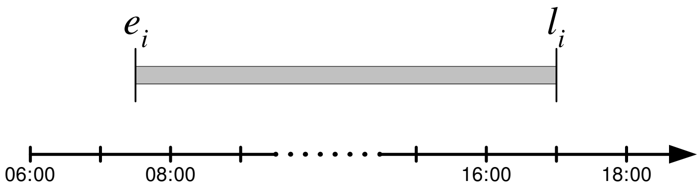


Figure 2.1: Double sided hard time window

variable a_i denotes the actual time of arrival at node i. Should the actual arrival time at node i, denoted by a_i, be earlier than the earliest allowed arrival at the node, e_i, then the vehicle will incur a waiting time, w_i, which can be calculated as $w_i = \max\{0, e_i - a_i\}$. The introduction of time windows to the basic VRP sees the introduction of three new constraints.

$$a_0 = w_0 = s_0 = 0 \tag{2.28}$$

$$\sum_{k=1}^{K} \sum_{i=0;i\neq j}^{N} x_{ij}^k (a_i + w_i + s_i + t_{ij}) \leq a_j \qquad \forall j \in \{1, 2, \ldots, N\} \tag{2.29}$$

$$e_i \leq (a_i + w_i) \leq l_i \qquad \forall i \in \{1, 2, \ldots, N\} \tag{2.30}$$

Constraint (2.28) assumes that vehicles are ready and loaded by the time the depot opens, which is indicated as time 0 (zero). Constraint (2.29) calculates the actual arrival time, while (2.30) ensures that each customer i is serviced within its time window.

When both an earliest and latest allowed arrival is stipulated, the time window is referred to as *double sided*. If no arrivals are allowed outside of the given parameters, the time window is said to be *hard*, as is the case in Figure 2.1. When delivery is allowed outside the specified time window, the time window is said to be *soft*, and customer i may penalize lateness at a cost of α_i (Koskosidis et al., 1992). Customer i may specify a maximum lateness, L_i^{max}. The example illustrated in Figure 2.2 sees customer i specifying a time window between 07:30 and 15:30. The customer will, however, allow late deliveries until 17:00. A hard time window

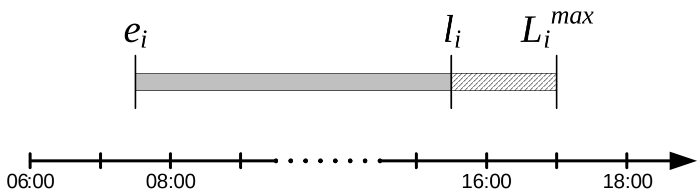


Figure 2.2: Soft time window

is therefor a special type of soft time window where $L_i^{max} = 0$. Should a vehicle arrive after the latest allowed arrival time, l_i, but prior to the maximum lateness, L_i^{max}, the lateness at node i, L_i, can be calculated as $L_i = \max\{0, a_i - l_i\} | a_i \leq L_i^{max}$. The lateness is penalized by introducing a penalty term to the VRP objective function (2.19), resulting in(2.31).

$$\min z = \sum_{i=0}^{N} \sum_{j=0,j\neq i}^{N} \sum_{k=1}^{K} c_{ij} x_{ij}^k + \sum_{i=1}^{N} \alpha_i \times \max\{0, L_i\} \tag{2.31}$$

The time window for the depot, node 0, can be specified. The case illustrated in Figure 2.3 sees the depot specifying operating hours (time window) from 06:00 to 18:00, while the first customer on the route, customer 1, specifies a time window between 07:00 and 09:00, and the last customer, customer N, requests delivery between 15:00 and 17:00.

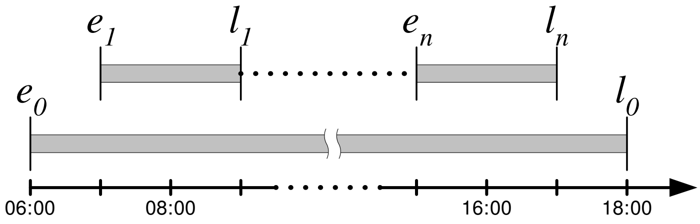


Figure 2.3: Time window for the depot, node 0

Should a customer specify multiple time windows, an indexing symbol, a, is introduced as superscript to the earliest and latest allowed arrival times, respectively, where $a \in \{1, 2, \ldots, A\}$ in which A indicates the maximum number of time windows allowed for each customer. Consider the example where customer n requests delivery either between 06:30 and 09:00, or between 16:00 and 17:30 as illustrated in Figure 2.4. This example is

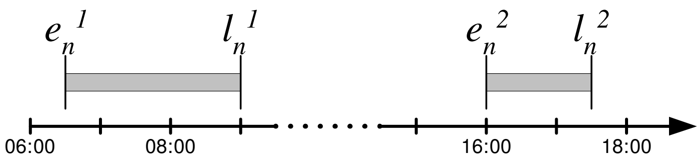


Figure 2.4: Multiple time windows

typical of residents requesting home shopping deliveries outside business hours. The formulation changes with the introduction of the decision variable

$$\psi_i^a \triangleq \begin{cases} 1 & \text{if the } a^{\text{th}} \text{ time window of customer } i \text{ is used, where } i \in \{1, 2, \ldots, N\}, \\ & a \in \{1, 2, \ldots, A\} \\ 0 & \text{otherwise.} \end{cases}$$

To ensure that the decision variable is appropriately enforced in the formulation, we change constraint (2.30) to distinguish between different time windows, as proposed in (2.32)

$$e_i^a - (1 - \psi_i^a)\, M \leq (a_i + w_i) \leq l_i^a + (1 - \psi_i^a)\, M \quad \forall i \in \{1, 2, \ldots, n\}, a \in \{1, 2, \ldots, A\} \tag{2.32}$$

where M is a sufficiently large number, typically greater than the scheduling horizon. An enforcement of a single time window for each customer is required, and is subsequently introduced in (2.33).

$$\sum_{a=1}^{A} \psi_i^a = 1 \qquad \forall i \in \{1, 2, \ldots, N\} \tag{2.33}$$

2.2.2 Capacity constraints and vehicle characteristics

Gendreau et al. (1999) propose a solution methodology for cases where the fleet is heterogeneous, that is, where the fleet is composed of vehicles with different capacities and costs. Their objective is to determine what the optimal fleet composition should be, and is referred to as either a Heterogeneous Fleet Vehicle Routing Problem (HVRP) or a Fleet Size and Mix Vehicle Routing Problem (FSMVRP). Liu and Shen (1999b) adds time windows in their problem application and refer to the problem as a Fleet Size and Mix Vehicle Routing Problem with Time Windows (FSMVRPTW). In yet another paper, Liu and Shen (1999a) refers to the heterogeneous fleet variant as the Vehicle Routing Problem with Multiple Vehicle Types and Time Windows (VRPMVTTW). Taillard (1999) formulates the Vehicle Routing Problem with a Heterogeneous fleet of vehicles (VRPHE) where the number of vehicles of type t in the fleet is limited; the objective being to optimize the utilization of the given fleet. Salhi and Rand (1993) incorporate vehicle routing into the vehicle composition problem, and refer to it as the Vehicle Fleet Mix problem (VFM).

The implication of a heterogeneous fleet on the standard VRP is that T type of vehicles are introduced, with $t \in \{1, 2, \ldots, T\}$. The vehicle capacity parameter p is changed. The new parameter, p_t, represents the capacity of vehicles of type t, resulting in each vehicle k having a unique capacity, p_k. The use of one vehicle of type t implies a fixed cost f_t. A unique fixed cost, f_k, is introduced for each vehicle k, based on its vehicle type. The objective function changes to

$$\min z = \sum_{i=0}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} \sum_{k=1}^{K} c_{ij} x_{ij}^{k} + \sum_{k=1}^{K} \sum_{j=1}^{n} f_k x_{0j}^{k} \tag{2.34}$$

while (2.23) changes to indicate the new capacity parameter

$$\sum_{i=1}^{n} q_i \left(\sum_{j=0}^{n} x_{ij}^{k} \right) \leq p_k \qquad \forall k = \{1, 2, \ldots, K\} \tag{2.35}$$

Taillard (1999) introduces a variable c_{ijt} to represent the cost of traveling between nodes i and j, using a vehicle of type t. It is possible to introduce the variable portion of the vehicle cost into the objective function proposed in (2.34). The introduction will lead to (2.36)

$$\min \sum_{i=0}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} \sum_{k=1}^{K} \sum_{t=1}^{T} c_{ijt} x_{ij}^{k} \xi_{t}^{k} + \sum_{k=1}^{K} \sum_{j=1}^{n} f_k x_{0j}^{k} \tag{2.36}$$

where

$$\xi_t^k \triangleq \begin{cases} 1 & \text{if vehicle } k \text{ is of type } t \text{, where } k = \{1,2,\ldots,K\} \text{, and } t = \{1,2,\ldots,T\} \\ 0 & \text{otherwise} \end{cases}$$

2.2.3 Uncertainty in vehicle routing

The statements in Section 2.1.3 do not adequately describe a variety of practical VRP situations where one or several parameters are uncertain. Powell (2003) confirms that research into routing and scheduling algorithms, which explicitly captures the uncertainty of future decisions made now, is extremely young. Laporte et al. (1992), Lambert et al. (1993), and Ong et al. (1997) provide examples including vehicles collecting random quantities at various customers; and customers being visited on a random basis. A vehicle incurs a penalty proportional to the duration of its route in excess of a predetermined constant B — typical of applications where drivers are paid overtime for work done after normal hours. Laporte et al. (1992) propose an attractive and relatively simple chance constrained model (from a computational point of view). However, as the expected cost related to excess route duration needs to be taken into account, this thesis reverts to proposing a stochastic programming model with recourse.

First stage decisions made are the number of vehicles required, as well as their respective routes. Once the random travel time and service time variables are realized in the second stage, penalties are incurred for the excess duration. The following variables are defined.

$$x_{ij}^k \triangleq \begin{cases} 1 & \text{if vehicle } k \text{ travels from node } i \text{ to } j \text{, where} \\ & i,j = \{1,2,\ldots,n\} | i \neq j \text{, and } k = \{1,2,\ldots,K\} \\ 0 & \text{otherwise} \end{cases}$$

$$z_i^k \triangleq \begin{cases} 1 & \text{if node } i \text{ is visited by vehicle } k \text{, where } i = \{1,\ldots,n\}, k = \{1,\ldots,m\} \\ 0 & \text{otherwise} \end{cases}$$

$\tilde{\xi} \triangleq$ a vector of random variables corresponding to travel and service times. Each realization r of $\tilde{\xi}$, denoted by ξ^r, is referred to as a *state of the world* (Kall and Wallace, 1994)

$\Xi \triangleq$ the finite support of $\tilde{\xi}$ such that $\Xi = \left\{1,2,\ldots,\xi^r,\ldots,\xi^R\right\}$ where R is the total number of states in the problem *world*

$y^k(\tilde{\xi}) \triangleq$ the excess duration of route k as a function of the realization of $\tilde{\xi}$

$c_{ij}^k \triangleq$ the travel cost from node i to j with vehicle k, where $i, j = \{1, \ldots, n\}, k = \{1, \ldots, K\}$

$t_{ij}^k(\tilde{\xi}) \triangleq$ the travel time from node i to j with vehicle k, where $i, j = \{1, \ldots, n\}, k = \{1, \ldots, K\}$ expressed as a function of the realization of $\tilde{\xi}$

$\tau_i^k(\tilde{\xi}) \triangleq$ the service time at node i with vehicle k, where $i = \{1, \ldots, n\}, k = \{1, \ldots, K\}$, expressed as function of the realization of $\tilde{\xi}$

$\beta^k \triangleq$ the positive unit penalty cost for excess duration traveled by vehicle k, where $k = \{1, \ldots, m\}$

$f^k \triangleq$ the fixed cost of vehicle k, where $k = \{1, \ldots, K\}$

$B^k \triangleq$ the maximum time for route k over which a penalty is incurred, where $k = \{1, \ldots, K\}$

The model is then

$$\min z = \sum_{k=1}^{K} f^k z_0^k + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{k=1}^{K} c_{ij}^k x_{ij}^k + E_{\tilde{\xi}} \left(\sum_{k=1}^{K} \beta^k y^k(\tilde{\xi}) \right) \tag{2.37}$$

subject to

$$\sum_{k=1}^{K} z_i^k = 1 \qquad \forall i \in \{1, \ldots, n\} \tag{2.38}$$

$$\sum_{j=1}^{n} \left(x_{0j}^k + x_{j0}^k \right) = 2 z_0^k \qquad k \in \{1, \ldots, K\} \tag{2.39}$$

$$\sum_{j=1}^{n} \left(x_{ij}^k + x_{ji}^k \right) = 2 z_i^k \qquad \forall i \in \{1, \ldots, n\}, k \in \{1, \ldots, K\} \tag{2.40}$$

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{ij}^k \leq |S| - 1 \qquad S \subset V, 3 \leq |S| \leq n-3, k = \{1, \ldots, K\} \tag{2.41}$$

$$B^k - \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} t_{ij}^k(\tilde{\xi}) x_{ij}^k - \frac{1}{2} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\tau_i^k(\tilde{\xi}) + \tau_j^k(\tilde{\xi}) \right) x_{ij}^k + y^k(\tilde{\xi}) \geq 0 \tag{2.42}$$

$$\forall k \in \{1, \ldots, K\}, \tilde{\xi} \in \Xi$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i, j \in \{1, \ldots, n\}, k \in \{1, \ldots, K\} \tag{2.43}$$

$$z_i^k \in \{0, 1\} \qquad \forall i \in \{1, \ldots, n\}, k \in \{1, \ldots, K\} \tag{2.44}$$

$$y^k \left(\tilde{\xi} \right) \geq 0 \qquad \forall k \in \{1, \ldots, K\}, \tilde{\xi} \in \Xi \tag{2.45}$$

The objective function minimizes total cost in (2.37) that includes fixed vehicle costs, travel costs, as well as the expected penalty costs as a result of exceeded route duration. All vehicles must be routed according to (2.38), while (2.39) calculates the number of routed vehicles. Degree constraints are introduced in (2.40). Subtours are eliminated through (2.41) where the reader may infer that $n > 6$. Constraint (2.42) combined with (2.45) implies a penalty to be calculated for vehicle k, but only if the total route length including service times exceed B^k.

2.2.4 Time-dependent travel time

Although unpredictable events such as accidents and vehicle breakdowns render travel times as stochastic, the candidate postulates that the subtle, yet partially predictable event of congestion during peak hours of the day requires more attention. The assumption is made that by addressing the time-dependent nature of travel times, a modeling approach that is a stronger approximation of the actual real-world conditions of vehicle routing and scheduling than by catering for stochastic travel times, will be achieved.

Hill and Benton (1992) review the two main approaches in estimating travel distance between two nodes i and j, denoted by d_{ij}, namely Minkowski distance and Pythagorean distance. The former is presented in (2.46).

$$d_{ij} = \left[|x_i - x_j|^\omega + |y_i - y_j|^\omega\right]^{\frac{1}{\omega}} \tag{2.46}$$

When ω is 2, the Minowski distance, denoted by d_{ij}, is the Pythagorean distance. When ω is 1, the Minowski distance is the *city-block* right-angled distance. In (2.46) the coordinate pair (x_i, y_i) of each node i is required. A similar approach can be followed if only latitude and longitude data is available, i.e. from a Geographical Information System (GIS) database. The problem, however, is that researchers often reduce vehicle travel speed to an approximate speed, denoted by r_c, and simply apply the scalar transformation of distance in (2.47) to find the travel time between the two nodes,

$$t_{ij} = \frac{d_{ij}}{r_c} \tag{2.47}$$

without cognisance of an acceleration stage to get onto the road, the cruising stage, and the deceleration stage at the destination node (Assad, 1988). If the three stages were to be acknowledged, d_c denotes the distance required for the vehicle to reach its cruising speed, and α denotes the acceleration, a more appropriate way of calculating the travel time is given in (2.48).

$$t_{ij} = \begin{cases} 2\left(\frac{d_{ij}}{\alpha}\right)^{\frac{1}{2}} & \text{if } d_{ij} \leq 2d_c \\ \frac{d_{ij}}{r_c} + \frac{r_c}{\alpha} & \text{if } d_{ij} > 2d_c \end{cases} \tag{2.48}$$

In most metropolitan areas, travel times are much longer during the start and end of workday *rush hours*, especially on main arterial routes. If one were to inflate all route times equally during peak periods, one would be able to route and schedule vehicles without taking time-dependent travel times into consideration, and not compromise optimality of routes. However, road networks are unevenly congested, i.e. traveling from A to B during the morning rush hour traffic might be more congested than when traveling from B to A at the same time.

Malandraki and Daskin (1992) state that the travel time is not only a function of the distance, but should take the time of day into account as well. Ichoua et al. (2003) state that research on time-dependent problems started towards the end of the 1950s with references to the *time-dependent shortest path problem*, the *time-dependent path choice problem*, and the Time Dependent Traveling Salesman Problem (TDTSP). Of the earliest research found on the Time Dependent Vehicle Routing Problem (TDVRP) is Hill et al. (1988), followed by Hill and Benton (1992). In their papers customer *nodes* were assigned time-dependent piecewise constant speeds — these speeds reflect the traveling speed *surrounding* the nodes. The edge travel time between two nodes were derived as the average speed of the two nodes concerned. At the time Hill and Benton (1992) attribute the lack of time-dependent travel time research to:

- Immense efforts to estimate travel time parameters
- Prohibitive data storage requirements
- Inefficient solution algorithms

Malandraki and Daskin (1992) formulate an elegant variant of the Vehicle Routing Problem with Time Windows (VRPTW) with the introduction of piecewise constant travel times on the edges. Approaches to accommodate time-dependent travel times mentioned so far all allow *passing*: the event where one vehicle my *pass* another vehicle on the same edge although it started later than the vehicle it passed, but in a different time period with shorter traveling time.

Ahn and Shin (1991) use similar notation as used in the introduction of the VRPTW, and also introduce:

$\tau_{ij}(x) \triangleq$ travel time from node i to node j via arc $(i, j) \in A$, given that the trip starts from node i at time x

$s_i \triangleq$ the constant service time at node i

$t_i \triangleq$ the time at which service begins at node i

$A_{ij}(t_i) \triangleq$ arrival time at node j through arc $(i, j) \in A$ given t_i, that is $A_{ij}(y) = y + s_i + \tau_{ij}(t_i + s_i)$

$d_i \triangleq$ the *effective* latest service start time at node i that allows us to maintain the feasibility of a current route

Each customer i is to be serviced within its time window $[e_i, l_i]$. The internode travel time $\tau_{ij}(\cdot)$ and the arrival time $A_{ij}(\cdot)$ are functions of the departure time representing time-dependent congestion levels. In this thesis multiple links are not considered. The non-passing property can be expressed as:

> *For any two nodes i and j, and any two service start times x and y at node i such that $x < y$, $A_{ij}(x) < A_{ij}(y)$ must hold, that is, earlier departure from node i guarantees earlier arrival at node j.*

Raw travel time data in the form of a step function is not appropriate for use in the routing of vehicles, as it only provides average travel time data for specific time periods. In such data sources, let:

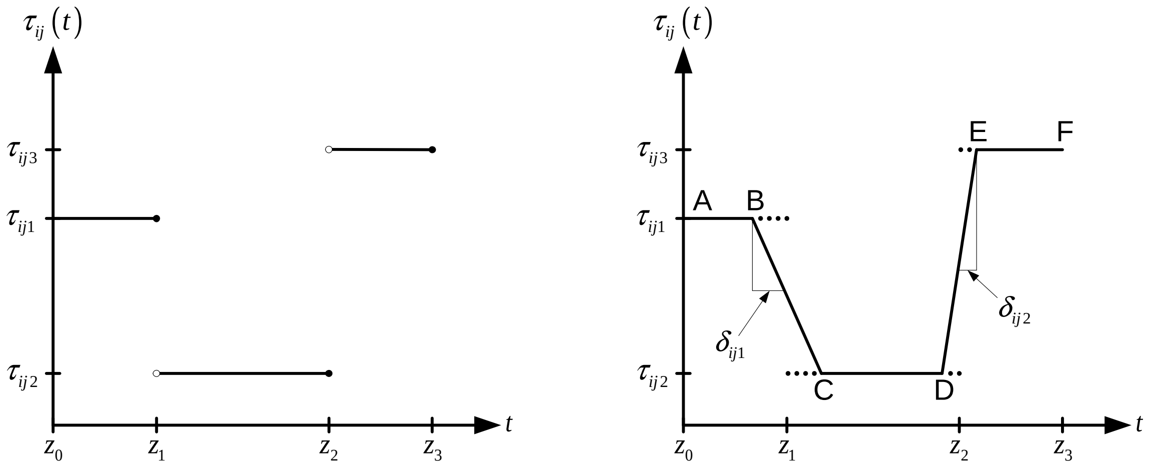
$\tau_{ijk} \triangleq$ the shortest travel time from node i to node j if the start time at node i is in time slot Z_k, where $i, j \in A$, and $k \in \{1, 2, \ldots, K\}$,

where the day (planning horizon) is divided into time slots such that

$$Z_k = [z_{k-1}, z_k] \qquad \forall k \in \{1, 2, \ldots, K\},$$

where the interval $[z_0, z_K]$ reflects the full day, or planning horizon under consideration. Figure 2.5 is used for illustrative purposes. The travel time, being a function of the time of day, is not continuous in the point z_k and may lead to passing if travel time decrease for the $k + 1^{\text{th}}$ segment. To obtain a smoothed travel time function, let:

$\tau_{ij}(t) \triangleq$ the travel time from node i to node j given that the travel started at time t from node i



(a) Step function

(b) Smoothed function (Fleischmann et al., 2004)

Figure 2.5: Travel time function

A parameter δ_{ijk} is introduced for each breakpoint z_k, where $k \in \{1, 2, \ldots, K\}$, between two consecutive time slots Z_{k-1} and Z_k. The values of $\delta_{ij0} = \delta_{ijK} = 0$. The jump between two consecutive travel times segments Z_{k-1} and Z_k is linearized in the interval $[z_k - \delta_{ijk}, z_k + \delta_{ijk}]$ provided the parameter δ_{ijk} and determining the slope

$$s_{ijk} = \frac{\tau_{ij,k+1} - \tau_{ijk}}{2\delta_{ijk}} \tag{2.49}$$

The travel time function, as illustrated by Figure 2.5(b), is expressed as

$$\tau_{ij}(t) = \begin{cases} \tau_{ijk} & \text{for} \quad z_{k-1} + \delta_{ij,k-1} \leq t \leq z_k - \delta_{ijk} \\ \tau_{ijk} + (t - z_k + \delta_{ijk})\, s_{ijk} & \text{for} \quad z_k - \delta_{ijk} < t < z_k + \delta_{ijk} \end{cases} \tag{2.50}$$

The travel time function holds for all $k \in \{1, 2, \ldots, K\}$. Fleischmann et al. (2004) prove that if $\delta_{ijk} > 0$ for all intermediate breakpoints and the slope $s_{ijk} > -1$, that the arrival time function

$$A_{ij}(t) = t + \tau_{ij}(t) \tag{2.51}$$

is continuous and monotonic[1], i.e. adheres to the non-passing property. The papers by Ichoua et al. (2003) and Potvin et al. (2006) also refer to the non-passing property as the First-In-First-Out (FIFO) property. As $A_{ij}(\cdot)$ is a strictly increasing function, it possesses

[1]There is a designated sequence such that successive members are either consistently increasing or decreasing with no oscillation in relative value, i.e. each member of a monotone increasing sequence is greater than or equal to the preceding member; each member of a monotone decreasing sequence is less than or equal to the preceding member.

the inverse function $A_{ij}^{-1}(\cdot)$. $A_{ij}^{-1}(x)$ is interpreted as the departure time at node i so that node j can be reached at time x. Let $(i_0, i_1, i_2, \ldots, i_m, i_0)$ denote a partially constructed feasible route with m customer nodes where i_0 denotes the depot. The partial route could be simplified for illustration purposes to $(0, 1, 2, \ldots, m, 0)$.

In the presence of the non-passing property, the effective latest service start time at node i on the partial feasible route, denoted by d_i, could then be given by the backward recursive relation given in (2.52).

$$d_i = \begin{cases} \min\left\{l_i, A_{i0}^{-1}(l_0)\right\} & \text{for} \quad i = m \\ \min\left\{l_i, A_{i,i+1}^{-1}(d_{i+1})\right\} & \text{for} \quad 0 \leq i \leq m-1 \end{cases} \tag{2.52}$$

The actual service start time for each node i can be determined by the forward recursion given in (2.53).

$$t_i = \begin{cases} \max\left\{e_i, A_{01}(t_0)\right\} & \text{for} \quad i = 1 \\ \max\left\{e_i, A_{i-1,i}(t_{i-1})\right\} & \text{for} \quad 2 \leq i \leq m \end{cases} \tag{2.53}$$

The computation of both d_i and t_i is fairly elementary. The advantage is only apparent when route improvements are made, and subsequent feasibility check routines are eased.

The formulation used in this thesis refers to both travel and service times as *uncertain* and dependent on the realization of uncertain events. A principle distinction, however, is made between *stochastic* service times and *time-dependent* travel times. The implications of such a distinction will become apparent in the calculations and feasibility checks when solution algorithms are developed in later chapters, as only time-dependent travel time is considered. In the majority of applications, demand is assumed to be known at the time of establishing the actual route.

2.2.5 Multiple scheduling

It is often not viable to assume that each vehicle will only complete a single route. *Multiple scheduling* is concerned with the case where a vehicle could complete deliveries on a scheduled route, return to the depot where its capacity is renewed, after which a second, or consecutive trip is executed with the renewed capacity. Taillard et al. (1996) refer to this type of problem as the Vehicle Routing Problem with Multiple use of vehicles (VRPM). Butt and Ryan (1999) consider the Multiple Tour Maximum Collection Problem (MTMCP) and assumes that the routes are constrained in such a way that all of the customers cannot be visited. Their approach aims to maximize the number of customers serviced. Brandão and Mercer (1997) introduce the Multi-Trip Vehicle Routing Problem (MTVRP) and address the combination of multiple trips with time windows. The special case of multiple scheduling where only trips are considered is referred to as *Double Scheduling*.

This thesis considers a vehicle that starts and ends its tour at the depot. A *tour* consists of one or more *routes*, each starting and ending at the depot. The same vehicle can only be used for two or more routes if the routes do not overlap. As opposed to (2.28) multiple routes require a service time to be specified for the depot. Consider the example illustrated in Figure 2.6. The depot has a time window from 06:00 to 18:00. A vehicle fills its capacity

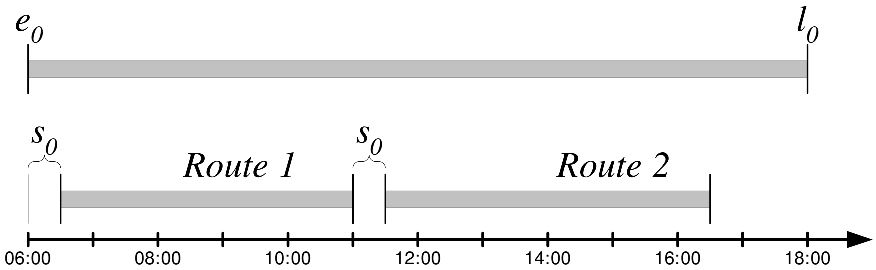


Figure 2.6: Double scheduling

at the depot for a time period of $s_0 = 0.5$ hours. It leaves the depot at 06:30, services the first route, and returns to the depot at 11:00, where its capacity is renewed. A second route, of five hours, is serviced before the vehicle returns to the depot.

Taillard et al. (1996) state that the multiple scheduling type of problem has received very little attention in literature. This thesis proposes a way to deal with multiple routes. The proposed solution involves a time verification process. If a vehicle arrives back at the depot at time a_m, and the service time is specified as s_0, then the vehicle is considered for an additional route on its current tour if, after the capacity has been renewed, the depot's time window is still open. The case is presented in (2.54).

$$a_m + s_0 \leq l_0 \tag{2.54}$$

The mathematical formulation of the VRPM requires a redefinition of the decision variables, as well as the constraints. The VRPM is addressed in the next section where the complete problem is defined and formulated.

2.3 The integrated problem at hand

An extended variant of the VRP, where multiple soft time windows, a heterogeneous fleet, and multiple scheduling are considered in an environment with uncertain travel and service times, is presented. Due to the complexity associated when concatenating elements from various variant acronyms, we revert to using a simple reference, the Thesis Problem (TP). To formulate the complex problem, we will redefine some of the variables and parameters used earlier, and introduce a few additional variables. We define the following basic parameters.

$N \triangleq$ total number of customers to be serviced

$q_i \triangleq$ deterministic demand for customer i, where $i = \{1, 2, \ldots, N\}$

$K \triangleq$ total number of vehicles available

$$z_i^k \triangleq \begin{cases} 1 & \text{if node } i \text{ is visited by vehicle } k\text{, where } i = \{1, \ldots, N\}, k = \{1, \ldots, K\} \\ 0 & \text{otherwise} \end{cases}$$

$\tilde{\xi} \triangleq$ a vector of uncertain variables corresponding to travel and service times. Each realization γ of $\tilde{\xi}$, denoted by ξ^γ, is referred to as a *state of the world* (Kall and Wallace, 1994)

$\Xi \triangleq$ the finite support of $\tilde{\xi}$ such that $\Xi = \left\{1, 2, \ldots, \xi^\gamma, \ldots, \xi^\Gamma\right\}$ where Γ is the total number of states in the problem *world*

$t_{ij}^k\left(\tilde{\xi}\right) \triangleq$ the travel time from node i to j with vehicle k, where $i, j = \{1, \ldots, N\}, k = \{1, \ldots, K\}$ expressed as a function of the realization of $\tilde{\xi}$

$\tau_i^k\left(\tilde{\xi}\right) \triangleq$ the service time at node i with vehicle k, where $i = \{1, \ldots, N\}, k = \{1, \ldots, K\}$, expressed as function of the realization of $\tilde{\xi}$

To expand the formulation and to include a heterogeneous fleet, we let:

$T \triangleq$ number of different types of vehicles available

$c_{ijt} \triangleq$ travel cost if a vehicle of type t travels from customer i to customer j, where $t = \{1, 2, \ldots, T\}$, and $i, j = \{0, 1, 2, \ldots, N\}$

$p_t \triangleq$ capacity of a vehicle of type t, where $t = \{1, 2, \ldots, T\}$

$f_t \triangleq$ fixed cost of a vehicle of type t, where $t = \{1, 2, \ldots, T\}$

$$\phi_t^k \triangleq \begin{cases} 1 & \text{if vehicle } k \text{ is of type } t\text{, where } k = \{1, 2, \ldots, K\}\text{, and} \\ & t = \{1, 2, \ldots, T\} \\ 0 & \text{otherwise} \end{cases}$$

Multiple soft windows will be addressed by introducing the following parameters:

$A_i \triangleq$ number of time windows for customer i, where $i = \{0, 1, 2, \ldots, N\}$

$a_i\left(\tilde{\xi}\right) \triangleq$ the actual arrival time at customer i, where $i = \{0, 1, 2, \ldots, N\}$, expressed as a function of the realization of $\tilde{\xi}$

$e_i^a \triangleq$ earliest allowed arrival time for customer i's a^{th} time window, where $i = \{0, 1, 2, \ldots, N\}$ and $a = \{1, 2, \ldots, A_i\}$

$l_i^a \triangleq$ latest allowed arrival time for customer i's a^{th} time window, where $i = \{0, 1, 2, \ldots, N\}$ and $a = \{1, 2, \ldots, A_i\}$

$L_i^{\max} \triangleq$ maximum lateness allowed by customer i, where $i = \{0, 1, 2, \ldots, N\}$

$\alpha_i \triangleq$ lateness penalty at customer i in cost per time unit, where $i = \{0, 1, 2, \ldots, N\}$

$\lambda_i\left(\tilde{\xi}\right) \triangleq$ actual lateness at customer i, where $i = \{0, 1, 2, \ldots, N\}$, expressed as a function of the realization of $\tilde{\xi}$

$w_i\left(\tilde{\xi}\right) \triangleq$ waiting time at customer i, where $i = \{0, 1, 2, \ldots, N\}$, expressed as a function of the realization of $\tilde{\xi}$

To ensure that multiple scheduling is considered, we let:

$R^k \triangleq$ number of routes scheduled for vehicle k, where $k = \{1, 2, \ldots, K\}$

$Q \triangleq$ maximum number for routes allowed for any one vehicle

$M^k \triangleq$ maximum tour time (all routes) allowed for vehicle k, where $k = \{1, 2, \ldots, K\}$

$d^{kr}\left(\tilde{\xi}\right) \triangleq$ vehicle k's departure time from the depot as it embarks on servicing its r^{th} route, where $k = \{1, 2, \ldots, K\}$ and $r = \{1, 2, \ldots, R_k\}$, expressed as a function of the realization of $\tilde{\xi}$

$g^{kr}\left(\tilde{\xi}\right) \triangleq$ vehicle k's return time at the depot after servicing its r^{th} route, where $k = \{1, 2, \ldots, K\}$ and $r = \{1, 2, \ldots, R^k\}$, expressed as a function of the realization of $\tilde{\xi}$

$\delta^k\left(\tilde{\xi}\right) \triangleq$ the amount by which vehicle k exceed its allowable tour time, where $k = \{1, 2, \ldots, K\}$, expressed as a function of the realization of $\tilde{\xi}$

$\beta^k \triangleq$ the positive unit penalty cost for vehicle k when exceeding its allowable tour time, where $k = \{1, \ldots, K\}$

With the notation established the decision variables for the TP are defined as:

$$x_{ij}^{kr} \triangleq \begin{cases} 1 & \text{if vehicle } k \text{ travels from customer } i \text{ to customer } j \text{ on its } r^{\text{th}} \text{ route,} \\ & \text{where } i,j = \{1,2,\ldots,N\}, k = \{1,2,\ldots,K\}, r = \{1,2,\ldots,R_k\} \\ 0 & \text{otherwise} \end{cases}$$

$$\psi_i^a \triangleq \begin{cases} 1 & \text{if the } a^{\text{th}} \text{ time window of customer } i \text{ is used, where } i \in \{1,2,\ldots,N\}, \\ & a \in \{1,2,\ldots,A\} \\ 0 & \text{otherwise.} \end{cases}$$

The mathematical formulation of the TP is provided.

$$\min z = \sum_{i=0}^{N} \sum_{\substack{j=0 \\ j\neq i}}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} \sum_{r=1}^{R^k} c_{ijt} x_{ij}^{kr} \phi_t^k + \sum_{j=1}^{N} \sum_{k=1}^{K} \sum_{r=1}^{R^k} \frac{f^k x_{0j}^{kr}}{R^k}$$
$$+ E_{\tilde{\xi}} \left[\sum_{i=1}^{N} \alpha_i \lambda_i \left(\tilde{\xi}\right) + \sum_{k=1}^{K} \beta^k \delta^k \left(\tilde{\xi}\right) \right] \tag{2.55}$$

subject to

$$\sum_{j=1}^{N} \sum_{r=1}^{Q} x_{0j}^{kr} = R^k \qquad \forall k \in \{1,2,\ldots,K\} \tag{2.56}$$

$$\sum_{j=1}^{N} \sum_{r=1}^{Q} x_{j0}^{kr} = R^k \qquad \forall k \in \{1,2,\ldots,K\} \tag{2.57}$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{N} \sum_{k=1}^{K} \sum_{r=1}^{R^k} x_{ij}^{kr} = 1 \qquad \forall j \in \{1,2,\ldots,N\} \tag{2.58}$$

$$\sum_{\substack{j=1\\j\neq i}}^{N}\sum_{k=1}^{K}\sum_{r=1}^{R^k} x_{ij}^{kr} = 1 \qquad \forall i \in \{1,2,\ldots,N\} \tag{2.59}$$

$$\sum_{q=1}^{N} q_i \sum_{\substack{j=0\\j\neq i}}^{N} x_{ij}^{kr} \leq p^k \qquad \forall k \in \{1,2,\ldots,K\}, \quad r = \{1,2,\ldots,R_k\} \tag{2.60}$$

$$e_i^a - (1-\psi_i^a)\,M \leq a_i\left(\tilde{\xi}\right) + w_i\left(\tilde{\xi}\right) \qquad \forall i \in \{1,2,\ldots,N\}, \quad \forall a \in \{1,2,\ldots,A_i\} \tag{2.61}$$

$$L_i^{\max} + (1-\psi_i^a)\,M \geq a_i\left(\tilde{\xi}\right) + w_i\left(\tilde{\xi}\right) \qquad \forall i \in \{1,2,\ldots,N\}, \quad \forall a \in \{1,2,\ldots,A_i\} \tag{2.62}$$

$$\sum_{a=1}^{A_i} \psi_i^a = 1 \qquad \forall i \in \{1,2,\ldots,N\} \tag{2.63}$$

$$\max\left\{0, e_j - \left(d^{kr}\left(\tilde{\xi}\right) + t_{0j}\right)\sum_{k=1}^{K}\sum_{r=1}^{R_k} x_{0j}^{kr}\right\} = w_j\left(\tilde{\xi}\right) \qquad \forall j \in \{1,2,\ldots,N\} \tag{2.64}$$

$$\max\left\{0, \left(a_i\left(\tilde{\xi}\right) - l_i^a\right)\right\} = \lambda_i^a\left(\tilde{\xi}\right) \qquad \forall i \in \{1,2,\ldots,N\}, \quad \forall a \in \{1,2,\ldots,A_i\} \tag{2.65}$$

$$d^{k1} \geq e_0 + s_0 \qquad \forall k \in \{1,2,\ldots,K\} \tag{2.66}$$

$$\sum_{k=1}^{K}\sum_{r=1}^{R^k} x_{0j}^{kr}\left(d^{kr}\left(\tilde{\xi}\right) + t_{0j}\right) \leq a_j\left(\tilde{\xi}\right) \qquad \forall j \in \{1,2,\ldots,N\} \tag{2.67}$$

$$\sum_{\substack{i=1\\i\neq j}}^{N}\sum_{k=1}^{K}\sum_{r=1}^{R^k} x_{ij}^{kr}\left(a_i\left(\tilde{\xi}\right) + w_i\left(\tilde{\xi}\right) + \tau_i^k\left(\tilde{\xi}\right) + t_{ij}^k\left(\tilde{\xi}\right)\right) \leq a_j\left(\tilde{\xi}\right) \qquad \forall j \in \{1,2,\ldots,N\} \tag{2.68}$$

$$\sum_{i=1}^{N} x_{i0}^{kr}\left(a_i\left(\tilde{\xi}\right) + \tau_i^k\left(\tilde{\xi}\right) + w_i\left(\tilde{\xi}\right) + t_{i0}\right) \leq g^{kr}\left(\tilde{\xi}\right) \qquad \forall k \in \{1,2,\ldots,K\}, \quad r \in \{1,2,\ldots,R^k\} \tag{2.69}$$

$$g^{k,r-1}\left(\tilde{\xi}\right) + s_0 = d^{kr}\left(\tilde{\xi}\right) \qquad \forall k \in \{1,2,\ldots,K\}, \quad r \in \{2,3,\ldots,R_k\} \tag{2.70}$$

$$g^{kr}\left(\tilde{\xi}\right) + s_0 \leq l_0 \qquad \forall k \in \{1,2,\ldots,K\}, \; r \in \{2,3,\ldots,R_{k-1}\} \tag{2.71}$$

$$g^{kR^k}\left(\tilde{\xi}\right) \leq M_k + \delta^k\left(\tilde{\xi}\right) \qquad \forall k \in \{1,2,\ldots,K\} \tag{2.72}$$

$$R^k \leq Q \qquad \forall k \in \{1,2,\ldots,K\} \tag{2.73}$$

$$\sum_{r=R^{k+1}}^{Q} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} x_{ij}^{kr} = 0 \qquad \forall k \in \{1,2,\ldots,K\} \tag{2.74}$$

$$x_{ij}^{kr} \in \{0,1\} \qquad \forall i,j \in \{1,2,\ldots,N\}, \; k \in \{1,2,\ldots,K\}, \; r \in \{1,2,\ldots,R^k\} \tag{2.75}$$

$$\psi_i^a \in \{0,1\} \qquad \forall i \in \{1,2,\ldots,N\}, \; \forall a \in \{1,2,\ldots,A_i\} \tag{2.76}$$

The objective function in (2.55) minimizes a combination of deterministic and stochastic cost components. The first expression represents the total variable traveling cost, followed by the total fixed fleet cost. The third expression represents the expected lateness penalties and constitutes firstly the lateness at each customer, and secondly the lateness for each vehicle.

The combination of (2.56) and (2.57) calculates the total number of routes and ensures that the same number of routes that starts for each vehicle, also finishes. Each customer is visited exactly once according to the constraint combination (2.58) and (2.59). Vehicular capacity is enforced through (2.60) by ensuring that the sum of the demands of all customers assigned to a specific route of a given vehicle do not exceed the vehicle's capacity, which may either by represented as weight or volumetric capacity, or both if additional constraints are added.

Constraints (2.61) and (2.62) ensure that the multiple soft time windows are adhered to where the parameter M represents a sufficiently large number, as discussed when multiple soft time windows were introduced. Actual arrival times and waiting times at any given customer is a function of the stochastic travel and service times of all customers preceding that specific customer, hence the stochastic notation. As each customer is visited only once, (2.63) ensures that only one time window for each customer is considered. The waiting time and lateness at each customer, both expressed as a stochastic variable, are determined in (2.64) and (2.65), respectively.

The departure time for each vehicle's first route is determined by (2.66), while the actual arrival time at the first customer on each route is determined by (2.67). Arrival times for subsequent customers are determined by (2.68).

The return time for each route is determined by (2.69). Consecutive route start times is determined by (2.70) by taking the service time of the depot into account where vehicles' capacities are renewed as proposed in (2.54). Constraint (2.71) enforces all routes to finish within the operating hours of the depot, while (2.72) determines the lateness for each vehicle when exceeding its allowed tour time. Each vehicle may not execute more than a predetermined number of routes as provided for in (2.73). Should it be determined in equations (2.56) and (2.57) that the required number of routes is less than the preset limit Q, then all *allowed* routes not required are eliminated through the introduction of (2.74). Binary decision variables are provided for with the introduction of (2.75) and (2.76).

2.4 Conclusion

This chapter deals with the background of the VRP, as well as the integration of multiple variants into a single problem instance — each contributing to the already complex nature of the problem. Although the model formulation is the first step in describing the problem comprehensively, only very small instances of the problem is currently solvable to optimality.

The following chapter introduces the complexity of the problem at hand, and reviews solution approaches for solving the problem. Exact, heuristic, as well as metaheuristic solution algorithms are considered.

Chapter 3

Intelligence in solution algorithms

Once a model of the perceived reality is formulated, a process is required to obtain a solution to the model which, in turn, could be implemented to solve the problem in reality. It should always be noted that *models* are solved, not *real problems*. Rardin (1998) refers to *numerical search* as the process of systematically trying different choices for decision variables so that the best feasible solution could be found. This chapter is dedicated to review the three primary search strategies used to solve mathematical programming models.

The first of these are exact solution algorithms where one can prove that the *best* feasible solution found is in fact the global optimum for the problem. The first section of the chapter introduces some of the fundamental exact solution algorithms, with reference to further review articles for interested readers.

Exact solution algorithms are unfortunately not always viable when the size of a problem increases. To compensate for the time-consuming computational burden, solution seekers opt for *approximate* solutions, also referred to as *heuristics*, where the *best* solution may, or may not, be the true optimum for the problem. Yet, heuristics offer solutions that are often better than the typical industrial solutions obtained through intuition and common sense. The second section introduces a number of heuristics dating back from the 1950's, and follows a few variations of these heuristics.

Heuristics have evolved during the 1990's to what is referred to as *metaheuristics* — intelligent strategies governing the execution of various heuristics in order to find even better solutions. The third section introduces a number of metaheuristics and its variations, from where a conclusion is drawn and a motivation is provided for the choice of solution algorithms for this thesis.

3.1 Exact solution algorithms

It might at first seem counterintuitive that integer (discrete) linear and combinatorial problems are more difficult to solve than their continuous counterparts, seeing that the algebra for Linear Programming Problem (LP) algorithms can be quite daunting. A discrete model with a finite number of possible decision variable values, on the other hand, seems much easier. This reasoning holds only for small instances of discrete problems, and Rardin (1998) confirms that total enumeration of all possible combinations is the most effective method to find the best solution. Consider a problem with only two binary variables, x_1 and x_2. There are only $2^2 = 4$ possible cases. Although ten binary variables will require $2^{10} = 1024$ cases to be enumerated, it is still viable using computers. The exponential growth in the number of case evaluations when enumerating requires alternative algorithms for problems of practical size.

This thesis follows the classification proposed by Laporte and Nobert (1987) and Laporte (1992) whereby exact algorithms are grouped into three primary categories, each covered in the following subsections.

3.1.1 Direct tree search methods

The analogy of a tree in search methods represents the primary stem being some initial solution, from where the stem is split into branches, or secondary stems that are related to the primary stem. These secondary stems, in turn, branch into tertiary stems, etc.

The first step in direct tree search methods is to find the primary, or initial solution. Because discrete optimization models are typically hard to solve, it is natural to find related, yet easier formulations of the problem. Auxiliary models are referred to as *relaxations* of the original discrete problem and are *easier* to solve as some of the constraints, or objective function(s) of the discrete problem are weakened. Solving the relaxations can lead the modeler to make solution interpretations of the original problems. Various relaxation techniques vary in *strength*. Rardin (1998) defines a relaxation as *strong* or *sharp* if the relaxation's optimal value closely bounds that of the original model, and the relaxation's solution closely approximates an optimum in the original problem. Various relaxation methods exist in introductory Operations Research textbooks and include LP relaxations, stronger Big-M constants and the introduction of valid inequalities (for example the *Cutting Plane* algorithm (Jeroslow, 1979)) (Hillier and Lieberman, 2005; Rardin, 1998; Taha, 2003; Winston and Venkataramanan, 2003). An even stronger relaxation, referred to as *Lagrangian* *relaxation*, do not weaken integrality constraints, but rather relax some of the main linear constraints after which they are dualised (or weighed) in the objective function with Lagrangian multipliers (Fisher and Jaikumar, 1981). Desrosiers et al. (1988) use Lagrangian relaxation methods to solve a variant of the MTSP with time windows.

Once a relaxation is solved to optimality, and the solution also conforms to all constraints in the original problem, the solution *is also* the optimal solution for the original problem. If not, various strategies and algorithms are employed to systematically work towards a relaxation of which the optimal solution is also optimal for the original problem.

The *branch-and-bound* search algorithms combine relaxations with an enumeration strategy to find optimal candidate solutions, while bounding the search by previous solutions. Laporte et al. (1989) adapt the branch-and-bound algorithm in solving a class of stochastic location routing problems with networks of 20 and 30 nodes. Laporte et al. (1986) solve the asymmetrical Capacitated Vehicle Routing Problem (CVRP) for 260 nodes. The structure of the VRP and its relationship with one of its relaxations, the MTSP, is exploited by Laporte (1992) in a similar manner.

The branch-and-bound algorithm has been modified with the introduction of stronger relaxations prior to the branching of a partial solution. The modified algorithm is referred to as *branch-and-cut* as the stronger relaxations are obtained with the inclusion of new inequalities. The inequalities should hold for all feasible solutions of the original discrete problem, but should render the last relaxation's optimum as infeasible, hence the term *cut*. Padberg and Rinaldi (1987) illustrate the generation of cuts in a symmetrical TSP with 532 nodes. Laporte et al. (1992) describe a general branch and cut algorithm for the VRP with stochastic travel times. The authors introduce cuts in the form of subtour elimination constraints, and introduce lower bounds on penalties if a route exceeds its predetermined route duration limit.

Van Slyke and Wets (1969) introduce the L-shaped method as a variant of the cutting plane algorithm for specific linear programs. Birge and Louveaux (1988) acknowledge that the method holds opportunity for stochastic programming applications and modify the method to create multiple cuts in each major iteration. Laporte and Louveaux (1993) further expand the method and refer to their general branch-and-cut procedure as the *integer L-shaped method* and apply it to stochastic integer programs and note that fathoming rules are different than in branch-and-bound trees. In another variation on branching, Christofides et al. (1979) propose a depth-first tree search in which single feasible routes are generated as and when required in their VRP formulation based on the TSP.

In their recent review of exact algorithms based on branch-and-bound, Toth and Vigo (2002a) state that these types of algorithms remain the state of the art with respect to exact solutions, especially in the case where asymmetric cost matrices exist. Giaglis et al. (2004) confirm that exact approaches are applicable to problems of practical size only if they have low complexity.

3.1.2 Dynamic Programming (DP)

DP determines the optimum solutions of an n-variable problem by decomposing it into n stages each consisting of a single-variable subproblem (Taha, 2003). The objective is to *divide-and-conquer* real-life problems by enumerating in an intelligent way through a state space of solutions (Brucker, 2004). In solving shortest path problems, Rardin (1998) claims that DP methods exploit the fact that it is sometimes easiest to solve one optimization problem by taking on an entire family of shortest path models. DP was first proposed for solving VRPs by Eilon et al. (as cited by Laporte (1992)).

Hamacher et al. (2000) faced the requirement that the nodes to be routed in a tour must be chosen from a small region of the map, and motivate their choice by the fact that the truck drivers have a local knowledge of the environment and is subjected to business constraints. Although the most natural DP formulation results in a DP with infinite state and action spaces, an optimality-invariance condition recently introduced by Lee et al. (2006) establishes leads to an equivalent problem with finite state and action spaces. Their formulation leads to a new exact algorithm for solving the Multi-Vehicle Routing Problem with Split Pick-ups (MVRPSP), based on a shortest path search algorithm, which they claim to be conceptually simple and easy to implement.

Although in a different problem context, Beaulieu and Gamache (2006) present an enumeration algorithm based on DP for optimally solving the fleet management problem in underground mines. Their problem consists of routing and scheduling bidirectional vehicles on a haulage network composed of one-lane bidirectional road segments.

Li et al. (2005) integrate the machine scheduling problem with a delivery routing problem and formulate a DP recursion since there are a finite number of time points for the start time of a trip of the vehicle. They conclude, however, that the problem can be simplified by limiting deliveries to direct shipments, a situation that is inappropriate if there is a large number of customers and small shipments across a geographically dispersed network.

3.1.3 Integer linear programming

The set partitioning formulation is a method that starts by assuming that the totality of routes which a single vehicle can operate feasibly, can be generated (Christofides et al., 1979). Let A denote the set of nodes representing customers. Then if $S \subseteq A$ is the subset of nodes which can be feasibly supplied on a single route by a vehicle v_k, it is assumed that the total variable cost associated with the optimal routing of nodes in S can be calculated. This is not trivial if $|S|$ is large as it relates to a TSP (one vehicle with a single route). For each vehicle v_k a family S_k of all feasible single routes for that specific vehicle is generated. A matrix $G = [g_{ij}]$ is produced with row i representing customer x_i and M blocks of columns where the k^{th} block of columns corresponds to vehicle v_k and column j_k of the block corresponds to a feasible single route S_{j_k} of vehicle v_k. The VRP now becomes the problem of choosing at most one column from each block of G so that each row of G has an entry when

$$g_{ij_k} \triangleq \begin{cases} 1 & \text{if customer } x_i \text{ is an element of the single route } S_{j_k} \\ 0 & \text{otherwise} \end{cases}$$

Balinksi and Quandt (as cited by Laporte (1992)) were among the first to propose such a set partitioning formulation for VRPs. But combinatorial problems often result in extremely large arrays of possibilities too complex to be modeled concisely (Rardin, 1998). Column generation adopt a two-part strategy for such problems. It first enumerates a sequence of columns representing viable solutions to parts of the problem, often employing DP. Part two of the strategy solves a set partitioning model to select an optimal collection of these alternatives fulfilling all problem requirements. It results in a flexible and convenient approach employing a multitude of schemes to generate columns which are complex. In this approach it becomes possible to address constraints that are often difficult to model. It suffers, however, from the shortcoming that the number of columns in G can be enormous.

A variant of the VRPTW is solved by Desrochers et al. (1992) who admit that exact solution algorithms have lagged considerably behind the development of heuristics. Their algorithm attempts to use best of breed by solving various subproblems using a branch and bound scheme, DP, and column generation. The drawback remains that the set partitioning problem stops being competitive when a large number of customers are to be serviced on a single route, for example when demands are small in relation to the vehicle capacity. This results in the LP relaxation to become more dense — leading to possible degeneracy.

The MTMCP is closely related to both the TSP and the VRP with the major difference that it is not possible to service all nodes in the graph in the allocated time on a given set of tours. Hence the objective to maximize the earned reward of those nodes visited. In their paper Butt and Ryan (1999) combine column generation and constraint branching to achieve an optimal solution algorithm that solves problems with 100 nodes.

In more recent work Righini and Salani (2004) note that a trade-off remains between the time spent on the column generation, and the quality of the lower bound achieved, indicating that research into effective exact algorithms remain active. Choi and Tcha (2006) use a column generation approach in solving the HVRP with a maximum of 100 nodes in the test problems used. Column generation, however, is not easily adapted to the stochastic variant of a routing problem (Lambert et al., 1993).

3.2 A case for heuristics

Maffioli (1979) indicates that real life combinatorial problems have a number of *unpleasant features*: problems are usually dimensionally large; problems have integrated constraints; and problems can not always be decomposed or generalized to simpler subproblems. It is noteworthy that although researchers attempt to solve real-world problems, complex problems are already solved in industry where decision makers often settle for *good enough* solutions (Russell and Norvig, 2003).

3.2.1 Route construction

Savings-based heuristics

Christofides et al. (1979) indicate that the majority of heuristics are constructive in nature in the sense that at any given stage one or more incomplete routes exist. Incomplete routes are extended to consecutive stages until a final route exists. The construction of routes may be either *sequential* if one route is completed prior to another being started, or *parallel* where more than one incomplete route may exist at a particular stage. After routes are created, a number of local improvements may be initiated to refine a route.

The savings algorithm established by Clarke and Wright (1964) is without doubt the most widely known heuristic in VRPs and has formed the basis of a substantial number of heuristic variations. The Clarke-Wright algorithm remains a computationally efficient algorithm, and deserves attention (Lenstra and Rinnooy Kan, 1981). The algorithm is defined as follows:

Step 1 Calculate the savings for all pairs of customers i and j, denoted by s_{ij}, where both customers are serviced on one route, as opposed to customer i being serviced on a new dedicated route from the depot, using (3.1)

$$s_{ij} = c_{0i} - c_{ij} + c_{j0} \qquad \forall i, j \in \{1, 2, \ldots, N\} \tag{3.1}$$

where N is the total number of customers in the network, and c_{ij} denotes the cost of traveling from node i to node j, and where $i = j = 0$ represents the depot.

Step 2 Arrange the savings in descending order of magnitude.

Step 3 Starting from the top, use one of the following approaches:

Sequential approach

1. Find the first feasible link in the list which can be used to extend one of the two ends of the currently constructed route.
2. If the route cannot be expanded, or no route exist, choose the first feasible link in the list to start a new route.
3. Repeat (1) and (2) until no more links can be chosen.

Parallel approach

1. If making a given link results in a feasible route according to the constraints of the VRP, add the given link to the solution. If not, reject the link.
2. Try the next link in the list and repeat (1) until no more links can be chosen.

Step 4 The links form the solution to the VRP.

Christofides et al. (1979) suggest that in the parallel approach a maximum number of routes, M, be introduced to ensure that vehicle feasibility constraints are adhered to. Mole and Jameson (1976) motivate why a sequential approach yields more benefit and adapt the savings procedure to calculate the best insertion position on edge (i, j) of the partially constructed route C for customer u, denoted by $s(i, u^{\star}, j)$, using the expression in (3.2)

$$s(i, u^{\star}, j) = \min_{i,j \in C} \{s(i, u, j)\} \qquad \forall u \in \{1, 2, \ldots, N\} | u \ni C \tag{3.2}$$

where C is the subset of the N nodes already routed, with

$$s(i, u, j) = 2d_{0u} + (d_{ij} - d_{iu} - d_{uj}) \tag{3.3}$$

The criteria used to determine the best edge to insert a specific customer is referred to as the *insertion criteria*. Once the best edge for insertion has been identified for each customer, the customer with the highest saving will be selected and inserted in its best position. The criteria used to select the best customer is referred to as the *selection criteria*.

Although a number of schemes have been suggested by Christofides et al. (1979) to identify the first customer on a new route, termed the *seed customer*, such as customer with earliest time window deadline; unrouted customer furthest from depot; or customer with largest demand, this thesis will propose a new method for identifying seed customers in Chapter 4.

Nelson et al. (1985) review and test a number of data structures to employ when implementing the Clarke-Wright algorithm. The authors establish methods of choice for VRPs with given characteristics of the network topology.

The savings heuristic has since its inception been adapted in quite a number of research contributions. Golden et al. (1984) refer to the basic savings algorithm as Clarke-Wright (CW), and introduced minor changes through their Combined Savings (CS) algorithm. They proceeded to introduce both the Optimistic Opportunity Savings (OOS) and Realistic Opportunity Savings (ROS). The latter was extended to the ROS-γ that included variety into the algorithm. Solomon (1987) not only applied the savings technique in solving the VRPTW, but also established benchmark problems which have since been used extensively. Paessens (1988), Salhi and Rand (1993) and Tung and Pinnoi (2000) propose various adaptions to the savings heuristic and apply the algorithms to generate feasible routes prior to an improvement stage. In a banking application Lambert et al. (1993) use the savings algorithm on both a deterministic and stochastic variant of the VRPTW. Dullaert et al. (2001) continue the development and adapt the original criteria for sequential insertion, referred to as the Adapted Combined Savings (ACS), Adapted Optimistic Opportunity Savings (AOOS), and the Adapted Realistic Opportunity Savings (AROS).

Liu and Shen (1999b) challenge the prior research by stating that a parallel approach to route construction actually yields superior results, and use the savings algorithm in solving the VRPMVTTW.

Ong et al. (1997) introduce new selection criteria and use the sequential approach on a variant of the Multi Period Vehicle Routing Problem (MPVRP) with time windows, specific vehicle type constraints, multiple depots and stochastic demand constraints. Liu and Shen (1999a) considered the FSMVRPTW and introduced some modifications on the savings expressions with added route shape parameters.

The basic Clarke-Wright algorithm is adapted by Hill et al. (1988), Ahn and Shin (1991), Hill and Benton (1992) and Malandraki and Daskin (1992) to accommodate forward scheduling where time-dependent travel times are modeled. Fleischmann et al. (2004) test three saving algorithms on a time-dependent travel time variant of the VRPTW.

Sweep algorithm

A different approach was introduced by Gillett and Miller (1974). Their proposed algorithm divides the locations into a number of routes. The following notation is introduced to explain the algorithm. Let:

$$
\begin{aligned}
N &\triangleq \text{ number of locations including the depot (where the depot is always referred} \\
&\quad\;\; \text{to as location 1)} \\
q_i &\triangleq \text{ the demand at location } i, \text{ where } i \in \{2,3,\ldots,N\} \\
(x_i, y_i) &\triangleq \text{ rectangular coordinates of the } i^{\text{th}} \text{ location, where } i \in \{1,2,\ldots,N\} \\
C &\triangleq \text{ the capacity of each vehicle} \\
d_{ij} &\triangleq \text{ the distance between locations } i \text{ and } j, \text{ where } i,j \in \{1,2,\ldots,N\} \\
\angle_i &\triangleq \text{ the polar coordinate angle (measured from the depot) of the } i^{\text{th}} \text{ location,} \\
&\quad\;\; \text{where } i \in \{2,3,\ldots,N\} \\
r_i &\triangleq \text{ the radius from the depot to location } i, \text{ where } i \in \{2,3,\ldots,N\}
\end{aligned}
$$

The polar coordinate angle is calculated through (3.4).

$$\angle_i = \arctan\left[\frac{y_i - y_1}{x_i - x_1}\right] \tag{3.4}$$

This results in $-\pi < \angle_i < 0$ if $y_i - y_1 < 0$, and $0 \leq \angle_i \leq \pi$ if $y_i - y_1 \geq 0$. The locations are renumbered in ascending order according to the size of their polar coordinate angle such that

$$\angle_i < \angle_{i+1} \qquad\qquad \forall i \in \{2,3,\ldots,N-1\}$$

The forward sweep portion of the algorithm partitions locations into routes beginning with the location with the smallest angle. Locations are added until the vehicle's capacity is reached, or a preset distance constraint on a route is reached. Subsequent routes are generated in a similar manner until all locations are routed. Each route is then optimised using either exact or heuristic algorithms for the TSP. The minimum distance traveled is then the sum of the distances of each optimised route.

The x-y axis is then rotated counterclockwise so the first location becomes the last, the second becomes the first, the third the second, etc. The minimum distance is calculated again. The rotation of the x-y axis and the calculation of the distance traveled is repeated for all possible axis configurations. The minimum forward sweep distance is the least total distance traveled taken from all axis configurations that was calculated.

The backward sweep portion is similar to the forward portion except that it forms the routes in reverse order, i.e. it start with the last reordered entry based on the polar coordinate angle.

Gillett and Miller (1974) state that the two portions often produce different routes and minimum distances traveled, hence the sweep algorithm's result is the route with the lowest distance traveled of the two portions.

Generalized assignment

Where assignment problems involve the optimal pairing of objects of two distinct types, for example exactly one job order to exactly one machine, or exactly one customer to exactly one sales representative, the *generalized* assignment problem allows for each object i to be assigned to some j, and each j being allowed to receive a number of i (Rardin, 1998). Fisher and Jaikumar (1981) reformulate the VRP in a two-stage approach. First customers are assigned to vehicles, hence the relation to generalized assignment problems. Secondly, for each vehicle the customers assigned to that vehicle is sequenced using the TSP formulation or some other route construction algorithm. The approach is heuristic as the assignment problem's objective function is a linear approximation of the second stage's distance traveled. A number of methodological variants are provided in Nygard et al. (1988). Koskosidis et al. (1992) extend the approach to solve a time window variant of the routing problem.

Giant tours

In the VRP version, a giant tour, including the depot, is first created. A giant tour is a single tour that starts from the depot, passes through *all* customer sites and returns to the depot. A directed cost network is then constructed. Define the tour T_{ab} as a tour beginning with an arc from the depot to customer a, then following the giant tour between customers a and b (which might include other nodes), finishing with an arc from customer b to the depot. There exist a directed edge in the cost network from a to b if and only if the tour T_{ab} is feasible in terms of vehicle capacity and distance restriction. The length of the edge ab in the cost network is the length of T_{ab}. The shortest path problem is subsequently solved using Dijkstra's (1959) algorithm, providing a partitioning of the giant tour.

The procedure is repeated starting from different giant tours and the overall least cost solution is chosen. In their experiments Nagy and Salhi (2005) constructed 5 giant tours; one using the nearest neighbor, another using the least insertion cost rule, and the remaining three tours are generated randomly. A detailed description on how to generate these giant tours and how to construct the associated cost networks can be found in Salhi et al. (1992).

3.2.2 Route improvement

Numerical search is the process of systematically trying different values for the decision variables in an attempt to find a better solution. The process keeps track of the feasible solution with the best objective function value found thus far, referred to as the *incumbent* solution. Rardin (1998) states that most optimization procedures can be thought of as variations of a single theme: *improving search*. Synonyms of the theme include *local improvement*, *hill climbing*, *local search*, and *neighborhood search*.

An improving search heuristic for vehicle routing and scheduling usually starts with a feasible solution created through the route construction heuristics suggested in Section 3.2.1. A characteristic, and unfortunately a drawback of an improving search heuristic is that it advances along its search path of feasible solutions only while the objective function value improves. The search space in which new solutions are investigated is best explained through the analogy of a neighborhood: nearby points of the current solution, each within a small distance from the current solution.

Slight modifications to the current route are referred to as *perturbations*, and are accepted if they yield feasible solutions with an improved objective function value. Although the discussion in this section is by no means exhaustive, it introduces some of the basic mechanisms for creating perturbations. Authors such as Nagy and Salhi (2005) apply combinations of these perturbations sequentially to obtain improved solutions. For purposes of this discussion nodes will be denoted by a, b, c, etc., and routes by bolded characters $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, etc.

Route reversal

A procedure introduced by Nagy and Salhi (2005) in their Vehicle Routing Problem with Pickups and Deliveries (VRPPD). They observed that changing the direction of a route does not lead to an increase in the route length, and may lead to increased feasibility. In their application the objective is to minimize the infeasibilities when integrating both pickups and deliveries simultaneously, as opposed to sequentially.

2-Opt

A routine introduced by Lin (1965) based on interchanging two edges, say ab and cd, to form two new edges ac and bd.

3-Opt

A modification of the *2-Opt* routine. In this case three arcs are exchanged with three other edges.

Shifting node

Similar to the *3-Opt* routine involving two routes. A single node a is removed from a route $\boldsymbol{x}$ and inserted into another route $\boldsymbol{y}$.

Exchanging nodes

An extension of the *Shifting node* routine. A node a is identified on route $\boldsymbol{x}$, and node b on route $\boldsymbol{y}$. The two nodes a and b are exchanged in their respective positions.

λ-Interchange

When an equal number of nodes, λ, are exchanged between two routes, the perturbation is referred to as λ-Interchange (Tan et al., 2001c; Thompson and Psaraftis, 1993). The *Exchanging nodes* perturbation is therefor a special case where $\lambda = 1$.

Double shift

A more complex extension of the *Shifting node* routine where two nodes, a and b, and three routes, $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, are considered. Node a is removed from route $\boldsymbol{x}$ and inserted into route $\boldsymbol{y}$, while node b is removed from route $\boldsymbol{y}$ and inserted into route $\boldsymbol{z}$. This is different from performing the *Shifting* routine twice, as after the first *Shift* the resulting route may be infeasible. It should be noted that this routine is computationally more complex as the possible combinations to consider increases substantially.

Splitting a route

According to Mosheiov (as cited by Nagy and Salhi (2005)) a route can be improved if the depot is reinserted into the route, resulting in two routes being created from the original one route considered.

Combining routes

If feasible, two routes $\boldsymbol{x}$ and $\boldsymbol{y}$ are combined, considering both orders $\boldsymbol{xy}$ and $\boldsymbol{yx}$.

3.3 Metaheuristics

The improving search heuristics discussed in the previous section are applied until there are no solutions in the immediate neighborhood hat include a solution that is both feasible and improving. The incumbent solution is then referred to as a *local optimum*. The advantage of heuristics is that good feasible solutions can still be found even though optimality can not be guaranteed; the disadvantage is that uncertainty exists about how close the solutions actually came to the optimal. Herein lies the drawback of heuristics, as the initial solution may negatively influence the optimality of the local optimum found. Refer to the overly simplified illustration in Figure 3.1 and note that if the heuristic starts with a solution at

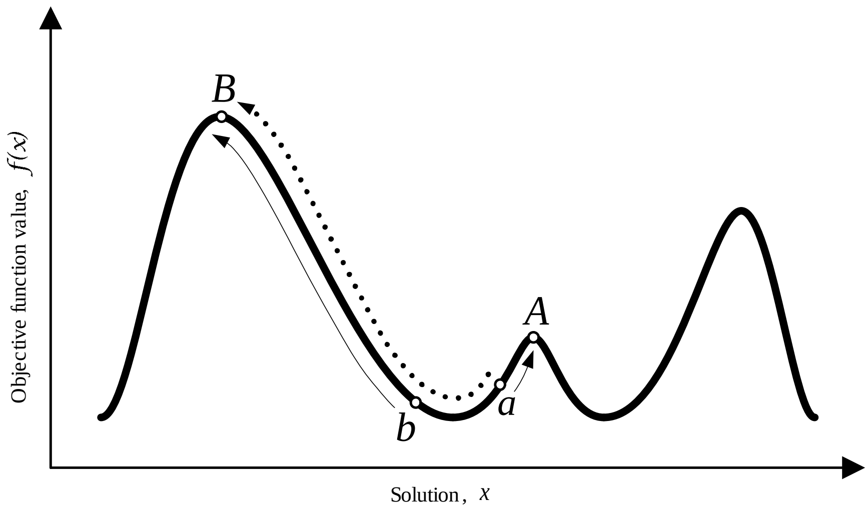


Figure 3.1: Local vs. global optimum

a, it can only improve until it reaches the local optimum at A. If the same heuristic starts with a solution at b it can reach the local optimum at B which also happens to be the *global optimum*: a feasible solution such that no other solution has a superior objective function value.

The interested reader is referred to the TOP Program (2006) research group within the *Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology* (SINTEF). The group has an extensive bibliography of research contributions in the field of vehicle routing, with the majority being on metaheuristics and future research opportunities.

Metaheuristics are master strategies which uses intelligent decision making techniques to guide algorithms to find global optimal solutions by temporarily allowing moves in the neighborhood that result in solutions with inferior objective function values compared to the incumbent solution. Such a deteriorating move is illustrated in Figure 3.1 with a solution starting at a, and deteriorating towards b along the dotted line before improving towards B. A problem arises with accepting temporarily deterioration moves. Consider a and b to be neighbor solutions. Each solution can thus be reached from the other with a single perturbation. A move from a to b may be accepted as a temporarily deteriorating move. However, a move from b to a will always be accepted as it improves the objective function. This may lead to indefinite cycling around a single solution. All metaheuristics follow a similar process, although their specific naming conventions, analogies and detailed routines may vary.

Initialization The process of finding an initial solution. Some metaheuristics, such as the *Genetic Algorithm* performs well with randomly generated initial solutions that need not be feasible, while the *Tabu Search* is highly sensitive to the quality of the initial solution.

Diversification The mechanism that ensures that the underlying heuristics are searching a diversified neighborhood, and thus not getting trapped within local optima.

Intensification A mechanism that ensures the heuristic starts *zooming in* towards a single solution. The most promising neighborhoods are identified and those areas of the solution space are searched more thoroughly.

The diversification and intensification can repeat indefinitely, and hence requires a stopping criteria to terminate the metaheuristic (Van Breedam, 2001). The longer the metaheuristic is run, the higher the probability of converging to the global optimum.

An elementary metaheuristic would entail running the improving search heuristic with multiple initial solutions, each yielding a single local optimum. The best of these local optima is the incumbent solution, denoted by $\hat{x}$. Such a process would be highly reliant on the choice of initial solutions and may yield inferior local optima if initial solutions are not selected and generated carefully.

Four promising metaheuristics are introduced, and interested readers can refer to Gendreau et al. (1998) for a general review of metaheuristics for the VRP.

3.3.1 Tabu Search (TS)

The TS is a memory-based local search metaheuristic introduced by Glover (1986) that searches the neighboring solution space (neighborhood) in search of an improving solution, updating the incumbent solution when improving moves are made. Deteriorating moves are allowed and the metaheuristic deals with cycling by declaring moves to recently visited solutions as *tabu*, hence the name. A thorough and recent review of the TS can be found in Bräysy and Gendreau (2001), where the authors focus on time window variants of the VRP.

A general TS approach is presented in Algorithm 3.1. An initial solution x^0 and a

---

Algorithm 3.1: Tabu Search

Input: Initial feasible solution x^0; Iteration limit $t^{\max}$

$t \leftarrow 0$

$\hat{x} \leftarrow x^t$

Clear Tabu-list, $T = \{\cdot\}$

Generate feasible move set M^{x^t}

while either $(\triangle x \in M$ **and** $\triangle x \ni T$ **and** $t < t^{\max})$ **or** $(\triangle x$ *satisfies* aspiration$)$ **do**

    $x^{t+1} \leftarrow x^t + \triangle x$

    $T \leftarrow T \cup \{x^{t+1}\}$

    if $c\left(x^{t+1}\right) < c\left(\hat{x}\right)$ **then**

        $\hat{x} \leftarrow x^{t+1}$

    endif

    $t \leftarrow t + 1$

    Generate feasible move set M^{x^t}

endw

---

stopping criteria is required. In this case the stopping criteria is determined to be a preset maximum iteration count $t^{\max}$. The algorithm is initialized by setting the iteration count to zero, setting the initial solution to be the incumbent solution, and clearing the tabu list. The objective function value $c(x)$ is expressed as a function of the solution x. A feasible move set M^{x^t} that represents the neighborhood around the current solution x^t is generated. The neighborhood is established through any of the perturbations discussed in Section 3.2.2. If either no non-tabu move $\triangle x \in M$ leads to a feasible neighbor of the current solution x^t within the preset iteration limit, or some aspiration criteria[1] is met, the metaheuristic

[1] The aspiration criteria may override the tabu list, or the iteration limit criteria.

terminates and the incumbent solution $\hat{x}$ is the approximate optimum. If not, the new neighbor becomes the current solution and is added to the tabu list. The current solution replaces the incumbent if it has a superior objective function value $c\left(x^{t+1}\right)$. The iteration count is incremented, a move set is generated for the new current solution, and the process is repeated.

In a comparison of heuristics and metaheuristics, Van Breedam (2001) identifies TS as a dominant improvement heuristic with the certainty of achieving at least a *local* optimum. Their observation is confirmed by Lee et al. (2006). Ichoua et al. (2003) implement the TS in both a static and dynamic setting, and claim that the model provides substantial improvements over a model based on fixed travel times. Recent developments include drastically reducing the size of the search neighborhood, so called *granular* neighborhoods (Toth and Vigo, 2003). Results obtained when using promising moves, as proposed by granular neighborhoods, yielded good solutions within short computing times.

3.3.2 Simulated Annealing (SA)

As opposed to a local search method, SA is a randomized search method (Brucker, 2004). To understand the concept of simulated annealing in optimization, one has to look at its analogy to the physical annealing system as first introduced by Kirkpatrick et al. (1983). The ground state of a solid, for example steel, is that state in which its atoms or particles are arranged into a minimum energy configuration – the most stable state of the solid. The ground state of a metal can be obtained through the process of physical annealing. The metal is first heated to a high temperature to induce its transformation from a solid to a liquid. This temperature is called the *melting point* of the metal. In its liquid state the metal is unstable, the particles move about freely, exhibiting high energy, since they are not arranged in any set configuration. The temperature is then carefully reduced to allow the particles to gradually settle into the arrangement of minimum energy and the ground state is obtained.

Similarly, SA is aimed at obtaining the minimum value of the objective function of an optimization problem which corresponds to the ground state of the solid (Tan et al., 2001c). Any other state of the solid corresponds to a feasible solution for the optimization problem, and the energy of a state of the solid is equivalent to the objective function value of a solution. A control parameter q, analogous to the temperature of the physical system, is used to control the gradual convergence of the SA algorithm towards the global optimum by regulating the acceptance of moves that deteriorates the objective function value. Similar to a small displacement of the atoms of the solid, the current solution at stage t, x^t, undergoes small perturbations $\triangle x$ as it converges towards the optimum solution.

The general SA metaheuristic provided in Algorithm 3.2 requires an initial solution x^0, and a stopping criteria. Alfa et al. (1991) indicate that the computational time required for finding good solutions are sensitive to the quality of initial solutions. As for the TS algorithm, an iteration limit count $t^{\max}$ is used. The algorithm also requires an initial temperature q^0 and a cooling parameter δ that reduces the temperature of the system after a sufficient number of iterations, denoted by $q^{\max}$.

Initializing the SA algorithm entails setting both the iteration count and the temperature control count to zero, setting the temperature to the initial temperature, and assigning the initial solution as the incumbent $\hat{x}$. The neighborhood is established through any of the perturbations discussed in Section 3.2.2.

If either the iteration count limit $t^{\max}$ is reached, or there are no more feasible moves $\triangle x$ in the neighborhood move set M^{x^t} for the current solution x^t, the algorithm terminates. Otherwise the move is tested for acceptance. If the move is improving the objective function value, it is accepted with a probability of 1. If the move is deteriorating, it will still be accepted with probability

$$P\left[accept\right] = e^{\left(\frac{c(\hat{x})-c(x')}{q}\right)}$$

Returning to the analogy between the physical annealing of a solid and the simulated annealing algorithm, the acceptance criterion for the SA algorithm is deducted from the Metropolis criterion. The Metropolis algorithm, as introduced by Metropolis et al. (as cited by Aarts and Korst (1989)) is a simple algorithm for simulating the physical annealing of a solid. It states that, given a current state i of the solid with energy E_i, a subsequent state j, with energy E_j is generated via a small displacement of the atoms of the solid. If the resulting energy difference, $E_j - E_i$, is less than or equal to zero, j is accepted as the new current state. If, however, the energy difference should be greater than zero, the state j will only be accepted with probability

$$P\left[accept\right] = e^{\left(\frac{E_i-E_j}{k_B T}\right)}$$

where T is the current absolute temperature of the solid and k_B is known as the physical *Boltzmann constant*. Kirkpatrick et al. (1983) noted that since the temperature is merely a control parameter, the Boltzmanns constant can be omitted. The control parameter is

Algorithm 3.2: Simulated Annealing

Input: Initial feasible solution x^0; Iteration limit $t^{\max}$

Input: Initial temperature $q^0 \gg 0$; Temperature limit $q^{\max}$; Cooling factor $0 < \delta \leq 1$

$t \leftarrow 0$

$q^{\text{count}} \leftarrow 0$

$q \leftarrow q^0$

$\hat{x} \leftarrow x^t$

Generate feasible move set M^{x^t}

while $\triangle x \in M^{x^t}$ **and** $t < t^{\max}$ **do**

  $x' \leftarrow x^t + \triangle x$

  if $q^{count} = q^{\max}$ **then**

    $q \leftarrow \delta q$

    $q^{\text{count}} \leftarrow 0$

  else

    $q^{\text{count}} \leftarrow q^{\text{count}} + 1$

  endif

  if either $c(x') < c(\hat{x})$ **or** $Probability\left(e^{\frac{c(\hat{x})-c(x')}{q}}\right)$ **then**

    $x^{t+1} \leftarrow x'$

    if $c\left(x^{t+1}\right) < c(\hat{x})$ **then**

      $\hat{x} \leftarrow x^{t+1}$

    endif

    else

      $x^{t+1} \leftarrow x^t$

    endif

  $t \leftarrow t + 1$

  Generate feasible move set M^{x^t}

endw

formulated so as to allow virtually all deteriorating moves during the initial stages of the algorithm. As the control parameter is gradually decreased, the probability of accepting deteriorating moves also decreases, and the algorithm converges to the global optimum.

Robusté et al. (1990) indicate in their application of SA that a human can actually outperform the algorithm for large problems in terms of the quality of the solution. Development of the SA have since continued with Van Breedam (1995) reviewing and comparing variants of the SA. Tan et al. (2001c) attribute a number of advantages to the SA metaheuristic:

- Deals with arbitrary systems and cost functions.
- Statistically guarantees an optimal solution (provided sufficient processing time).
- Relatively easy to code, even for complex problems.
- Generally gives a good solution within reasonable processing time.

The latter point has been supported by Van Breedam (2001) stating that the difference in solution quality between TS and SA never exceeded 4% in his evaluation. In their comparative analysis of three metaheuristics, Tan et al. (2001c) conclude that SA is a good compromise between computational effort and quality of solution.

3.3.3 Genetic Algorithm (GA)

GAs were developed and published by John Holland in 1975. GAs are algorithms that search for global optimal solutions by intelligently exploiting random search methods, emulating biological evolution (Rardin, 1998). The relationships between genetic evolution and optimization are:

- *Populations* are represented by groups, each representing a feasible solution.
- In a population, parents mate according to *natural selection*. This is analogous to randomly selected feasible parent solutions.
- *Offspring* are produced by the mating of the selected parents and represent newly created solutions.
- In nature, offspring exhibit some characteristics of each parent since *chromosomes* are exchanged to form new chromosome strings. The algorithm draws on the analogy by creating two new offspring solutions using perturbations such as swapping, on parts of the parent solutions. In GAs the perturbations are often referred to as *crossovers*.

- *Survival of the fittest* is also incorporated as the fitness of a solution can be related to its objective function value. The fittest solutions will typically reproduce to ensure the survival of the fittest solution in the next generation.

- *Mutation* for diversity is represented in the metaheuristic by the random modification of chromosomes, i.e. possible solutions.

Goldberg (1989) reviews GA applications in search strategies an optimization. The general GA metaheuristic provided in Algorithm 3.3 indicates p unique feasible initial solutions

---

Algorithm 3.3: Genetic Algorithm

Input: Generation limit $t^{\max}$

Input: Population size p; Initial feasible solutions $x_1^0 \ldots x_p^0$

Input: Population subdivisions p_e, p_i, and p_c such that $p_e + p_i + p_c = p$

$t \leftarrow 0$

while $t < t^{\max}$ **do**

    begin elite

        Copy p_e best solutions from generation t to generation $t+1$

    end

    begin immigrant

        Include p_i new solutions in generation $t+1$

    end

    begin crossover

        Choose $\frac{p_c}{2}$ non-overlapping pairs of solutions from generation t

        Perform crossover perturbations

        Include new solutions in generation $t+1$

    end

    $t \leftarrow t+1$

endw

$x^\star \leftarrow \min_{i \in \{1,\ldots,p\}} \{x_i^t\}$

$\hat{x} \leftarrow$ locally optimized $x^\star$

---

required to constitute generation 0. Filipec et al. (1998) test their GA with various population sizes and conclude that too small a population may terminate the algorithm prematurely as diversification is compromised, while too large populations slows down the convergence rate as more generations are required (increased computational effort) to initiate dominance of quality solutions. Initial solutions are created either randomly or using route construction heuristics as discussed in Section 3.2.1 (Vas, 1999). Skrlec et al. (1997) and Tan et al. (2001c) suggest using only heuristics so as to improve the rate of convergence.

The algorithm only terminates when a sufficient number of generations have existed. Survival of the fittest is ensured as the p_e best solutions of generation t is *cloned* exactly into generation $t+1$. A number, p_i, of new *immigrant* solutions are generated and included in generation $t+1$. The balance of generation $t+1$ is made up by performing various crossover perturbations on a random selection of $\frac{p_c}{2}$ solutions from generation t.

Two distinct approaches are found in literature to solve constrained VRPs with GAs.

Cluster first, route second

This approach was popular in early writings. Thangiah et al. (1991) developed *GIDEON*, a GA program used to solve the VRPTW. At the time it was the best algorithm available for the VRPTW as it produced the best known solutions for 41 of the 56 benchmark problems introduced by Solomon (1987). *GIDEON* has two distinct modules:

Clustering This module assigns customers to specific vehicles in a process called *genetic clustering*. It uses a GA to sector customers into clusters, with each cluster serviced by one vehicle. Figure 3.2 shows the sweeping motion that is is used together with seed angles to create clusters. Each vehicles cluster is routed to minimize route cost, not taking into account vehicle capacities or time windows. The first customer per route, referred to as the *seed customer*, is randomly selected out of the cluster, the rest of the route is formed by determining which customer, when inserted in the route, will produce the lowest route cost, i.e. using a savings heuristic. The best set of clusters obtained by this module is transferred to the next module.

Local route optimization Customers are exchanged between clusters to ensure the feasibility of the solution — taking into account time windows and vehicle capacities. To change a customers cluster, its angle is artificially altered. When a cluster is changed, a cheapest insertion algorithm is used to improve the cluster route.

Nygard and Kadaba (1991), Thangiah and Gubbi (1993), Malmborg (1996), Filipec et al. (1997), Skrlec et al. (1997) and Karanta et al. (1999) were among the contributors using the cluster first, route second approach. Nygard and Kadaba (1991) found that GAs for VRPs tend not to perform well when customers are geographically clustered and a small fleet is

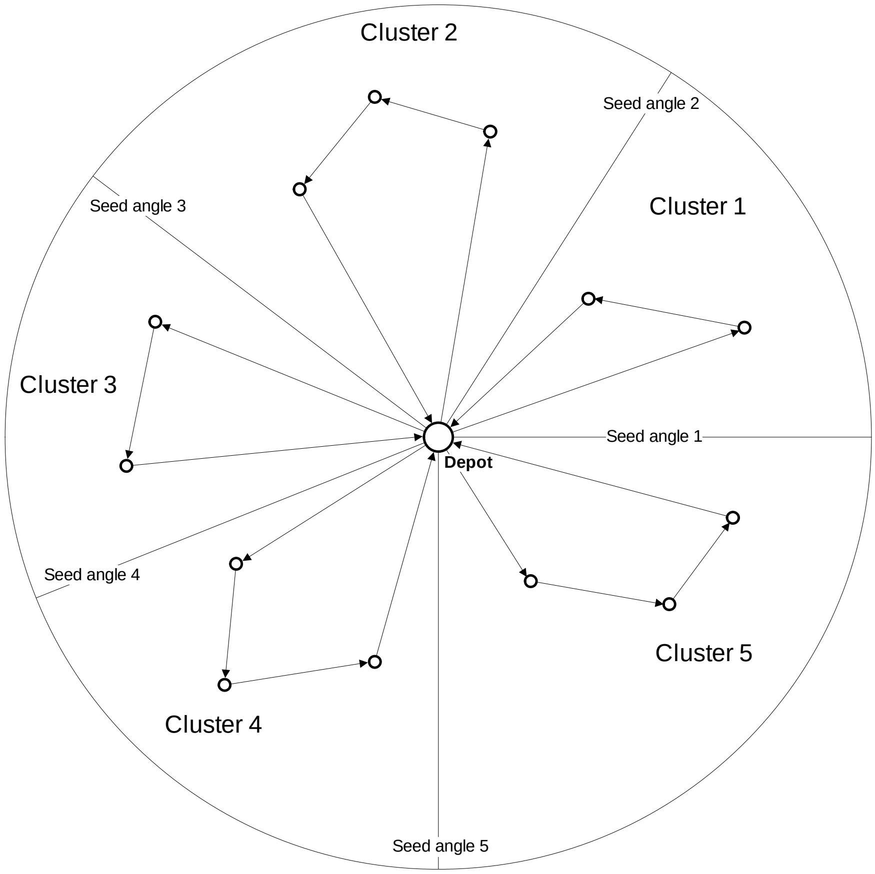


Figure 3.2: Division of customers using seed angles (Thangiah et al., 1991)

used. For all other problem instances the GA performs well. Tan et al. (2001c) claims that the approach *"is only a hybrid heuristic that constitutes some GA element"*.

Route first, cluster second

Recently, path representations are implemented more often for all VRP variations (Filipec et al., 1998; Hwang, 2002; Maeda et al., 1999; Ochi et al., 1998; Prins, 2004; Tan et al., 2001b,c; Zhu, 2003). To indicate separate routes in a chromosome, extra partitioning characters need to be inserted into the chromosome. These extra characters may render the GA useless. GAs use two phases to solve VRP variations, with each chromosome representing a specific path through all the customers. In the first *routing* phase, the *GA* improves the long chromosome string by solving a TSP for all customers. The second *clustering* phase creates a route for each vehicle out of the long route. This is done by another algorithm that adds customers to a vehicle only if time windows are not violated, until the vehicle is full. The following customer in the single route chromosome is then assigned to the next vehicle.

Tan et al. (2001c) were the first to compare three popular metaheuristics, TS, SA and GA for VRP variants. They conclude that GAs were successful in solving the VRPTW, but deduce that there is still no single metaheuristic generic enough to solve all routing problems.

3.3.4 Ant Colony Optimization (ACO)

ACO algorithms are classified as iterative, probabilistic metaheuristics for finding solutions to combinatorial optimization problems. ACO is a general term proposed by Dorigo and Stützle (2002) that includes all ant algorithms. The ant algorithm is an evolutionary approach where several generations of artificial ants search for good solutions. Every ant of a generation builds a solution in a step by step manner, going through several decisions. Ants that found good solution(s) mark their paths through the decision space by placing *pheromone* on the edges of the path. The ants of the next generation are attracted to pheromone and they are more likely to search the solution space near good solutions (Middendorf et al., 2002).

Ant algorithms are inspired by the foraging mechanism employed by real ants attempting to find a shortest path from their nests to food sources. A foraging ant will mark its path by distributing an amount of pheromone on the trail, thus encouraging, but not forcing, other foraging ants to follow the same path (Dorigo et al., 1999). Pheromone is the generic name for any endogenous chemical substance secreted by an organism to incite reaction in other organisms of the same specie. This principle of modifying the environment to induce a change in the ants' behavior via communication is known as *stigmergy*. The effect of stigmergy provides the basis for the ant foraging behavior and artificial ant metaheuristics. Dorigo et al. (1999) discuss the experiments conducted that suggest that the social structure of ant colonies can determine shortest paths between the nest and food sources. A formal proof, however, is absent.

There are a number of direct relationships between real foraging ants and artificial ants used in the ACO metaheuristic.

Colony of cooperating individuals Similar to real ants, artificial ants are composed of a population (or colony) of concurrent and asynchronous entities cooperating to find food timeously. The artificial food are good *solutions* to the optimization problem under consideration. Although the complexity of each artificial ant is such that it can build a feasible solution, high quality solutions are the result of the cooperation among the individuals of the whole colony. This is analogous to a real ant that can by chance find a path between the nest and the food. But only the cooperation of the whole colony can ensure that sufficient food sources are located as close as possible to the nest. Ants cooperate by means of the information they concurrently read and write on the problems states.

Pheromone trail and stigmergy Artificial ants modify some aspects of their environment as real ants do. While real ants deposit a chemical substance, pheromone, on the world state they visit, artificial ants change some numeric information locally stored in the problem state they visit. This information takes into account the ants current history or performance and can be read and written by any ant accessing the state. By analogy, this numeric information is called the artificial pheromone trail, *pheromone trail* for short. In ACO algorithms local pheromone trails are the only communication channels among the ants. This stigmergetic form of communication plays a major role in the utilization of collective knowledge. Its main effect is to change the way the environment (the problem landscape) is locally perceived by the ants as a function of all the past history of the whole ant colony.

Usually, in ACO algorithms an evaporation mechanism is employed, similar to real pheromone evaporation, that modifies pheromone information over time. Pheromone evaporation allows the ant colony slowly to forget its past history so that it can direct its search toward new directions without being over-constrained by past decisions, hence addressing the diversification issue raised for metaheuristics in general.

Shortest path searching and local moves Artificial and real ants share the common task of finding a shortest (minimum cost) path joining an origin (nest) and destination (food). Real ants systematically walk through adjacent terrains' states. Similarly, artificial ants move step-by-step through the neighborhood of solutions of the problem. The exact definitions of state and neighborhood are problem specific.

Stochastic and myopic state transition policy Artificial ants, as does real ants, build solutions applying a probabilistic decision policy to move through adjacent states. As for real ants, the artificial ants' decision policy makes use of local information only and it does not make use of lookahead to predict future states. Therefore, the applied policy is completely local, in space and time. The policy is a function of both the *a priori* information represented by the problem specifications (equivalent to the terrains structure for real ants), and of the local modifications in the environment (pheromone trails) induced by past ants.

Artificial ants also have some characteristics that do not have counterparts in real ants. Artificial ants live in a discrete world and their moves consist of transitions from discrete states to discrete states. Artificial ants have an internal state. This private state contains the memory of the ants' past actions. Artificial ants deposit an amount of pheromone that is a function of the quality of the solution found. Timing in pheromone laying is problem dependent and often does not reflect real ants behavior. For example, in many cases artificial ants update pheromone trails only after having generated a solution. To improve overall system efficiency, ACO algorithms can be enriched with extra capabilities such as the ability to forecast, local optimization, and backtracking that cannot be found in real ants.

An ant is a simple computational agent, which iteratively constructs a solution for the instance to solve. Partial problem solutions are seen as *states*. At the core of the ACO algorithm lies a loop, where at each iteration, each ant moves (performs a step) from a state i to another one j, corresponding to a more complete partial solution.

Algorithm 3.4 is based on Maniezzo et al. (2004) and requires an *a priori* desirability

Algorithm 3.4: Ant Colony Optimization

Input: Attractiveness η_{ij}; Trail level τ_{ij}

Input: Number of ants k

while $t < t^{\max}$ **do**
    for *each ant* k **do**
        repeat
            choose in probability the state j to move to
            append ant k's set $tabu_k$
        until *ant k's solution is complete*
    endfor
    for *each ant move* (i, j) **do**
        compute $\triangle\tau_{ij}$
        update trail matrix
    endfor
    $t \leftarrow t + 1$
endw

of the route referred to as the attractiveness, η_{ij}, for each origin-destination pair (i, j). The attractiveness is often also referred to as the *heuristic information* (Meuleau and Dorigo, 2002). The trail level τ_{ij} of the move from i to j is required and indicates how beneficial it has been in the past to make that particular move. The trail level therefore represents an *a posteriori* indication of the desirability of the move. As in previous metaheuristics discussed, an iteration limit $t^{\max}$ terminates the ACO.

For each ant k a solution is incrementally built using both the attractiveness and the trail level, weighted with preset parameters. Each ant's memory of tabu moves are updated accordingly to ensure only feasible solutions are created. Once all ants have their solutions, the pheromone trail matrix is updated by determining how many ants (solutions) traversed specific edges (i, j). The iteration number is incremented, and the process repeated until the maximum number of iterations have been reached.

Although Bullnheimer et al. (1999) could not improve on the best solutions found for sets of benchmark problems, the competitiveness of ACO is applaudable, given the immaturity of the approach to VRP variants compared to established, and well-researched metaheuristics. Detailed algorithmic approaches are provided by Dorigo and Gambardella (1997a,b), and Meuleau and Dorigo (2002) for the TSP and Gambardella et al. (1999) for the VRPTW which should stimulate and accelerate research in the respective fields and its variants. A robust algorithm presented by Reimann et al. (2004) is able to solve a number of VRP variants.

3.4 Conclusion

In the first review article of ACO theory, Dorigo and Blum (2005) comprehensively state that research contributions using metaheuristics as new as the ACO focus on proof-of-concept. This, however, is still true for the majority of theoretical papers on heuristics and metaheuristics. Solution quality and computational burden of various algorithm contributions are compared using benchmark problems (Van Breedam, 2001). The state-of-the-art for generic variants of the VRP are often implemented in commercial software applications. In such applications the parameter values for the specific metaheuristic are usually fixed, and are based on experiments with the benchmark data.

The majority of literature reviewed in this chapter either suggest parameter values that perform well in the majority of cases, or confirm that parameter settings are inherently problem specific. This review concludes with the observation that an intelligent routing system is required that will be able to observe the problem environment in which it is implemented, and dynamically adjust parameter settings in order to improve future solutions.

Chapter 4

An improved initial solution algorithm

Although Ichoua et al. (2003) employ a random insertion heuristic to create initial solutions, Van Breedam (2001) introduces an initial solution parameter in his evaluation of improvement algorithms, and finds that, in most cases, a good initial solution results in significantly better final results. This thesis proposes the use of a savings route construction heuristic based on Joubert (2003)[1]. Solomon (1987) concludes that, from the five initial solution heuristics evaluated, the Sequential Insertion Heuristic (SIH) proved to be very successful, both in terms of the quality of the solution, as well as the computational time required to find the solution. Section 3.2.1 reviews a number of route construction heuristics.

4.1 A route construction heuristic

An overview of the initial solution algorithm proposed in this thesis is provided in Algorithm 4.1. Initializing the algorithm requires a distance matrix. When using benchmark data sets only customer coordinates are provided, and the Minkowski distances are calculated using (2.46). If a Geographical Information System (GIS) is used, the travel distances can be determined through a process referred to as *geocoding* and *route calibration*. The initial solution algorithm also requires a travel time matrix for all node pairs (i, j).

4.1.1 Time-dependent travel times

Congestion effects become critical when time windows are imposed by customers, because in routing the temporal issue is of greater concern than the spatial issue. Three valuable contributions that incorporate both time dependent travel time and time windows are Ahn

[1]A revised version of this chapter has been published by Joubert and Claasen (2006)

Algorithm 4.1: Initial solution heuristic

Input: Customer data

Input: Fleet data

Initialize algorithm
repeat `Initialize tour`
    Establish tour starting time
    Assign vehicle
    repeat `Build tour`
        Establish route start time
        Identify seed customer
        repeat `Expand partial route`
            Determine insertion criteria
            Determine selection criteria
            Insert node
        until either *all nodes are routed* **or** *no node identified for insertion*
        Determine multi route feasibility
    until either *all nodes are routed* **or** *route expansion infeasible*
until either *all nodes are routed* **or** *vehicles are depleted*
Establish orphans
Report initial solution s

and Shin (1991), Fleischmann et al. (2004), and Ichoua et al. (2003).

Fleischmann et al. (2004) implement their routing algorithm when dynamic travel data is available through the Berlin traffic management system. Let:

$\tau_{ijk} \triangleq$ shortest travel time from node i to node j when the start time is in the time slot Z_k

with the day divided into K time slots $Z_k = [z_{k-1}, z_k], k \in \{1, 2, \ldots, K\}$. The planning horizon is denoted by the time interval $[z_0, z_K]$ which may coincide with the time window for the depot, becoming the time interval $[e_0, L_0^{\max}]$. The authors propose a smoothing of the travel time function with the introduction of

$\tau_{ij}(t) \triangleq$ travel time from node i to node j for the start time t at node i.

This is similar to the travel time proposed by Ichoua et al. (2003) where real traffic data is not accessible. A computationally efficient routine is introduced to acquire the travel time. A distance matrix $D = (d_{ij})$ is created for all $i, j \in \{1, 2, \ldots, n\}$ nodes. The planning horizon is also divided into K planning periods, while the edges are partitioned into C subsets $A = (A_c)_{1 \leq c \leq C}$ based on, for example, road type. To limit the number of speed values stored for each edge (i, j) for each time slot t, a travel speed v_{ct} is associated with each edge partition c for each time slot t. The dynamic travel time between nodes i and j can consequently be determined through Algorithm 4.2, if the travel start time at node i is denoted by $t_0 \in Z_k = [z_{k-1}, z_k]$.

Calculating the travel time matrix, however, is computationally expensive. Instead of calculating a travel time between each (i, j) pair for *each* time unit k in the scheduling period, Algorithm 4.3 introduces Time Window Compatibility (TWC) to only calculate travel time values for node pairs that have compatible time windows.

4.1.2 Time window compatibility

The introduction of the TWC concept assists in identifying, and eliminating, obvious infeasible nodes. This results in a more effective and robust route construction heuristic. The purpose of TWC is to determine the time overlap of all edges, or node combinations, (i, j), where $i, j \in \{0, 1, 2, \ldots, N\}$, and N the total number of nodes in the network. During the route construction phase, time window compatibility can be checked, and obvious infeasible nodes can be eliminated from the set of considered nodes. The Time Window Compatibility

Algorithm 4.2: Travel time calculation procedure

Input: Distance matrix $D = (d_{ij})$

Input: Travel speed matrix $V = (v_{ct})$

$t \leftarrow t_0$
$d \leftarrow d_{ij}$
$t' \leftarrow t + \frac{d}{v_{cZ_k}}$
while $t' > z_k$ **do**
    $d \leftarrow d - v_{cZ_k}(z_k - t)$
    $t \leftarrow z_k$
    $t' \leftarrow t + \frac{d}{v_{cZ_k}}$
    $k \leftarrow k + 1$
endw
$t_{ijt} = t' - t_0$

Algorithm 4.3: Incorporating time window compatibility with time dependent travel time

foreach *node pair* (i, j) **do**
    calculate TWC_{ij}
    if $TWC_{ij} \neq -\infty$ **then**
        foreach *time period* $k \in \{1, \ldots, K\}$ **do**
            calculate τ_{ijk} using Algorithm 4.2
        endfch
    else
        foreach *time period* $k \in \{1, \ldots, K\}$ **do**
            $\tau_{ijk} \leftarrow \infty$
        endfch
    endif
endfch

Matrix (TWCM) is a non-symmetrical matrix as the sequence of two consecutive nodes, i and j, is critical. Let:

$N \triangleq$ be the total number of nodes

$e_i \triangleq$ be the earliest allowed arrival time at customer i, where $i = \{0, 1, \ldots, N\}$

$l_i \triangleq$ be the latest allowed arrival time at customer i, where $i = \{0, 1, \ldots, N\}$

$s_i \triangleq$ be the service time at node i, where $i = \{0, 1, \ldots, N\}$

$t_{ij} \triangleq$ be the travel time from node i to node j, where $i, j = \{0, 1, \ldots, N\}$

$a_j^{e_i} \triangleq$ be the actual arrival time at node j, given that node j is visited directly after node i, and that the actual arrival time at node i was e_i, where $i, j = \{0, 1, \ldots, N\}$

$a_j^{l_i} \triangleq$ be the actual arrival time at node j, given that node j is visited directly after node i, and that the actual arrival time at node i was l_i, where $i, j = \{0, 1, \ldots, N\}$

$TWC_{ij} \triangleq$ be the time window compatibility when node i is directly followed by node j

TWC_{ij} indicates the entry in row i, column j of the TWCM. Consider the following five scenarios that illustrate the calculation of time window compatibility. Each scenario assume customer j to be serviced directly after customer i, a service time of one hour, and a travel time of two hours from node i to node j.

Scenario 1: if $a_j^{e_i} > e_j$ and $a_j^{l_i} < l_j$, illustrated in Figure 4.1. Customer i specifies a time

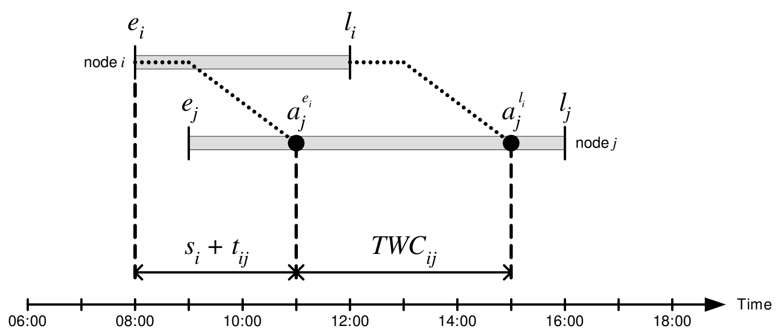


Figure 4.1: Time window compatibility scenario 1

window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [09:00,16:00]. If service at customer i starts at the earliest allowed time, e_i, then the actual arrival time at customer j would be calculated as

$$a_j^{e_i} = e_i + s_i + t_{ij} \tag{4.1}$$

In this scenario $a_j^{e_i} = 11{:}00$. Similarly, $a_j^{l_i}$ would be the actual arrival time at customer j, given that the actual arrival time at customer i was l_i, and is calculated as

$$a_j^{l_i} = l_i + s_i + t_{ij} \tag{4.2}$$

The difference between $a_j^{e_i}$ and $a_j^{l_i}$ indicates the time window overlap between the two nodes. The time window compatibility is calculated as

$$TWC_{ij} = a_j^{l_i} - a_j^{e_i} \tag{4.3}$$

For this example, the time window compatibility is four hours (04:00).

Scenario 2: if $a_j^{e_i} > e_j$ and $a_j^{l_i} > l_j$, illustrated in Figure 4.2. Customer i specifies a time

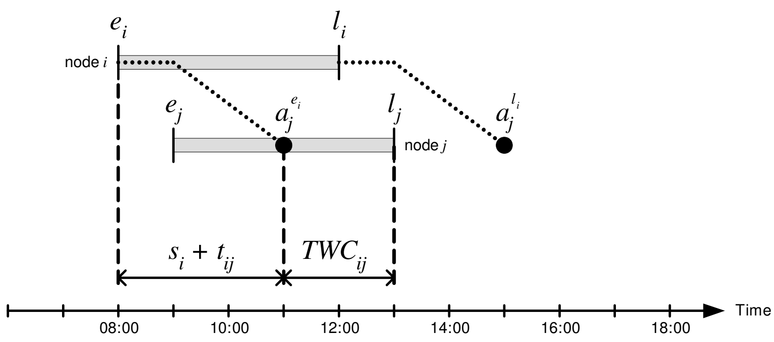


Figure 4.2: Time window compatibility scenario 2

window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [09:00,13:00]. The calculations for $a_j^{e_i}$ and $a_j^{l_i}$ are similar to (4.1) and (4.2), respectively. The time windows of customer i and customer j only partly overlap, and the time window compatibility is calculated as

$$TWC_{ij} = l_j - a_j^{e_i} \tag{4.4}$$

For this example, the time window compatibility is two hours (02:00).

Scenario 3: if $a_j^{e_i} < e_j$ and $a_j^{l_i} < l_j$, illustrated in Figure 4.3. Customer i specifies a time window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [12:00,16:00]. The calculations for $a_j^{e_i}$ and $a_j^{l_i}$ are similar to (4.1) and (4.2),

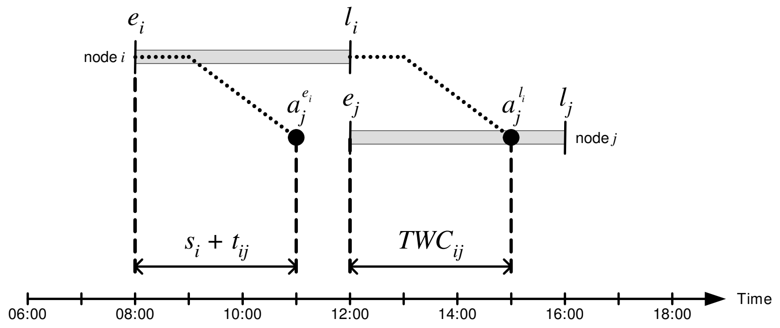


Figure 4.3: Time window compatibility scenario 3

respectively. The time windows of customer i and customer j only partly overlap, and the time window compatibility is calculated as

$$TWC_{ij} = a_j^{l_i} - e_j \tag{4.5}$$

For this example, the time window compatibility is three hours (03:00).

Scenario 4: if $a_j^{e_i}$ and $a_j^{l_i} < e_j$, illustrated in Figure 4.4. Customer i specifies a time

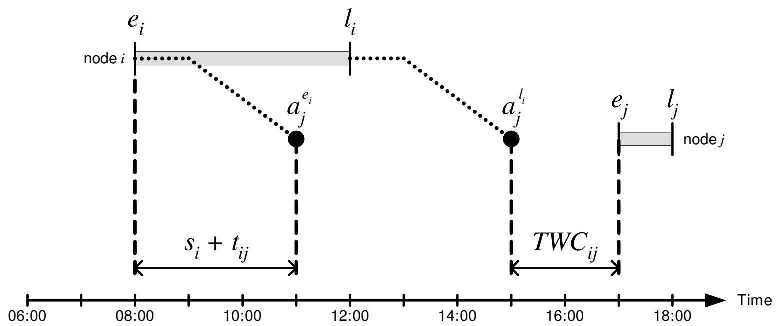


Figure 4.4: Time window compatibility scenario 4

window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [17:00,18:00]. The calculations for $a_j^{e_i}$ and $a_j^{l_i}$ are similar to (4.1) and (4.2), respectively. The time windows of customer i and customer j do not overlap. Even if customer i is serviced as late as possible, l_i, a waiting time is incurred at customer j. The time window compatibility is calculated as

$$TWC_{ij} = a_j^{l_i} - e_j \tag{4.6}$$

For this example, the time window compatibility is negative two hours (-02:00). The significance of the negative time is that it is possible, in this case, to service customer j after customer i, although the waiting time is penalized.

Scenario 5: if $a_j^{e_i}$ and $a_j^{l_i} > l_j$, illustrated in Figure 4.5. Customer i specifies a time

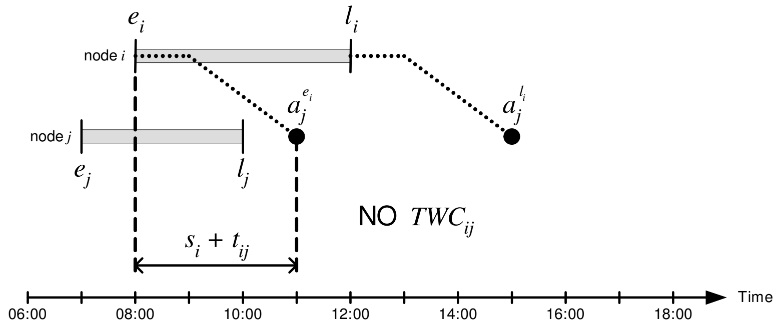


Figure 4.5: Time window compatibility scenario 5

window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [07:00,11:00]. The calculations for $a_j^{e_i}$ and $a_j^{l_i}$ are similar to (4.1) and (4.2), respectively. Although the time windows of customer i and customer j partly overlap, it is impossible to service customer j, even if customer i is serviced as early as possible, e_i. Therefor, no time window compatibility exist.

A generalized equation is proposed that will address all five scenarios illustrated, and is given by (4.7).

$$TWC_{ij} = \begin{cases} \min\{a_j^{l_i}, l_j\} - \max\{a_j^{e_i}, e_j\} & \text{if } l_j - a_j^{e_i} > 0 \\ -\infty & \text{otherwise} \end{cases} \tag{4.7}$$

The higher the value, the better the compatibility of the two time windows considered. Therefore an incompatible time window is defined to have a compatibility of negative infinity.

Example. Consider the following example with five nodes geographical distributed around a depot in Figure 4.6. In the example, node c has indicated two possible time windows. To accommodate multiple time windows, the customer is artificially split and treated as two separate nodes, c^1 and c^2, respectively, each having a single time windows. The time windows for each customer, including the depot, as well as the service time at each node, are given in Table 4.2. The distance matrix, $\overline{D}$, is calculated using the rectangular distance between nodes. With the grid provided in Figure 4.6, the

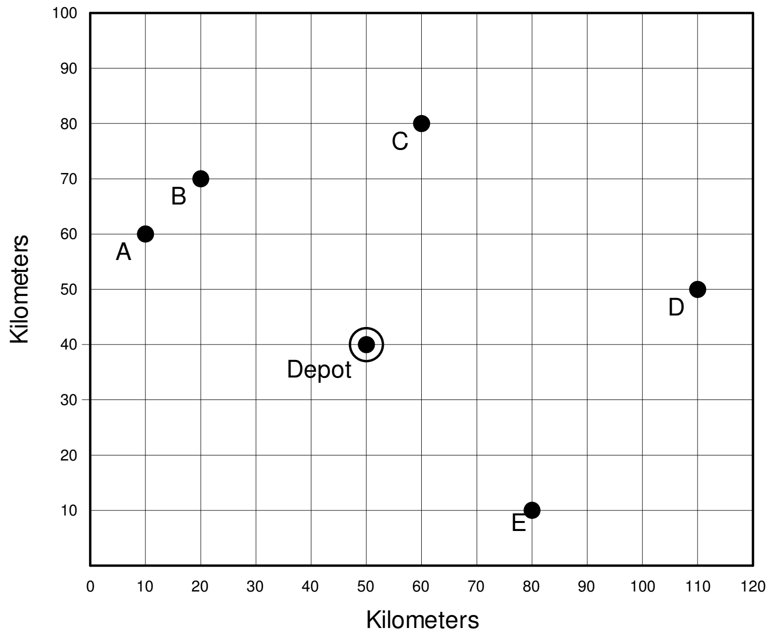


Figure 4.6: Geographical distribution of nodes around a depot

Table 4.2: Time windows and service times

Node (i)	**Time window** $(e_i; l_i)$	**Service time (in hours)** s_i
Depot	07:00 – 18:00	0.00
a	08:00 – 12:00	0.50
b	11:00 – 13:00	0.25
c^1	08:00 – 09:00	0.25
c^2	15:00 – 17:00	0.25
d	08:00 – 12:00	0.50
e	10:00 – 15:00	0.25

distances can be obtained through inspection.

$$\overline{D} = \begin{bmatrix} 0 & 60 & 60 & 50 & 50 & 70 & 60 \\ 60 & 0 & 20 & 70 & 70 & 110 & 120 \\ 60 & 20 & 0 & 50 & 50 & 110 & 120 \\ 50 & 70 & 50 & 0 & 0 & 80 & 90 \\ 50 & 70 & 50 & 0 & 0 & 80 & 90 \\ 70 & 110 & 110 & 80 & 80 & 0 & 70 \\ 60 & 120 & 120 & 90 & 90 & 70 & 0 \end{bmatrix}$$

If the average speed is known, the time matrix, $\overline{T}$, can be calculated, but in the presence of time dependent travel time, the travel times are calculated using Algorithm 4.2. For illustrative purposes in this example only, $\overline{T}$ is given. Values are in hours.

$$\overline{T} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0.5 & 1 & 1 & 2 & 2 \\ 1 & 0.5 & 0 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 0 & 0 & 1.5 & 1.5 \\ 1 & 1 & 1 & 0 & 0 & 1.5 & 1.5 \\ 1 & 2 & 2 & 1.5 & 1.5 & 0 & 1 \\ 1 & 2 & 2 & 1.5 & 1.5 & 1 & 0 \end{bmatrix}$$

With the information at hand, the time window compatibility matrix can be calculated. For the given example,

$$\overline{TWCM} = \begin{bmatrix} 11 & 4 & 2 & 1 & 2 & 4 & 5 \\ 4 & 3.5 & 2 & -\infty & -1.5 & 1.5 & 4 \\ 2 & 0.25 & 1.75 & -\infty & -0.75 & -\infty & 1.75 \\ 1 & 1 & -0.75 & 0.75 & -5.75 & 1 & 0.75 \\ 1.75 & -\infty & -\infty & -\infty & 1.75 & -\infty & -\infty \\ 4 & 1.5 & 2 & -\infty & -1 & 3.5 & 3.5 \\ 5 & -\infty & 0.75 & -\infty & 1.75 & 0.75 & 4.75 \end{bmatrix}$$

4.2 Improving the initial solution heuristic

Initialization criteria in Algorithm 4.1 refer to the process of finding the *seed customer*: the first customer to be inserted into a new route. Joubert (2003) proposes the use of the TWC concept to identify seed customers. When looking at the TWCM example, it is clear that the

Table 4.3: Number of infeasible time window instances

	Number of infeasible time windows		
Node	**as origin**	**as destination**	**Total**
Depot	0	0	0
a	1	2	3
b	2	1	3
c^1	0	5	5
c^2	5	0	5
d	1	2	3
e	2	1	3

incompatibility is distinct for specific nodes. It is therefor possible to identify *incompatible* nodes. As opposed to the two most common initialization criteria, namely *customer with earliest deadline*, and *furthest customer*, as suggested by Dullaert et al. (2001), the author of this thesis proposes the use of the TWCM to identify seed nodes based on their time window compatibility. Table 4.3 indicates the number of instances where a node has an infeasible time window with another node, either as origin, or as destination. Both nodes c^1 and c^2 have five infeasible instances. The two artificial nodes are representing the same customer c. It can be concluded that customer c is the most incompatible node, and is identified as the seed customer. Ties are broken arbitrarily. Should two nodes have the same number of infeasible time window instances, either of the two customers could be selected as seed customer.

It may be possible to not have any infeasible time window instances. In such a scenario, a *total compatibility* value, denoted by C_a^{total}, can be determined for each node a, and is calculated using either (4.8) or (4.9),

$$C_a^{\text{total}} = \sum_{i=1, i\neq a}^{M} TWC_{ia} + \sum_{j=1, j\neq a}^{M} TWC_{aj} + TWC_{aa} \qquad \forall a \tag{4.8}$$

$$C_a^{\text{total}} = \sum_{i=1}^{M} TWC_{ia} + \sum_{j=1}^{M} TWC_{aj} - TWC_{aa} \qquad \forall a \tag{4.9}$$

where M refers to all the unrouted nodes, including all instances of those nodes that are split artificially. The customer with the lowest total compatibility is selected as seed customer.

Once the seed customer has been identified and inserted, the SIH algorithm considers, for all unrouted nodes, the insertion position that minimizes a weighted average of the additional distance and time needed to include a customer in the current partially constructed route. This second step is referred to as the *insertion criteria*. Note that the terms *nodes* and *customers* are used interchangeably. The insertion and selection criteria can be simplified using the example illustrated in Figure 4.7. The partially constructed route in the example

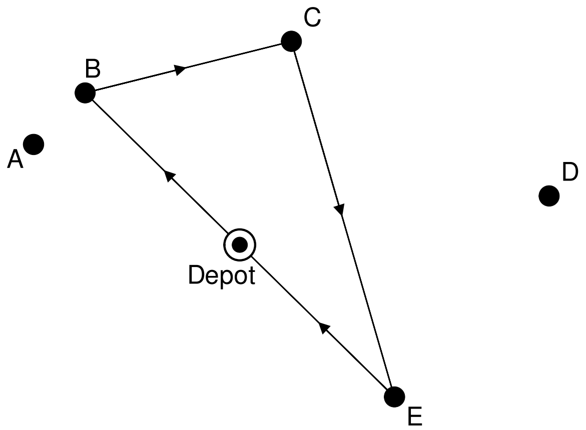


Figure 4.7: Sequential insertion of customers

consists of the depot and three routed nodes, namely B, C, and E. The route can be expressed as *Depot-B-C-E-Depot*. Nodes A and D are unrouted. The insertion criteria, denoted by $c_1(i, u, j)$, calculates the best position and associated cost, between two adjacent nodes i and j on the partial route, to insert a customer u, and is calculated for each of the unrouted nodes. Consider node A in the example. There are four edges where the node can be inserted, namely *Depot-B*, *B-C*, *C-E*, or *E-Depot*, as illustrated in Figure 4.8. Dullaert et al. (2001) extend Solomon's heuristic and determines $c_1(i, A, j)$ for the unrouted node A as

$$c_1(i, A, j) = \min_{p=\{1,2,\ldots,m\}} [c_1(i_{p-1}, A, i_p)] \tag{4.10}$$

in which m represents the routed nodes in the partially constructed route. If the expressions are generalized for all unrouted nodes u, the insertion criteria is calculated as

$$c_1(i, u, j) = \alpha_1 c_{11}(i, u, j) + \alpha_2 c_{12}(i, u, j) + \alpha_3 c_{13}(i, u, j) \tag{4.11}$$

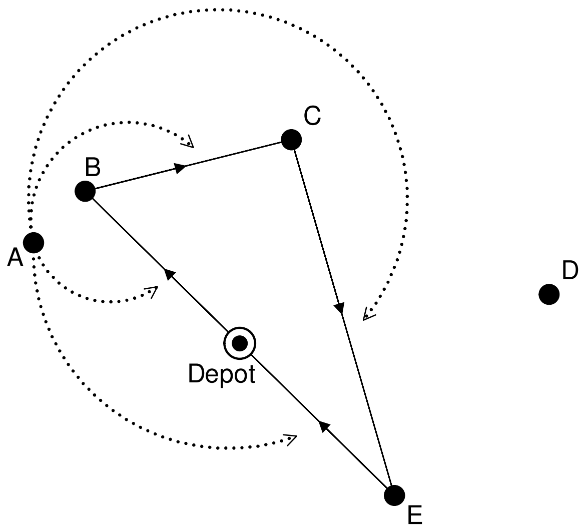


Figure 4.8: Selection criteria

with

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij}, \mu \geq 0 \tag{4.12}$$

$$c_{12}(i,u,j) = a_j^{new} - a_j \tag{4.13}$$

$$c_{13}(i,u,j) = ACS,\ AOOS, \text{ or } AROS \tag{4.14}$$

With the extension to Solomon's heuristic, the weighting factors α_i need not add up to 1. The additional distance, and the additional time needed to serve customer u after customer i, but before customer j is denoted by $c_{11}(i,u,j)$ and $c_{12}(i,u,j)$, respectively. The new actual arrival time at node j is denoted by b_j^{new} in (4.13). The vehicle savings criteria, denoted by $c_{13}(i,u,j)$, considers any one of three parallel approaches to vehicle cost, where the savings concepts introduced by Golden et al. (1984) are adapted. Let:

$F(z) \triangleq$ the fixed cost of the smallest vehicle that can service a cumulative route demand of z

$F'(z) \triangleq$ the fixed cost of the largest vehicle whose capacity is less than or equal to z

$P(z) \triangleq$ the capacity of the smallest vehicle that can service a demand of z

$Q \triangleq$ be the load of the vehicle currently servicing the route

$\overline{Q} \triangleq$ be the maximum capacity of the vehicle currently servicing the route

$Q^{new} \triangleq$ be the new load of the vehicle after the customer has been inserted into the route

$\overline{Q}^{new} \triangleq$ be the (new) capacity of the vehicle after the customer has been inserted into the route

The Adapted Combined Savings (ACS) is defined as the difference between the fixed costs of the vehicles capable of transporting the load of the route after, and before, inserting customer u, and is calculated by (4.15).

$$ACS = F(Q^{new}) - F(Q) \tag{4.15}$$

The Adapted Optimistic Opportunity Savings (AOOS) extends the ACS by subtracting the fixed cost of the vehicle that can service the unused capacity, and is calculated by (4.16).

$$AOOS = [F(Q^{new}) - F(Q)] - F(\overline{Q}^{new} - Q^{new}) \tag{4.16}$$

The Adapted Realistic Opportunity Savings (AROS) takes the fixed cost of the largest vehicle smaller than or equal to the unused capacity, $F'(\overline{Q}^{new} - Q^{new})$, into account as an opportunity saving. It only does so if a larger vehicle is required to service the current route after a new customer has been inserted. AROS is calculated by (4.17).

$$AROS = [F(Q^{new}) - F(Q)] - \delta(\omega)F'\left(\overline{Q}^{new} - Q^{new}\right) \tag{4.17}$$

where

$$\delta(\omega) = \begin{cases} 1 & \text{if } Q + q_u > \overline{Q} \\ 0 & \text{otherwise.} \end{cases}$$

Any *one* of these savings criteria can be used as all three outperformed previous best published results for the initial solution (Dullaert et al., 2001). Once the best position for each unrouted node has been determined, as illustrated in Figure 4.9, the customer that is best according to the *selection criteria*, is selected — the third step in the SIH algorithm. The procedure can be expressed mathematically as

$$c_2(i, u^\star, j) \quad = \quad \max_u [c_2(i,u,j)], u \text{ unrouted and feasible} \tag{4.18}$$

$$c_2(i,u,j) \quad = \quad \lambda(d_{ou} + t_{ou}) + s_u + F(q_u) - c_1(i,u,j), \lambda \geq 0 \tag{4.19}$$

The best customer, $u^\star$, is then inserted into the partially created route between its specific nodes i and j. From Figure 4.9, consider node D to be the best node. After inserting D into

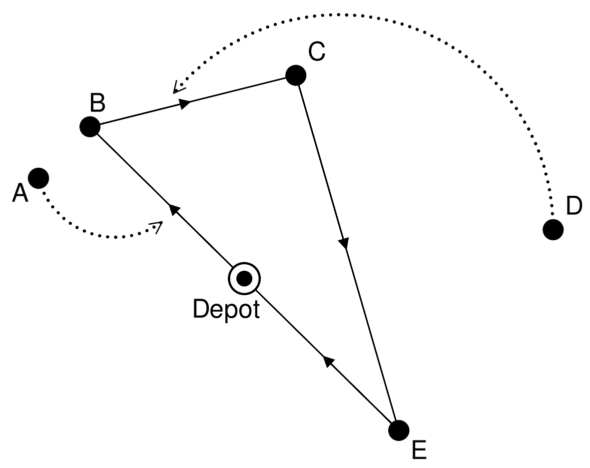


Figure 4.9: Best insertion position determined for each unrouted node

the current route, node A remains the only unrouted node, and the new route is illustrated in Figure 4.10, and can be expressed as *Depot-B-D-C-E-Depot*. The insertion process is

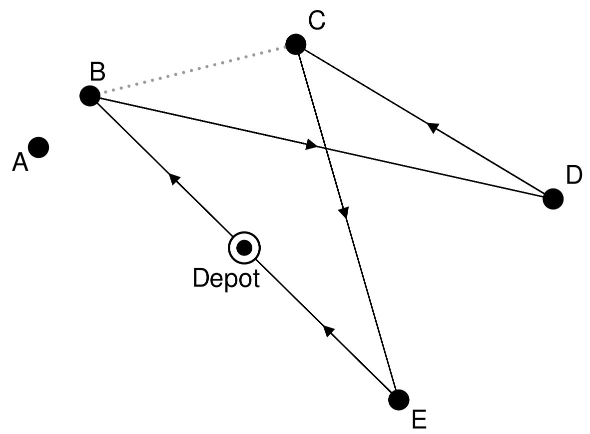


Figure 4.10: New route after inserting best customer

repeated until no remaining unrouted nodes have a feasible insertion place. A new route is then initialized and identified as the *current* route.

A shortcoming of Solomon's SIH 1987 is that it considers all unrouted nodes when calculating the insertion and selection criteria for each iteration. The fact that *all* unrouted nodes are considered makes it computationally expensive. The occurrence of obvious infeasible nodes in a partially constructed route becomes significant in the extended problem considered in this thesis. In each iteration, these criteria are calculated for each edge on the partially constructed route, irrespective of the compatibility of the time window of the node considered for insertion with the time windows of the two nodes forming the edge. For an improved case, consider the example where node u is considered for insertion between nodes i and j. As the TWCM is already calculated, it is possible to check the compatibility of node u with the routed nodes i and j. If either TWC_{iu} or TWC_{uj} is negative infinity ($-\infty$), indicating an incompatible time window, the insertion heuristic moves on and considers the next edge, without wasting computational effort on calculating the insertion and selection criteria. In the earlier example, eleven instances of infeasible time windows occur. If these instances are identified and eliminated, a computational saving in excess of 22% is achieved. The saving is calculated as the percentage of instances with time window incompatibilities of the total number of travel time instances.

4.3 Initial solutions

Solomon (1987) introduced 54 benchmark problems contained in six distinctive sets for the VRPTW, denoted by $c1$, $c2$, $r1$, $r2$, $rc1$, and $rc2$, each with 100 customer nodes. Each set highlights several factors that can affect the behavior of routing and scheduling heuristics. These factors include the geographical dispersion; the number of customers serviced by a vehicle, i.e. the relation between customer demand and vehicle capacity; and time window characteristics such as percentage of time-constrained customers, as well as the tightness and positioning of time windows.

The geographical data for the first group of problem sets are randomly generated using a uniform distribution (denote the corresponding problem sets by $r1$ and $r2$). The second group of sets are clustered (denote the corresponding problems sets by $c1$ and $c2$). A third semi-clustered group of sets have a combination of randomly distributed and clustered points (denote the corresponding problem sets by $rc1$ and $rc2$. Problem sets $r1$, $c1$, and $rc1$ have short scheduling horizons and along with vehicular capacities only allow a few customers to be serviced by a single vehicle. Problem sets $r2$, $c2$, and $rc2$ have long scheduling horizons, and when combined with large vehicular capacities, allows for a much higher number of customers being serviced by a single vehicle.

Homberger and Gehring (1999) extend the original problems to include problem sets having 200, 400, 600, and 1000 customer nodes. For illustrative purposes, Figure 4.11 shows the header of one of the Homberger and Gehring (1999) problem sets, as well as the first few customers. The depot is represented by customer '0'. The attributes for each customer include a customer number, coordinates, the demand, the earliest and latest allowed arrival, as well as the service time at each customer. The problem sets do unfortunately not accom-

```
c1_2_4

VEHICLE
NUMBER     CAPACITY
50          200

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

    0       70         70          0          0        1351           0
    1       33         78         20        750         809          90
    2       59         52         20          0        1240          90
    3       10        137         30          0        1172          90
    4        4         28         10          0        1183          90
    5       25         26         20        128         179          90
```

Figure 4.11: An excerpt of a problem set (Homberger, 2003)

modate a heterogeneous fleet, and the fleet structure proposed by Liu and Shen (1999b) is therefor used in this thesis — presented in Table 4.4 for each of the problem classes.

Time windows provided in the problem sets are *hard*, i.e. they allow neither early nor late arrivals. To create problem sets that will test the initial solution algorithm with *soft* time windows, a maximum lateness of $L^{\max} = 30$ time units is associated with each node, including the depot. Such time windows incur waiting time if arriving early, but allow late arrivals penalized at a unit cost of α.

Multiple scheduling is achieved through an elementary routine testing whether their is at least ρ time units between the return time of the current route and the end of the depot's time window. In this thesis the author uses an arbitrary value of $\rho = 60$ minutes.

Tables 4.5a through 4.5f show the results for 60 problem instances executed on an *Intel*® *Pentium*® *4* computer with a 3.6GHz processor (64Bit) and 3.25GB RAM.

Each table indicates the specific Homberger and Gehring (1999) problem instance from which the 100 customer data set as taken, the numbers of tours (vehicles) used in the initial solution, the total number of routes, the average time required to generate the initial solution, and the number of *orphans*. Orphans are customers from the data set that could not

Table 4.4: Heterogeneous fleet data (Liu and Shen, 1999a)

(a) Set $r1$

Type	Capacity	Cost
1	30	50
2	50	80
3	80	140
4	120	250
5	200	500

(b) Set $r2$

Type	Capacity	Cost
1	300	450
2	400	700
3	600	1200
4	1000	2500

(c) Set $c1$

Type	Capacity	Cost
1	100	300
2	200	800
3	300	1350

(d) Set $c2$

Type	Capacity	Cost
1	400	1000
2	500	1400
3	600	2000
4	700	2700

(e) Set $rc1$

Type	Capacity	Cost
1	40	60
2	80	150
3	150	300
4	200	450

(f) Set $rc2$

Type	Capacity	Cost
1	100	150
2	200	350
3	300	550
4	400	800
5	500	1100
6	1000	2500

Table 4.5a: Initial solution summary for the $c1$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$c1\_2\_1$	33	40	9	3
$c1\_2\_2$	27	30	14	1
$c1\_2\_3$	29	44	22	2
$c1\_2\_4$	19	19	30	1
$c1\_2\_5$	27	28	9	2
$c1\_2\_6$	28	37	12	2
$c1\_2\_7$	23	24	11	1
$c1\_2\_8$	23	23	14	0
$c1\_2\_9$	21	21	19	0
$c1\_210$	19	20	22	0

Table 4.5b: Initial solution summary for the $c2$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$c2\_2\_1$	39	50	10	11
$c2\_2\_2$	29	39	15	8
$c2\_2\_3$	27	46	20	7
$c2\_2\_4$	17	17	34	6
$c2\_2\_5$	24	24	10	6
$c2\_2\_6$	25	25	14	2
$c2\_2\_7$	27	30	15	3
$c2\_2\_8$	25	25	14	1
$c2\_2\_9$	28	35	19	2
$c2\_210$	23	24	20	0

Table 4.5c: Initial solution summary for the $r1$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$r1$_2_1	34	76	7	0
$r1$_2_2	37	71	6	0
$r1$_2_3	40	67	6	0
$r1$_2_4	59	70	6	0
$r1$_2_5	39	74	5	0
$r1$_2_6	42	69	6	0
$r1$_2_7	42	68	6	0
$r1$_2_8	57	70	7	0
$r1$_2_9	36	70	6	0
$r1$_210	39	68	6	0

Table 4.5d: Initial solution summary for the $r2$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$r2$_2_1	13	21	21	1
$r2$_2_2	9	17	34	1
$r2$_2_3	6	7	64	0
$r2$_2_4	6	9	84	0
$r2$_2_5	9	12	28	0
$r2$_2_6	9	9	57	0
$r2$_2_7	8	10	74	0
$r2$_2_8	6	7	94	0
$r2$_2_9	9	11	41	0
$r2$_210	10	11	41	0

Table 4.5e: Initial solution summary for the $rc1$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$rc1\_2\_1$	30	51	9	0
$rc1\_2\_2$	26	48	9	0
$rc1\_2\_3$	27	46	10	0
$rc1\_2\_4$	34	46	13	0
$rc1\_2\_5$	26	47	9	0
$rc1\_2\_6$	29	49	9	0
$rc1\_2\_7$	30	48	10	0
$rc1\_2\_8$	25	47	10	0
$rc1\_2\_9$	27	47	10	0
$rc1\_210$	35	48	11	0

Table 4.5f: Initial solution summary for the $rc2$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$rc2\_2\_1$	13	26	16	0
$rc2\_2\_2$	11	26	23	0
$rc2\_2\_3$	11	24	31	1
$rc2\_2\_4$	11	18	31	0
$rc2\_2\_5$	12	20	19	0
$rc2\_2\_6$	12	18	18	0
$rc2\_2\_7$	9	18	21	0
$rc2\_2\_8$	11	18	22	0
$rc2\_2\_9$	13	18	21	0
$rc2\_210$	13	19	25	0

feasibly be included in the initial solution. Ten iterations were used to calculate the average time values. Orphans are a result of the specific problem instance. The time dependent travel times that were calculated using randomly generated edge types, v_{cT}, may result in a situation whereby a customer can not be serviced within the time window of the depot, even if such customers are serviced by a dedicated vehicle.

A sample of an initial solution output file for the $r2\_2\_3$ problem set (see Table 4.5d) is provided in Appendix A. The initial solution indicates the algorithm's ability to generate more than one route per vehicle, and indicates the vehicle type assigned to the specific route. Each line represents a route, with each route starting and ending at the depot. Sequential numbers in each route represent the customers and the sequence in which customers are serviced. In the solution for the $r2\_2\_3$ problem all nodes are routed, and no orphans exist.

4.4 Conclusion

To establish an initial solution that addresses not only time windows, but also time dependent travel times and a heterogeneous fleet, requires a computational expensive routine. In this chapter the author introduced the concept of Time Window Compatibility (TWC) to ease the computational burden. The concept of TWC is also employed to identify seed customers as the most incompatible customer nodes.

Data sets from literature were adapted to create test problems for which the initial solution algorithm found solutions within seconds. The initial solutions generated in this chapter is used as inputs to the route improvement metaheuristics that are developed in Chapters 5 through 6.

Chapter 5

A Tabu Search solution algorithm

The TS examines a trajectory sequence of solutions and moves to the best neighbor of the current solution. To avoid cycling, solutions that were recently examined are forbidden, or tabu, for a number of iterations (Gendreau et al., 2002). Section 3.3.1 reviews the basic structure of the TS.

Taillard (1993) introduces a feature whereby the main problem is decomposed into independent subproblems so that the algorithm can be parallelized on multiple processors. Each subproblem is solved on a different processor before the tours are grouped together to construct a solution to the original problem. The new solution is then decomposed, and the process repeats itself for a given number of times. A random selection of components in the decomposition process ensures the algorithm produces different solutions from one execution to the next. In this thesis an approach similar to that of Taillard (1993) and Rochat and Taillard (1995) is followed, albeit on a single processor. The approach can be parallelized through the coding structure in future research, but recent software technology, i.e. cluster scheduling such as the *MATLAB Distributed Computing* system, provides the software the ability to automatically determine which segment of an algorithm can be parallelized on multiple clustered processors without adapting the code.

The chapter starts with a brief discussion of the main elements of a TS algorithm, followed by the TS proposed in this thesis, and a detailed discussion of each phase of the TS. The chapter concludes with an analysis of the algorithm's results for problems based on integrated data sets of Solomon (1987), Homberger and Gehring (1999), and Liu and Shen (1999a,b).

5.1 Elements of the tabu algorithm

Tabu list A list of the last few moves (or solutions). The *memory* of moves can be recency or frequency-based. Short-term recency-based memory forbids cycling around a local neighborhood in the solution space through setting the last T moves as Tabu. Recently made moves are stored in a mechanism that is referred to as the *Tabu-Move* list. The number of moves in the list is determined by the tabu list size, denoted by T. The list operates on a first-in-first-out principle. Other recency information that is stored in the Tabu list is the solution configurations. The larger the value of T, the longer the moves and solutions stay tabu. The *Tabu-Solution* list is a set of solutions that have been created recently by exchanging segments between routes. The solutions are coded into an integer string. The total cost of the solution is also attached to the string.

Long-term frequency-based memory allows searches to be conducted in the most promising neighborhoods. The frequency-based memory provides additional information of how many times a tabu move have been attempted. To alleviate time and memory requirements, it is customary to record an attribute of a tabu solution, and not the solution itself.

Candidate list TS makes use of a candidate list that provides a list of moves to evaluate. One move of the candidate list is chosen to proceed with the search. The candidate list plays an important role in the performance of TS.

Intensification and diversification Two memory-based strategies that form a fundamental principle of TS. Gendreau (2003) claims diversification to be the single-most important issue in designing a TS. With the use of the intensification strategy regions around attractive solutions are more thoroughly searched, and typically operates by restarting a search from a solution previously found to yield good results. The restart is achieved through the candidate list representing attractive regions. Diversification, on the other hand, encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from previous solutions. The probabilistic diversification and intensification introduced by Rochat and Taillard (1995) is also referred to as the Adaptive Memory Procedure (AMP).

Penalized objective function The objective function of a solution s is denoted by $f_1(s)$ and is calculated by (5.1) as the sum of the travel times of all routes and tours, and the total lateness at all customers (Ichoua et al., 2003).

$$f_1(s) = \sum_{\text{Tours}} \sum_{\text{Routes}} t + \sum_{\text{Customers}} \alpha_i y_i \tag{5.1}$$

In the calculation α_i denotes the lateness penalty for customer i, while $y_i = \max\{0, a_i - l_i\}$. The actual arrival time at customer i is denoted by a_i, while l_i denotes the latest allowed arrival time at customer i. The design of the algorithm ensures that $a_i \leq l_i + L_i^{\max}$, where $L_i^{\max}$ is the maximum allowed lateness at customer i. The objective function is artificially adapted to incorporate a significant penalty for any unrouted customers, referred to as *orphans*. The artificial objective function, $f_2(s)$, is expressed in (5.2),

$$f_2(s) = f_1(s) + \beta o \tag{5.2}$$

where β is a nonnegative penalty factor, and o the number of orphans in the final solution. Orphans are only created if the time window of the customer is completely incompatible with that of the depot, even if it is serviced by a dedicated vehicle.

Stopping criteria The search is terminated once a preset maximum number of iterations of the main TS algorithm have been reached. An alternative stopping criteria could be a predetermined number of attempts being made to set the same solution in the Tabu-Solution list as the new current solution. This indicates that the search has been caught in a local optimum, hence terminating the search.

5.2 Tabu algorithm

The phased approach of the TS algorithm, similar to the implementation of Taillard et al. (1997) and Gendreau et al. (1999), is illustrated in Algorithm 5.1. Data structures are indicated with sans serif font, while functional routines are indicated with `typewriter font`.

5.2.1 Initialization

The initial solution algorithm proposed in Chapter 4 forms the basis of the initialization phase, but generates only a single initial solution, s. As I, preferably *different*, initial solutions are required, the routine in Algorithm 5.2 is proposed. For each initial solution required, a random node $I_i^\star$ is identified and removed from the problem set P. The remaining nodes in P' are used to create an initial solution using the improved initial solution algorithm proposed in Chapter 4. After the nodes in P' have been routed, the identified node $I_i^\star$ is reinserted into the first feasible position. The result is a set of initial solutions

Algorithm 5.1: Tabu Search (TS) Overview

Input: stopping criteria

Input: Adaptive Memory size, M

begin Initialization (Section 5.2.1)

  construct I unique initial solutions $\boldsymbol{s} = \{s_1, s_2, \ldots, s_I\}$

  $\hat{x} \leftarrow \min\limits_{i \in \{1,\ldots,I\}} \{s_i\}$

  decompose $\boldsymbol{s}$ into independent tour set T

  store M best tours of $T \cup$(Adaptive Memory) in the Adaptive Memory

end

begin Optimization (Section 5.2.2)

  while *stopping criteria is not met* **do**

    construct a biased solution, x from the tours in Adaptive Memory

    $x^{\text{current}} \leftarrow x$

    for W *iterations* **do**

      $x^{\star} \leftarrow$ locally optimized x^{current}

      $x^{\text{current}} \leftarrow x^{\star}$

      if $x^{current} < \hat{x}$ **then**

        $\hat{x} \leftarrow x^{\text{current}}$

      endif

    endfor

  endw

  decompose x^{current} into independent tour set T

  store M best tours of $T \cup$(Adaptive Memory) in the Adaptive Memory

end

report incumbent $\hat{x}$

Algorithm 5.2: Tabu Search (TS) Initialization

Input: Problem set P, with $|P| = n$ nodes

Input: Number of initial solutions required, I

identify $I^\star \subset P$, a randomly identified subset with I nodes from problem set ;
foreach $I_i^\star \in I^\star$ **do**
    $P' \leftarrow P \backslash \{I_i^\star\}$;
    find initial solution s by executing `Initial solution heuristic` with P' ;
    re-insert $I_i^\star$ into initial solution to create s_i
endfch

$\boldsymbol{s} = \{s_1, s_2, \ldots, s_I\}$. Each initial solution's tours are stored in the adaptive memory, and associated with it the objective function value of the initial solution from which the tour originates. All tours consisting of only a single node are removed from the adaptive memory.

5.2.2 Optimization

The TS optimization routine listed in Algorithm 5.3 terminates after executing a predefined number of local optimization iterations, denoted by $I^{\max}$. A partially constructed tour is created through iteratively selecting tours from the adaptive memory, and removing all tours from the adaptive memory that share nodes with the selected tour. The probability of selecting any tour is based on the objective function associated with the tour, which in turn is taken from the solution from which the tour originates. Glover (1990) notes that the use of probabilities, based on past performance, as an underlying measure of randomization yields efficient and effective means of diversification. The better a solution, the higher the probability of selecting a tour from that solution. Once a tour is selected from the adaptive memory, all tours sharing nodes with the selected tour are removed from memory. Removing tours from the adaptive memory ensures each node is represented only once in the partially constructed tour. The selection of tours from the adaptive memory, and the removal of tours with common nodes, is repeated until no more tours remain in the adaptive memory. As not all nodes are represented, the partially constructed tour denoted by s, is completed by inserting the remaining unrouted nodes into feasible positions of s. The resulting tour, denoted by $s^\star$, is achieved through either identifying positions on a current route, creating a new route on a current tour, or creating a new tour with its associated vehicle.

Algorithm 5.3: Tabu Search (TS) Optimization

Input: Incumbent solution, $\hat{x}$

Input: Iteration limit for local optimization, $I^{\max}$

Input: Frequency parameter, ζ

$s = \{\cdot\}$
assign set of tours, $A \leftarrow$ Adaptive Memory
repeat
    select $a \in A$
    $s \leftarrow s \cup a$
    $A \leftarrow A \ominus (a \cap A)$
until $A = \{\cdot\}$
$s^{\star} \leftarrow s \oplus (\{1, 2, \ldots, N\} \ominus s)$
$i \leftarrow 0$
repeat
    $i \leftarrow i + 1$
    if $\left\lfloor \frac{i}{\zeta} \right\rfloor = \frac{i}{\zeta}$ **then**
        `exchange heuristic` $j = \{1, 2\}$
    else
        select `exchange heuristic` $j \in \{1, 2\}$ with probability p_j
    endif
    ${s'}_j \leftarrow e_j\left(s^{\star}\right)$
    $s' \leftarrow \min\limits_{j} \left\{ s'_j \right\}$
    $x' \leftarrow f\left(s'\right)$
    if $x' < \hat{x}$ **then**
        $\hat{s} \leftarrow s'$
        $s^{\star} \leftarrow s'$
    endif
until $i > I^{\max}$

Two exchange operators are considered. The first operator removes a randomly selected node from one tour and inserts the node into the best possible position in another tour that has the same vehicle type. The second operator also removes a randomly selected node from an origin tour, but selects the best insertion position for the node on a tour having a different vehicle type than the origin tour.

Initially the probability of selecting either of the operators is equal. A frequency parameter, ζ, ensures that every ζ iterations *both* operators are used to create perturbations. The probability of the operator producing the best solution is then increased relative to its current probability. Consider, during a general iteration, the first operator having a weight of $\alpha = 30$ and the second operator having a weight of $\beta = 60$. If both operators are executed, and the first operator yields a better solution, its weight will be increased by a factor γ. In this thesis γ is arbitrarily set to 2. The new probability of selecting the first operator is

$$\begin{aligned} p_1 &= \frac{\gamma\alpha}{\gamma\alpha + \beta} \\ &= \frac{2 \times 30}{2 \times 30 + 60} \\ &= 0.5\dot{0}, \end{aligned}$$

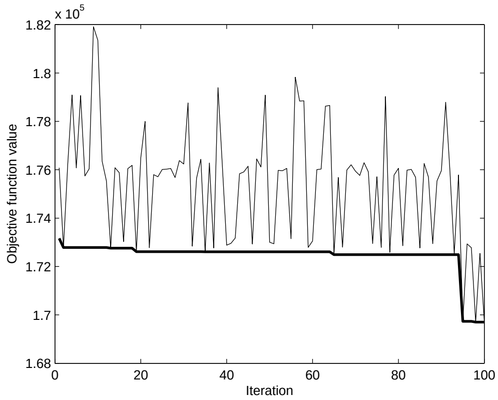
and the probability of selecting the second operator is calculated as

$$p_2 = 1 - p_1.$$

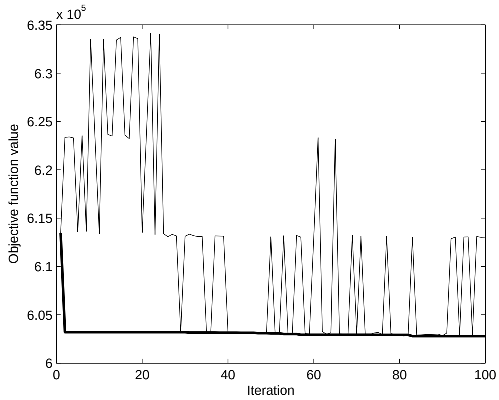
5.3 Results and analysis

The TS algorithm proposed in this thesis contains a random component similar to the algorithm proposed by Rochat and Taillard (1995). This means that two runs of the algorithm will generally produce two different solutions. Figure 5.1 provides graphs for a random selection of problems. Each graph indicates the iteration number on the x-axis, while the objective function value is represented on the y-axis. The thinner of the two lines on each graph represent the actual objective function value of the solution for the given iteration, while the thick line represents the incumbent — the best solution found thus far, at that iteration.

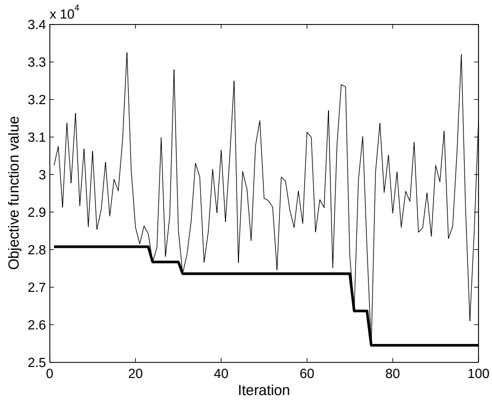
It is noticeable that the incumbent for the first iteration is frequently lower than the actual iteration value. This is the result of the incumbent being represented by one of the ten initial solutions created for the TS, whereas the first iteration's solution is created through the solution-building mechanism that selects tours from the adaptive memory. The incumbent,



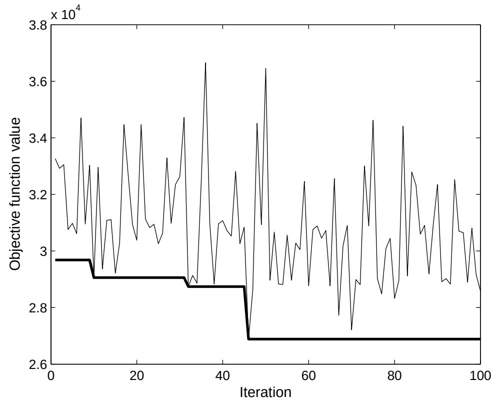
(a) Problem $c1\_2\_3$



(b) Problem $c2\_2\_3$



(c) Problem $rc1\_2\_8$



(d) Problem $rc2\_2\_8$

Figure 5.1: Selection of TS result graphs

furthermore, is never improved by more than 10% over the 100 iterations, reflecting on the high quality initial solution proposed in Chapter 4.

Because of the randomness inherent in the structure of the proposed TS, the results presented in Appendix B sees four independent runs executed, with Tables B.1(a) through B.1(f) providing the objective function values for each of the runs, as well as the average objective function value obtained. The last column of the result tables provide the average time (in seconds) required to obtain a solution. The average time is provided under the assumption that time-dependent travel time matrices are not available, and that such matrices have to be established once, and adhere to the triangular inequality

$$t_{ik} + t_{kj} \geq t_{ij} \qquad \forall i, j, k \in \{1, 2, \ldots, N\}. \tag{5.3}$$

Although Toth and Vigo (2002b) interpret the triangular inequality as being inconvenient to deviate from the direct link between nodes i and j, it may be practical to adjust the link from i to j to rather pass via node k without actually visiting node k. This occurs when the direct link is heavily congested during peak times. Adjusting the route selection in combination with time-dependent travel times are highly dependent on an accurate GIS.

5.4 Conclusion

A Tabu Search (TS) algorithm is proposed that generates a number of initial solutions as input, from where tours are added to an Adaptive Memory Procedure (AMP). During each consecutive iteration, tours are selected from the AMP in a biassed manner to construct a new solution. Non-tabu, feasible solutions are generated in an attempt to escape local minima.

The algorithm is coded in *MATLAB*, and tested on 60 benchmark data sets adapted from literature. The sets are adapted to accommodate multiple routes per tour, as well as a heterogeneous fleet in an environment where time dependent travel times occur. The results are promising, yielding solutions between 670 and 4762 seconds on a standard *Intel Pentium Centrino* laptop computer with a 1.5GHz processor and 512MB of RAM. Four independent runs are executed for each of the 60 problems. The Absolute Mean Deviation (AMD) of the solution quality between the 240 runs is 3.6%, indicating an algorithm that produces consistent solutions between runs.

In the next chapter, the GA is investigated as an alternative to the TS.

Chapter 6

A Genetic Algorithm

In this thesis the approach by Tan et al. (2001c) is followed whereby a Genetic Algorithm (GA) uses a path representation to code chromosomes (routes). For example, the chromosome string 4-5-2-3-1 represents a route that starts at node 4, followed by node 5, then 2, 3, and 1 before returning to node 4. Each element in the chromosome is referred to as an *allele*. For a problem with n customers, each chromosome will be an integer string with n elements. Although elementary crossover routines often destroy the validity of tours and routes, specific crossover routines have been developed to ensure that tours and routes remain valid, and keeps improving.

A slightly adapted version of the GA discussed in Algorithm 3.3 is provided in Algorithm 6.1. The GA requires a generation limit similar to the iteration limit for TS and SA. The population size determines the number of solutions in a single generation. The population subdivision parameters establishes the fraction of the population that will undergo specific genetic manipulation. To ensure the natural phenomena of *survival of the fittest*, the elitist parameter p_e ensures that the p_e fittest solutions in a given generation g is *exactly* copied to the next generation $g+1$. The mutation parameter p_m determines the number of chromosomes that will undergo random changes, or mutation. The crossover parameter, p_c, determines the number of solutions that will produce offspring by sharing elements of its chromosomes.

The algorithm is initialized with the generation of p solutions, each containing a single TSP string of nodes. Vas (1999) states that initial solutions can be generated either randomly or heuristically, while Tan et al. (2001c) suggest a combination of solutions: some generated using an efficient Push Forward Insertion Heuristic (PFIH), and the balance generated randomly.

Algorithm 6.1: Genetic Algorithm (GA) overview

Input: Generation limit $g^{\max}$

Input: Population size p

Input: Population subdivisions p_e, p_m, and p_c such that $p_e + p_m + p_c = p$

$g \leftarrow 0$
begin `initialization`
    generate feasible TSP solutions $x_1^0, \ldots, x_p^0$
end
repeat
    $g \leftarrow g + 1$
    cluster TSP solutions
    determine fitness of TSP solutions
    begin `elite`
        Copy p_e best solutions from generation g to generation $g + 1$
    end
    begin `mutation`
        Include p_m mutated solutions in generation $g + 1$
    end
    begin `crossover`
        Choose $\frac{p_c}{2}$ non-overlapping pairs of solutions from generation g
        execute crossover perturbations
        Include new solutions in generation $g + 1$
    end
until $g = g^{\max}$
$x^\star \leftarrow \min\limits_{i \in \{1,\ldots,p\}} \{x_i^g\}$
$\hat{x} \leftarrow$ locally optimized $x^\star$

The GA proceeds for $g^{\max}$ generations. During each generation, the single string solution, also referred to as a TSP solution, is clustered and assigned to vehicles. Each solution's fitness is calculated as the objective function of the specific solution. Based on the fitness, the algorithm reproduces the next generation through a combination of *cloning* the p_e fittest solutions exactly to the next generation, mutating p_m solutions through small changes referred to as perturbations, and creating p_c new offspring by performing crossover perturbations on a selection of generation g solutions.

The following sections discuss some of the elements of the GA in more detail.

6.1 Initialization

The simplest and computationally most efficient way of generating p initial solutions, each containing n customers, is to create p random permutations of integers between 1 and n. Each integer value represents a specific customer. To generate a population of 200 solutions (chromosomes), each with 200 nodes takes *MATLAB* on average 0.014 seconds (average obtained from 10,000 independent runs) on a standard *Intel Pentium Centrino* laptop computer with a 1.5GHz processor and 512MB of RAM.

As an alternative, initial solutions can be generated using the algorithm presented in Chapter 4, and adapted for the TS in Algorithm 5.2.

6.2 Clustering

Each chromosome represents a solution in the form of a single integer string, similar to the TSP strings proposed by Michalewicz (1992). The difficulty with having a single string to represent multiple tours and routes is that the chromosome needs to be *clustered*, and assigned to vehicles.

Although Tan et al. (2001c) simply adds the first allele of the chromosome to the end of the current tour until vehicle capacity is met, the author of this thesis propose the clustering routine presented in Algorithm 6.2 to address multiple scheduling. The first allele of the chromosome is considered for insertion on each edge of each route of the current tour, and not only at the end of the route. If no position is found for the customer, a new route on the current tour is considered. If an additional route leads to infeasibilities, a new tour is initialized, and the customer is inserted. A customer is only orphaned if it can not be serviced by a dedicated tour.

Algorithm 6.2: GA clustering

```
Input: population
foreach chromosome in population do
    repeat
        found ← 0
        forall the routes of current tour do
            forall the edges on current route do
                if feasible insertion then
                    found ← 1
                endif
            endfall
        endfall
        if found = 1 then
            insert customer
        else if multiple routes are feasible then
            insert customer into new route
        else
            create new current tour
            create new first route
        endif
    until all customers are routed, or vehicles are depleted
    report orphans
endfch
```

6.3 Mutation

A proportion, p_m of all chromosomes in a given generation is *mutated* to ensure that the GA does not get stuck in a local optimum (Vas, 1999). The proportion is typically very low to ensure that good chromosomes remain intact. Michalewicz (1992) introduces a non-uniform mutation rate whereby the number of chromosomes mutated decreases to ensure that the solution space is searched widely during early generations, and only searched locally in later generations.

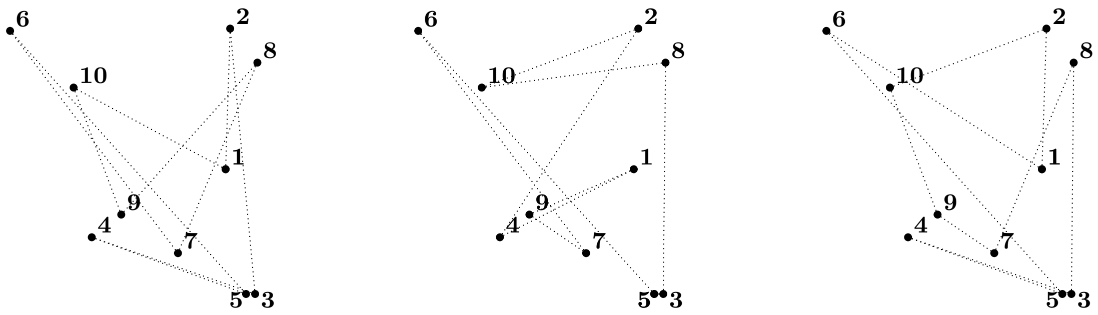
In the majority of applications binary representation is used and mutation involves changing a 0 value to 1, and vice versa. In this thesis the approach of Tan et al. (2001a) is followed whereby randomly selected customers are swapped in an integer string representation of a chromosome.

6.4 Crossover

Crossover operators are concerned with producing offspring solutions for the next generation from two parent solutions from the current generation. Parents are selected using a biased roulette wheel. A number of the operators produce only a single offspring from the two parents, while others produce two offspring. To illustrate the various crossover operators, the first ten nodes of the C2-2-2 problem set is used.

6.4.1 Enhanced Edge Recombination (EER)

Whitley, Starkweather & Fuquay (as cited by Michalewicz (1992)) developed the Edge Recombination (ER) crossover technique which they claim transfer more than 95% of the edges from the parents to a single offspring. To illustrate the ER, consider two single string TSP solutions, A and B, illustrated in Figures 6.1(a) and 6.1(b) respectively. The edge table created in Table 6.1(a) lists for each node all the neighbouring nodes from both parent solutions. The single offspring, denoted by C, starts by selecting a starting element. Starkweather et al. (1991) state that the starting element can be either chosen randomly from the set of elements which has the fewest entries in the edge table, or a random choice between the starting element from either parent A or B. The latter option is used in this thesis. Of the elements that have links to the last element in C, choose the element which has the fewest number of unassigned links in the edge table entry, breaking ties randomly. The process is repeated until the new offspring chromosome is complete.



(a) $A = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$ (b) $B = [5\ 6\ 9\ 1\ 4\ 2\ 10\ 8\ 3\ 7]$ (c) $C' = [1\ 2\ 10\ 9\ 7\ 8\ 3\ 4\ 5\ 6]$

Figure 6.1: Two parent solutions illustrating the ER crossover

Table 6.1: Edge lists

(a) ER edge list

City	Links
1	2, 4, 9, 10
2	1, 3, 4, 10
3	2, 4, 5, 8
4	1, 2, 3, 5
5	3, 4, 6
6	5, 7
7	6, 8, 9
8	3, 7, 9, 10
9	1, 7, 8, 10
10	1, 2, 8, 9

(b) EER edge list

City	Links
1	2, 4, 9, 10
2	1, 3, 4, 10
3	2, 4, 5, 8
4	1, 2, 3, 5
5	3, 4, -6
6	-5, -7
7	-6, 8, 9
8	3, 7, 9, 10
9	1, 7, 8, 10
10	1, 2, 8, 9

Suppose element 1 is selected from A as starting element in C. Since 1 has been assigned to C, all occurrences of 1 is removed from the edge list. Element 1 has links to 2, 4, 9 and 10, each having 3 remaining links in the edge table. Element 2 is randomly chosen as next element in C and all element 2's occurrences are removed from the edge table. Element 2 has links with 3, 4 and 10, of which 4 and 10 have only 2 remaining links in the edge table. Element 10 is chosen randomly as the next element in C, having links to elements 8 and 9. Element 9 has the least (2) number of remaining links in the edge list, and chosen as the next element in C. The process continues until $C = [1\ 2\ 10\ 9\ 7\ 8\ 3\ 4\ 5\ 6]$.

To enhance the random breaking of ties when selecting among elements, Starkweather et al. (1991) changed the edge list to indicate common edges. This is achieved by flagging a common edge by inverting, for example, 3 to -3 if an element has a common edge to element 3 in both parents. Table 6.1(b) indicates the edge list with flagged common edges. When a tie exist between elements, preference is given to the element with the highest number of remaining flagged elements. If a tie still exists, it may be broken randomly. Following the same procedure as for the ER example above, a slightly different offspring $C' = [1\ 2\ 10\ 9\ 7\ 6\ 5\ 3\ 4\ 8]$ is obtained. The offspring chromosome is illustrated in Figure 6.1(c). The only *new* edge in the offspring is the edge connecting elements 6 and 1. Hence, 90% of the edges are transferred from the parents to the offspring solution.

6.4.2 Merged Crossover (MX)

The MX was first introduced by Blanton and Wainwright (1993) and is based on the notion of a global precedence among genes of any chromosome, rather than defining a precedence among genes specific to parents in a local crossover such as the EER. A number of precedence vectors have been established in literature.

Latest allowed arrival time

Chen et al. (1998) state that there is a natural precedence relationship among all customers based on the upper limit of their time windows. The precedence list, denoted by P, for the example problem is $P = [2\ 8\ 3\ 5\ 7\ 1\ 10\ 6\ 9\ 4]$, based on the time window details provided in Table 6.2.

To illustrate the crossover, we consider parents A and B from Figure 6.1. The first elements from both parents are considered: element 5 from B appears before element 1 from A in the precedence list P, and is selected as first element in offspring C. To maintain

Table 6.2: Time window details for customers from the $C2$-2-2 problem set

Customer, i	**Earliest allowed arrival, e_i**	**Latest allowed arrival, l_i**
1	2808	2968
2	668	828
3	1021	1181
4	0	3481
5	1922	2082
6	0	3451
7	2597	2757
8	906	1066
9	0	3475
10	0	3445

validity, elements 1 and 5 are swapped in parent A.

$A = [\mathbf{5}\ 2\ 3\ 4\ \mathbf{1}\ 6\ 7\ 8\ 9\ 10]$

$B = [5\ 6\ 9\ 1\ 4\ 2\ 10\ 8\ 3\ 7]$

$C = [5 \star \star \star \star \star \star \star \star \star]$

Next, the second elements of each parent is considered. As element 2 from A appears before element 6 from B in the precedence list, element 2 is placed in the offspring, and elements 2 and 6 are swapped in parent B.

$A = [5\ 2\ 3\ 4\ 1\ 6\ 7\ 8\ 9\ 10]$

$B = [5\ \mathbf{2}\ 9\ 1\ 4\ \mathbf{6}\ 10\ 8\ 3\ 7]$

$C = [5\ 2 \star \star \star \star \star \star \star \star]$

The process is repeated until the offspring chromosome is completed with $C = [5\ 2\ 3\ 1\ 4\ 6\ 7\ 8\ 9\ 10]$. The MX approach is denoted by MX_{l_i}.

Earliest allowed arrival time

Louis et al. (1999) suggest using the earliest allowed arrival time, given by e_i in Table 6.2, to establish the precedence list, denoted by MX_{e_i}. Executing their recommendation results in a precedence list $P = [4\ 6\ 9\ 10\ 2\ 8\ 3\ 5\ 7\ 1]$. When using the precedence list on parents A and B from Figure 6.1, an offspring chromosome $C = [5\ 6\ 9\ 4\ 1\ 2\ 10\ 8\ 3\ 7]$ results.

Time window compatibility

Two novel ways of establishing a precedence list are suggested in this thesis. In the first novel approach denoted by MX_{twc}, the total compatibility for each customer is calculated using either (4.8) or (4.9), sorted in ascending order to create the precedence list. Ties are broken arbitrarily. The resulting precedence list sees incompatible nodes placed earlier in the chromosome. More compatible nodes are subsequently inserted to fill routes and tours.

Angles

The second novel way to establish the precedence list is to reconsider the fundamental way in which the the crossover operator is used. The simplicity, yet success of the sweep algorithm proposed by Gillett and Miller (1974) is incorporated in this MX approach denoted by $MX_{\angle}$. The angle for each customer is calculated, and the angles are sorted in ascending order to determine the precedence list. The resulting crossover ensures that customers that are located close to one another are assigned to the same route, time windows permitting.

With the depot's location indicated by an open circle in Figure 6.2, the precedence list

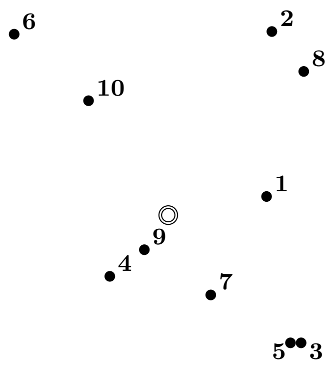


Figure 6.2: Depot and first 10 customers from the C2-2-2 problem set

$P = [1\ 8\ 2\ 10\ 6\ 9\ 4\ 7\ 5\ 3]$ is established.

6.4.3 Partially Matched Crossover (PMX)

PMX is a genetic operator often used with TSP problems using integer string representation (Goldberg and Lingle, 1985). The operator selects two parent chromosomes using the biassed roulette wheel, and produces two offspring chromosomes, as opposed to the previous operators producing only a single offspring. Consider again the two parent chromosomes A and B given in Figure 6.1. Two crossing positions a and b are randomly selected such that

$1 \leq a < b \leq \|A\| + 1$, where $\|A\|$ denotes the number of elements (alleles) in chromosome A. For illustrative purposes let $a = 3$ and $b = 7$. To create offspring 1, denoted by C_1, the strings between the crossing positions from parent 2 is copied to C_1.

$$C_1 = [\star \star 9\ 1\ 4\ 2 \star \star \star \star]$$

For each element in A between a and b, starting from position a, look for elements in A that have not been copied to C_1. In the example element 3 is identified. Element 3's position in A is occupied by element 9 in C_1, and hence element 9 in A's position is filled in C_1 with 3 such that

$$C_1 = [\star \star 9\ 1\ 4\ 2 \star \star 3 \star].$$

Next, element 5 in A is identified, as element 4 has already been copied to offspring C_1. Element 5's position in A is occupied by 4 in C_1, but since element 4 in A's position is already occupied in C_1 by element 1, element 1's position in A is identified for element 5 in C_1 such that

$$C_1 = [5 \star 9\ 1\ 4\ 2 \star \star 3 \star].$$

Element 6 in A is identified next. Element 6's position in C_1 is occupied by element 2, which in turn, is located in position 2 in A. Hence, element 6 is placed in position 2 in C_1 such that

$$C_1 = [5\ 6\ 9\ 1\ 4\ 2 \star \star 3 \star].$$

As all elements in A between positions a and b have been considered, C_1 is completed by duplicating the remaining elements from A such that

$$C_1 = [5\ 6\ 9\ 1\ 4\ 2\ 7\ 8\ 3\ 10].$$

The second offspring, denoted by C_2, is created in a similar fashion with the resulting offspring being $C_2 = [1\ 2\ 3\ 4\ 5\ 6\ 10\ 8\ 9\ 7]$.

6.5 Evaluating crossover operators

The proposed GA algorithm is executed to identify appropriate crossover operators for the varying problem sets. Due to computational time complexity, a single problem is randomly selected from each problem set. Each crossover operator is then tested using 4 independent iterations. The fitness is calculated using an objective function which considers total travel time, number of vehicles used, and total lateness at customers. The GA is executed for a maximum of 200 generations, each having 100 chromosomes. Of every new generation, 80% of chromosomes were generated through crossover operators. Initially 10% of a newly created generation is established through mutation to ensure that the solution space is widely searched. A non-uniform mutation rate introduced by Michalewicz (1992) reduces the number of mutated chromosomes as the number of generations increases. Hence the solution space is only locally searched towards the end of the algorithm. The balance of a new generation is created by cloning (copying exactly) the best chromosomes from the previous generation.

Figure 6.3 illustrates the performance of the various crossover mechanisms for each problem set. The performances are expressed and calibrated according to the best crossover operators for the specific problem set. Actual results are provided in Tables 6.3a and 6.3b, providing the best fitness (objective function value) obtained over the four independent runs, as well as the average time required (in seconds) to find a solution.

Contrary to Blanton and Wainwright (1993) claiming that most of their MX operators outperform the PMX operator, the results in this thesis indicates six instances in which the PMX is either significantly better in terms of fitness, or significantly faster than any of the other crossover operators. In only two instances, $c2\_2\_3$ and $rc2\_3\_8$, did MX prove significantly faster than the other crossover operators, of which one instance is the newly proposed MX_{twc}.

Using a standard statistical t-test, the EER crossover operators proved to be consistently worse and slower than other mechanisms, and is consequently omitted from further analysis. The remaining operators are again subjected to a t-test, resulting in some operators to be identified as significant, hence labels (e) and (f) in Tables 6.3a and 6.3b.

Self regulation can be achieved through a biased selection of operators based on past performance. Initially each operator (except EER) is assigned equal probability of being selected. A parameter λ indicates the frequency (in terms of generations) of testing *all* operators, and the probabilities are consequently adjusted based on the relative performance of each operator, similar to the self organizing mechanism proposed for exchange operators of the TS algorithm in Chapter 5.

Table 6.3a: Analysis of random problems for each data set

Problem			**EER**	$\mathbf{MX}_{l_i}$	$\mathbf{MX}_{e_i}$	$\mathbf{MX}_{\mathbf{twc}}$	$\mathbf{MX}_{\angle}$	**PMX**
$c1\_2\_8$	**Fitness**	Value	90026	88895	88510	85272	85043	84607
		Relative	1.064	1.051	1.046	1.008	1.005	1.000
		t-Value	-3.096[a]	-1.916	-1.514	1.865	2.104	2.559
	Time	Value	26478	25708	25747	25776	25560	25361
		Relative	1.044	1.014	1.015	1.016	1.008	1.000
		t-Value	-4.569[b]	0.412	0.160	-0.028	1.369	2.656[c]
$c2\_2\_3$	**Fitness**	Value	213070	212736	212572	212318	212261	212078
		Relative	1.005	1.003	1.002	1.001	1.001	1.000
		t-Value	-3.821[b]	-1.559[e]	-0.448	1.272	1.658	2.898[c]
	Time	Value	24498	23783	23695	23416	23595	23657
		Relative	1.046	1.016	1.012	1.000	1.008	1.010
		t-Value	-4.725[b]	-0.059	0.516	2.336[f]	1.168	0.764
$r1\_2\_1$	**Fitness**	Value	37147	36779	36358	36144	35610	34350
		Relative	1.081	1.071	1.058	1.052	1.037	1.000
		t-Value	-2.672[a]	-1.764	-0.724	-0.196	1.123	4.234[d]
	Time	Value	20348	19746	19736	19697	19658	19441
		Relative	1.047	1.016	1.015	1.013	1.011	1.000
		t-Value	-4.650[b]	0.201	0.282	0.596	0.911	2.659[c]

[a] Rejected with 97.5% certainty

[b] Rejected with 99.0% certainty

[c] Accepted with 97.5% certainty

[d] Accepted with 99.0% certainty

[e] Rejected with 97.5% certainty, **EER** omitted

[f] Accepted with 97.5% certainty, **EER** omitted

Table 6.3b: Analysis of random problems for each data set

Problem			EER	MX_{l_i}	MX_{e_i}	MX_{twc}	$MX_{\angle}$	PMX
$r2\_2\_5$	**Fitness**	Value	51472	50715	50392	50147	49975	50277
		Relative	1.030	1.015	1.008	1.003	1.000	1.006
		t-Value	-4.434[b]	-0.994[e]	0.474	1.588	2.369	0.997
	Time	Value	17685	17160	17279	17374	17612	17003
		Relative	1.040	1.009	1.016	1.022	1.036	1.000
		t-Value	-3.113[a]	1.797	0.684	-0.204	-2.430[e]	3.266[c]
$rc1\_2\_8$	**Fitness**	Value	42833	42310	41693	41327	41257	40772
		Relative	1.051	1.038	1.023	1.014	1.012	1.000
		t-Value	-3.679[b]	-1.983[e]	0.018	1.205	1.432	3.005[c]
	Time	Value	61481	60192	60236	60007	59988	60215
		Relative	1.025	1.003	1.004	1.000	1.000	1.004
		t-Value	-4.908[b]	0.701	0.510	1.507	1.589	0.601
$rc2\_2\_8$	**Fitness**	Value	37348	34763	34433	34294	34271	34045
		Relative	1.097	1.021	1.011	1.007	1.007	1.000
		t-Value	-4.909[b]	0.189[e]	0.840	1.114	1.160	1.605
	Time	Value	46954	46193	46561	46686	46605	47711
		Relative	1.016	1.000	1.008	1.011	1.009	1.033
		t-Value	-0.803	2.813[c]	1.064	0.470	0.855	-4.400[b]

[a] Rejected with 97.5% certainty

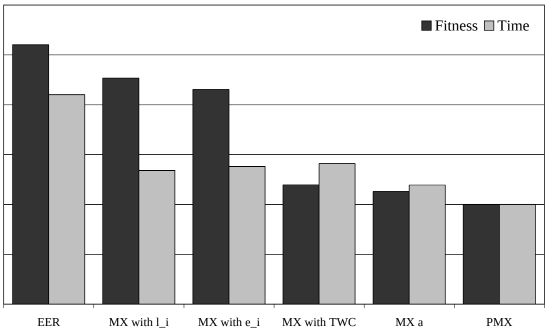
[b] Rejected with 99.0% certainty

[c] Accepted with 97.5% certainty

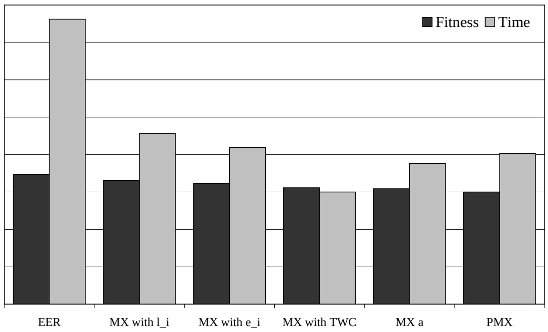
[d] Accepted with 99.0% certainty

[e] Rejected with 97.5% certainty, **EER** omitted

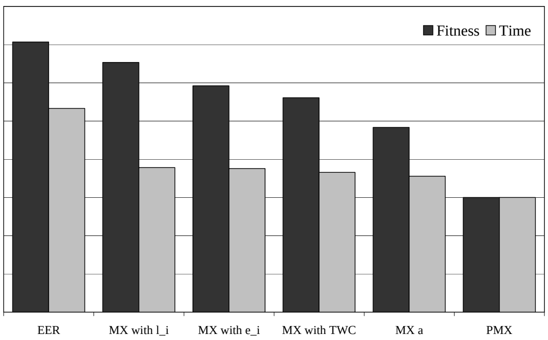
[f] Accepted with 97.5% certainty, **EER** omitted



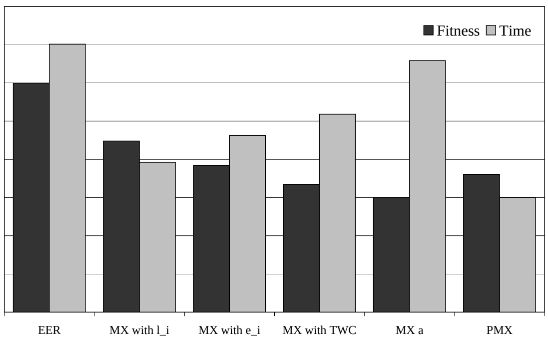
(a) $c1\_2\_8$



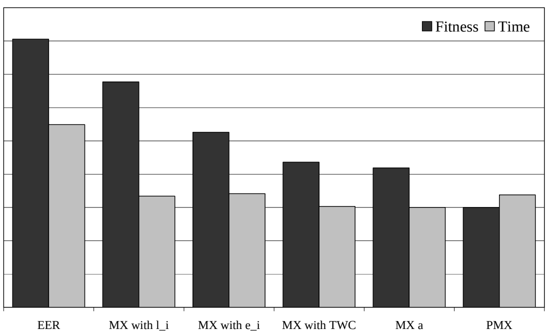
(b) $c2\_2\_3$



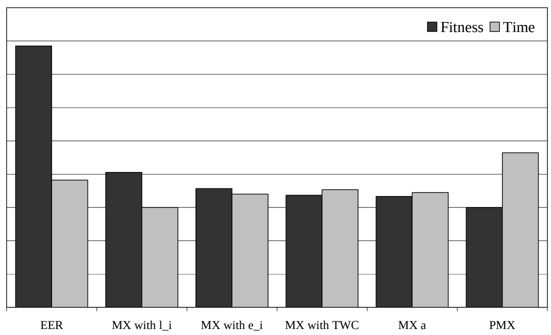
(c) $r1\_2\_1$



(d) $r2\_2\_5$



(e) $rc1\_2\_8$



(f) $rc2\_2\_8$

Figure 6.3: Results for a random problem from each set, expressed relative to the best crossover mechanism for each set.

6.6 Conclusion

In this chapter a GA with integer string representation is developed to test a variant of the VRP that uses time-dependent travel time and that accommodates time windows, a heterogenous fleet, and multiple scheduling. Six crossover mechanisms are tested, two of which are newly proposed in this thesis.

The results suggest that although there are performance differences among the crossover operators, few prove to be significant. Therefor, it is suggested that when integrating the multiple optimization algorithms, namely GA and TS, into the intelligent routing agent, internal learning or self regulation should be considered.

Chapter 7

Clustering input data

In this chapter the concept of pattern identification on input data is investigated. What is peculiar about the benchmark problem sets proposed by both Solomon (1987) and Homberger and Gehring (1999) are the fact that they are preempting specific theoretical characteristics, unlike problems found in real applications. This is clearly illustrated when the assignment of time windows is discussed. For the problem sets $R1$, $R2$, $RC1$, and $RC2$ a percentage of customers are selected to receive time windows, say $0 < f \leq 1$. Next n random numbers from the random uniform distributions is generated on the interval $(0, 1)$, and sorted. Customers $i_1, i_2, \ldots, i_{n_1}$ are then assigned time windows, where the number of customers requiring time windows can then be approximated by $n_1 \approx f.n$. The center of the time window for customer $i_j \in \{i_1, i_2, \ldots, i_{n_1}\}$ is a uniformly distributed, randomly generated number on the interval $\left(e_0 + t_{oi_j}, l_0 - t_{i_j 0}\right)$, where e_0 and l_0 denotes the opening and closing times of the depot, respectively, and t_{0i_j} and $t_{i_j 0}$ denotes the travel distance from the depot to customer i_j, and back, respectively.

For clustered problem sets $C1$ and $C2$ the process becomes questionable. Customers in each cluster are first *routed* using a *3-opt* routine as described in the previous chapter. An orientation is chosen for the route, and time windows are then assigned with the center being the arrival time at the customer. The width and density are derived in a similar fashion as for random and semi-clustered data. Although Solomon (1987) states that *"this approach permits the identification of a very good, possibly optimal, cluster-by-cluster solution which, in turn, provides an additional means of evaluating heuristic performance"*, it does not provide a credible means to evaluate real life problems where customers do not negotiate their sequence prior to stating a preferred time window.

Literature provides good references to what type of metaheuristics, or metaheuristic

Bibliography

Aarts, E. and Korst, J. (1989). *Simulated Annealing and Boltzmann Machines.* Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, Chichester.

Acid, S., de Campos, L. M., Fernández-Luna, J. M., Rodríguez, S., Rodríguez, J. M., and Salcedo, J. L. (2004). A comparison of learning algorithms for bayesian networks: a case study on data from an emergency medical service. *Artificial Intelligence in Medicine*, 30(3):215–232.

Ahn, B.-H. and Shin, J.-Y. (1991). Vehicle-routeing with time windows and time-varying congestion. *The Journal of the Operational Research Society*, 42(5):393–400.

Albus, J. S. (1999). The engineering of mind. *Information Sciences*, 117(1):1–18.

Alfa, A. S., Heragu, S. S., and Chen, M. (1991). A 3-opt ased simulated annealing algorithm for vehicle routing problems. *Computers & Industrial Engineering*, 21(1–4):635–639.

Assad, A. A. (1988). Modeling and implementation issues in routing. In Golden, B. L. and Assad, A. A., editors, *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*, chapter 2, pages 7–45. North-Holland, Amsterdam.

Banister, D. (1995). *Transport and urban development.* E & FN Spon, London, 1^{st} edition.

Beaulieu, M. and Gamache, M. (2006). An enumeration algorithm for solving the fleet management problem in underground mines. *Computers & Operations Research*, 33(6):1606–1624.

Bezdek, J. C. (1974). Numerical taxonomy with fuzzy sets. *Journal of Mathematical Biology*, 1:57–71.

Birge, J. R. and Louveaux, F. V. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392.

Blanton, J. L. and Wainwright, R. L. (1993). Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 452–459, San Francisco, CA. Morgan Kaufmann Publishers.

Bodin, L., Golden, B. L., Assad, A., and Ball, M. O. (1983). The state of the art in the routing and scheduling of vehicles and crews. *Computers & Operations Research*, 10(1):63–211.

Bolshakova, N. and Azuaje, F. (2003). Cluster validation techniques for genome expression data. *Signal Processing*, 83(4):825–833.

Brandão, J. and Mercer, A. (1997). A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100(1):180–191.

Bräysy, O. and Gendreau, M. (2001). Tabu search heuristics for the vehicle routing problem with time windows. Report stf42 a01022, SINTEF Applied Mathematics, Research Council of Norway.

Brucker, P. (2004). *Scheduling algorithms*. Springer-Verlag, Berlin, Germany, 4$^{\text{th}}$ edition.

Bullnheimer, B., Hartl, R. F., and Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89:319–328.

Burke, L. I. and Ignizio, J. P. (1992). Neural networks and operations research: An overview. *Computers & Operations Research*, 19(3–4):179–189.

Butt, S. E. and Ryan, D. M. (1999). An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, 26(4):427–441.

Caianello (1961). Outline of theory of though-processes and thinking machines. *Journal of Theoretical Biology*, 1(2):204–235.

Carter, A. E. and Ragsdale, C. T. (2005). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research*. Forthcoming.

Chen, X., Wan, W., and Xu, X. (1998). Modeling rolling batch planning as vehicle routing problem with time windows. *Computers & Operations Research*, 25(12):1127–1136.

Choi, E. and Tcha, D.-W. (2006). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*. Forthcoming.

Christofides, N. (1985). Vehicle routing. In Lawler, E., Lenstra, J., Rinnooy Kan, A., and Shmoys, D., editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, chapter 12, pages 431–448. John Wiley & Sons, UK.

Christofides, N., Mingozzi, A., and Toth, P. (1979). The vehicle routing problem. In Christofides, N., Mingozzi, A., Toth, P., and Sandi, C., editors, *Combinatorial Optimization*, chapter 11, pages 315–338. Wiley-Interscience, New York.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581.

CSIR Transportek (2004). The First State of Logistics Survey for South Africa.

Dermuth, H., Beale, M., and Hagan, M. (2005). *Neural Network Toolbox User's Guide*. The Mathworks, Natick, MA.

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

Desrosiers, J., Sauvé, M., and Soumis, F. (1988). Lagrangian relaxation methods for solving the minimum fleet size multiple traveling salesman problem with time windows. *Management Science*, 34(8):1005–1022.

Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2–3):243–278.

Dorigo, M., Di Caro, G., and Gambardelle, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172.

Dorigo, M. and Gambardella, L. M. (1997a). Ant colonies for the travelling salesman problem. *BioSystems*, 43(2):73–81.

Dorigo, M. and Gambardella, L. M. (1997b). Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M. and Stützle, T. (2002). *Ant Colony Optimization*. MIT Press, Cambridge.

Dullaert, W., Janssens, G. K., Sörensen, K., and Vernimmen, B. (2001). New heuristics for the fleet size and mix vehicle routing problem with time windows. In *9th World Conference on Transport Research, July 22–27, 2001*, COEX Convention Center, Seoul.

Filipec, M., Skrlec, D., and Krajcar, S. (1997). Darwin meets computers: new approach to multiple depot capacitated vehicle routing problem. In *Computational Cybernetics and Simulation*, volume 1 of *IEEE International Conference on Systems, Man, and Cybernetics*, pages 421–426, California. Systems, Man, and Cybernetics Society of the Institute of Electrical and Electronic Engineers, Inc, IEEE.

Filipec, M., Skrlec, D., and Krajcar, S. (1998). An efficient implementation of genetic algorithms for constrained vehicle routing problem. In *Intelligent Systems for Humans in a Cyberworld*, volume 3 of *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2231–2236, Florida. Systems, Man, and Cybernetics Society of the Institute of Electrical and Electronic Engineers, Inc, IEEE.

Fisher, M. L. and Jaikumar, R. (1981). A general assignment heuristic for vehicle routing. *Networks*, 11:109–124.

Fleischmann, B., Gietz, M., and Gnutzmann, S. (2004). Time-varying travel times in vehicle routing. *Transportation Science*, 38(2):160–173.

Gambardella, L. M., Taillard, E., and Agazzi, G. (1999). MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 63–76. McGraw-Hill, London.

Gath, I. and Geva, A. B. (1989). Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–781.

Gehring, H. and Homberger, J. (1999). A paralel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In Miettinen, K. and Mäkelä, M. M., editors, *Proceedings of EUROGEN99 — Short Course on Evolutionary Algorithms in Engineering and Computer Science*, volume No. A 2/1999 of *Reports of the Department of Mathematical Information Technology*, pages 57–64, Finland.

Gendreau, M. (2003). An introduction to tabu search. In Glover, F. and Kochenberger, G. A., editors, *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, chapter 2, pages 37–54. Kluwer Academic Publishers, Boston.

Gendreau, M., Laporte, G., Musaraganyi, C., and Taillard, É. D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 26(12):1153–1173.

Gendreau, M., Laporte, G., and Potvin, J.-Y. (1998). Metaheuristics for the vehicle routing problem. Report g-98-52, Les Cahiers du GERAD.

Gendreau, M., Laporte, G., and Potvin, J.-Y. (2002). Metaheuristics for the capacitated VRP. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, chapter 6, pages 129–154. Society for Industrial and Applied Mathematics (SIAM), Philadelphia.

Ghiani, G., Guerriero, F., Laporte, G., and Musmanno, R. (2003). Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11.

Giaglis, G., Minis, I., Tatarakis, A., and Zeimpekis, V. (2004). Minimizing logistics risks through real-time vehicle routing and mobile technologies. *International Journal of Physical Distribution & Logistics Management*, 34(9):749–764.

Gillett, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Glover, F. (1990). Tabu search — part II. *ORSA Journal on Computing*, 2(1):4–32.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts.

Goldberg, D. E. and Lingle, R. (1985). Alleles, loci, and the TSP. In Grefenstette, J. J., editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 154–159, Hillsdale, N.J. Lawrence Erlbaum Associates.

Golden, B., Assad, A., Levy, L., and Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1):49–66.

Hamacher, A., Hochstättler, W., and Moll, C. (2000). Tree partitioning under constraints — clustering for vehicle routing problems. *Discrete Applied Mathematics*, 99(1):55–69.

Hill, A., Mabert, V., and Montgomery, D. (1988). A decision support system for the courier vehicle scheduling problem. *Omega, International Journal of Management Science*, 16(4):333–345.

Hill, A. V. and Benton, W. (1992). Modelling intra-city time-dependent travel speeds for vehicle scheduling problems. *The Journal of the Operational Research Society*, 43(4):343–351.

Hillier, F. S. and Lieberman, G. J. (2005). *Introduction to Operations Research*. McGraw-Hill, New York, 8^{th} edition.

Homberger, J. (2003). Extended Solomon's VRPTW instances. World wide web at *http://www.fernuni-hagen.de/WINF/touren/inhalte/probinst.htm*.

Homberger, J. and Gehring, H. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR*, 37(3):297–318.

Hopfield, J. J. and Tank, D. W. (1985). "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152.

Hwang, H.-S. (2002). An improved model for vehicle routing problem with time constraint based on genetic algorithm. *Computers & Industrial Engineering*, 42(2–4):361–369.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., and Yagiura, M. (2005). Effective local search algorithms for routing and scheduling problems with general time window constraints. *Transportation Science*, 39(2):206–232.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396.

Jeroslow, R. (1979). The theory of cutting-planes. In Christofides, N., Mingozzi, A., Toth, P., and Sandi, C., editors, *Combinatorial Optimization*, chapter 2, pages 21–72. Wiley-Interscience, New York.

Joubert, J. W. (2003). An initial solution heuristic for the vehicle routing and scheduling problem. Master's thesis, Industrial and Systems Engineering, University of Pretoria, South Africa.

Joubert, J. W. and Claasen, S. J. (2006). A sequential insertion heuristic for the initial solution of a constrained vehicle routing problem. *ORiON*, 22(1):105–116.

Kall, P. and Wallace, S. (1994). *Stochastic Programming*. John Wiley & Sons, 1^{st} edition.

Kara, I. and Bektas, T. (2005). Integer linear programming formulations of multiple salesman problems and its variations. *European Journal of Operational Research*. Forthcoming.

Karanta, I., Mikkola, T., Bounsaythip, C., Jokinen, O., and Savlova, J. (1999). Genetic algorithms applied to a wood collection problem. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 635–639.

Karayannis, N. B. and Venetsanopoulos, A. N. (1993). *Artificial Neural Networks: Learning algorithms, performance evaluation, and aplications*. Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, Boston, Massachusetts.

Kim, D.-W., Lee, K. H., and Lee, D. (2003). Fuzzy cluster validation index based on inter-cluster proximity. *Pattern Recognition Letters*, 24(15):2561–2574.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimisation by simulated annealing. *Science*, 20:671–680.

Koskosidis, Y. A., Powell, W. B., and Solomon, M. M. (1992). An optimization-based heuristic for vehicle routing and scheduling with soft time windows. *Transportation Science*, 26(2):69–85.

Kwon, S. (1998). Cluster validity index for fuzzy clustering. *Electronic Letters*, 34(22):2176–2177.

Lambert, V., Laporte, G., and Louveaux, F. (1993). Designing collection routes through bank branches. *Computers & Operations Research*, 20(7):783–791.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358.

Laporte, G., Louveaux, F., and Mercure, H. (1989). Models and exact solutions for a class of stochastic location-routing problems. *European Journal of Operational Research*, 39(1):71–78.

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

Laporte, G. and Louveaux, F. V. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142.

Laporte, G., Mercure, H., and Nobert, Y. (1986). An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16(1):33–46.

Laporte, G. and Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. In Martello, S., Laporte, G., Minoux, M., and Ribeiro, C., editors, *Sureys in Combinatorial Optimization*, volume 31 of *Annals of Discrete Mathematics*, chapter 5, pages 147–184. Elsevier Science (North-Holland), Amsterdam.

Lee, C.-G., Epelman, M. A., White III, C. C., and Bozer, Y. A. (2006). A shortest path approach to the multiple-vehicle routing problem with split pickups. *Transportation Research Part B*, 40(4):265–284.

Leinbach, P. and Stansfield, T. (2002). Living up to expectations. *IE Solutions*, 34(11):24–30.

Lenstra, J. and Rinnooy Kan, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227.

Li, C.-L., Vairaktarakis, G., and Lee, C.-Y. (2005). Machine scheduling with deliveries to multiple customer locations. *European Journal of Operational Research*, 164(1):39–51.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44:2245–2269.

Liu, F.-H. and Shen, S.-Y. (1999a). The fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, 50(7):721–732.

Liu, F.-H. and Shen, S.-Y. (1999b). A method for Vehicle Routing Problem with Multiple Vehicle Types and Time Windows. *Proceedings of the National Science Council, Republic of China, ROC(A)*, 23(4):526–536.

Louis, S. J., Yin, X., and Yuan, Z. Y. (1999). Multiple vehicle routing with time windows using genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 1804–1808, Washington, D.C. IEEE.

Maeda, O., Nakamura, M., Ombuki, B. M., and Onaga, K. (1999). A genetic algorithm approach to vehicle routing problem with time deadlines in geographical information systems. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 595–600. IEEE.

Maffioli, F. (1979). The complexity of combinatorial optimization algorithms and the challenge of heuristics. In Christofides, N., Mingozzi, A., Toth, P., and Sandi, C., editors, *Combinatorial Optimization*, chapter 5, pages 107–129. Wiley-Interscience, New York.

Malandraki, C. and Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200.

Malmborg, C. J. (1996). A genetic algorithm for service level based vehicle scheduling. *European Journal of Operational Research*, 93(1):121–134.

Maniezzo, V., Gambardella, L. M., and de Luigi, F. (2004). Ant colony optimization. In Onwubolu, G. C. and Babu, B. V., editors, *New Optimization Techniques in Engineering*, chapter 5, pages 101–117. Springer-Verlag, Berlin.

Matsuyama, Y. (1991). Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. In *IEEE International Joint Conference on Neural Networks*, pages 385–390.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Meuleau, N. and Dorigo, M. (2002). Ant colony optimization and stochastic gradient decent. *Artificial Life*, 8(2):103–121.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Artificial Intelligence. Springer-Verlag, Berlin.

Middendorf, M., Reischle, F., and Schmeck, H. (2002). Multi colony ant algorithms. *Journal of Heuristics*, 8(3):305–320.

Mole, R. H. and Jameson, S. R. (1976). A sequential route-bulding algorithm employing a generalised savings criterion. *Operational Research Quarterly*, 27(2):503–511.

Müller, B., Reinhardt, J., and Strickland, M. T. (1995). *Neural Networks: An Introduction*. Physics of Neural Networks. Springer-Verlag, Berlin, 2$^{\text{nd}}$ edition.

Nagy, G. and Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162:126–141.

Nelson, M. D., Nygard, K. E., Griffin, J. H., and Schreve, W. E. (1985). Implementation techniques for the vehicle routing problem. *Computers & Operations Research*, 12(3):273–283.

Nygard, K. E., Greenberg, P., Bolkan, W. E., and Swenson, E. J. (1988). General assignment methods for the deadline vehicle routing problem. In Golden, B. L. and Assad, A. A., editors, *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*, chapter 6, pages 107–125. North-Holland, Amsterdam.

Nygard, K. E. and Kadaba, N. (1991). Algorithm management using genetic search for computer-aided vehicle routing. In 24^{th} *Annual Hawaii International Conference on Systems Sciences*, volume 3, pages 317–326.

Ochi, L. S., Vianna, D. S., Drummond, L. M. A., and Victor, A. O. (1998). A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet. *Future Generation Computer Systems*, 4(5–6):285–292.

OECD (2003). *Delivering the Goods: 21^{st} Century Challenges to Urban Goods Transport*. OECD, Paris, France.

Ong, H., Ang, B., Goh, T., and Deng, C. (1997). A vehicle routing and scheduling problem with time windows and stochastic demand constraints. *Asia Pacific Journal of Operational Research*, 14(1):1–17.

Padberg, M. and Rinaldi, G. (1987). Optimization of a 532-city symmetric travelling salesman problem by branch and cut. *Operations Research Letters*, 6(1):1–7.

Paessens, H. (1988). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research*, 34:336–344.

Pal, N. R. and Bezdek, J. C. (1995). On cluster validity for the fuzzy c-means model. *IEEE Transactions on Fuzzy Systems*, 3(3):370–379.

Potvin, J.-Y. (1993). The traveling salesman problem: A neural network perspective. *ORSA Journal on Computing*, 5(4):328–348.

Potvin, J.-Y. and Smith, K. A. (2003). Artificial neural networks for combinatorial optimization. In Glover, F. and Kochenberger, G. A., editors, *The handbook of metaheuristics*, International series in Operations Research & Management Science, chapter 15, pages 429–455. Kluwer Academic Publishers, Boston, Massachusetts.

Potvin, J.-Y., Xu, Y., and Benyahia, I. (2006). Vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 33(4):1129–1137.

Powell, W. B. (2003). Dynamic models of transportation operations. In de Kok, A. and Graves, S. C., editors, *Supply Chain Management: Design, Coordination and Operation*, volume 11 of *Handbooks in Operations Research and Management Science*, chapter 13, pages 677–756. Elsevier, Amsterdam, Netherlands.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002.

Rao, A. R. and Srinivas, V. V. (2006). Regionalization of watersheds by fuzzy cluster analysis. *Journal of Hydrology*, 318(1):57–79.

Rardin, R. (1998). *Optimization in Operations Research*. Prentice Hall, Upper Saddle River, New Jersey.

Reimann, M., Doerner, K., and Hartl, R. F. (2004). D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591.

Rezaee, M. R., Lelieveldt, B. P. F., and Reiber, J. H. C. (1998). A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*, 19(3–4):237–246.

Righini, G. and Salani, M. (2004). Dynamic programming algorithms for the elementary shortest path problem with resource constraints. *Electronic Notes in Discrete Mathematics*, 17:247–249.

Robusté, F., Daganzo, C. F., and Souleyrette, R. R. I. (1990). Implementing vehicle routing models. *Transportation Research Part B*, 24(4):263–286.

Rochat, Y. and Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(2):147–167.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan Books, Washington, D.C.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, New Jersey, 2$^{\text{nd}}$ edition.

Salhi, A., Sari, M., Saidi, D., and Touati, N. A. C. (1992). Adaption of some vehicle fleet mix heuristics. *OMEGA International Journal of Management Science*, 20(5–6):653–660.

Salhi, S. and Rand, G. K. (1993). Incorporating vehicle routing into the vehicle fleet composition problem. *European Journal of Operational Research*, 66(3):313–330.

Schrage, L. (2002). *Optimization modeling with LINGO*. LINDO Systems Inc, Illinois, 5^{th} edition.

Skrlec, D., Filipec, M., and Krajcar, S. (1997). A heuristic modification of genetic algorithm used for solving the single depot capacitated vehicle routing problem. In *Intelligent Information Systems '97*, pages 184–188. IEEE.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time windows. *Operations Research*, 35(2):254–265.

Spence, M. (1998). Western Cape Provincial Transport Policy. In Freeman, P. and Jamet, C., editors, *Urban transport policy — a sustainable development tool*, Rotterdam. CODATU, A.A. Balkema.

Starkweather, T., McDaniel, S., Mathias, K., Whitley, D., and Whitley, C. (1991). A comparison of genetic sequencing operators. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference of Genetic Algorithms*. San Mateo, Kaufmann.

Taha, H. (2003). *Operations research: an introduction*. Pearson Education, Inc., Upper Saddle River, New Jersey, 7^{th} edition.

Taillard, É. D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–673.

Taillard, É. D. (1999). A heuristics column generation method for the heterogeneous fleet VRP. *Operations Research – Recherche opérationnelle*, 33:1–14.

Taillard, É. D., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186.

Taillard, É. D., Laporte, G., and Gendreau, M. (1996). Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society*, 47(8):1065–1070.

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

Tan, K. C., Lee, L. H., Ou, K., and Lee, L. H. (2001b). A messy genetic algorithm for the vehicle routing problem with time window constraints. In *Congress on Evolutionary Computation*, volume 3, pages 679–686.

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

Taniguchi, E., Thompson, R. G., and Yamada, T. (2004). Visions for city logistics. In Taniguchi, E. and Thompson, R. G., editors, *Logistics Systems for Sustainable Cities*, pages 1–16, Oxford, UK. Institute for City Logistics, Elsevier Ltd.

Thangiah, S. R. and Gubbi, A. V. (1993). Effect of genetic sectoring on vehicle routing problems with time windows. In *IEEE International Conference on Developing and Managing Intelligent System Projects*, pages 146–153. IEEE.

Thangiah, S. R., Nygard, K. E., and Juell, P. L. (1991). GIDEON: A genetic algorithm system for vehicle routing with time windows. In *Seventh IEEE Conference on Artificial Intelligence for Applications*, volume 1, pages 322–328. IEEE.

Thompson, P. M. and Psaraftis, H. N. (1993). Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research*, 41(5):935–946.

TOP Program (2006). VRP: A bibliography. Retrieved online from `http://www.sintef.no/static/am/opti/projects/top/vrp/bibliography.html`, January.

Toth, P. and Vigo, D. (2002a). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1–3):487–512.

Toth, P. and Vigo, D. (2002b). An overview of vehicle routing problems. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, chapter 1, pages 1–26. Society for Industrial and Applied Mathematics (SIAM), Philadelphia.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333–346.

Tung, D. V. and Pinnoi, A. (2000). Vehicle routing-scheduling for waste collection in hanoi. *European Journal of Operational Research*, 125(3):449–468.

Van Breedam, A. (1995). Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86(5):480–490.

Van Breedam, A. (2001). Comparing descent heuristics and metaheuristics for the vehicle routing problem. *Computers & Operations Research*, 28(4):289–315.

Van Slyke, R. M. and Wets, R. (1969). l-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.

Vas, P. (1999). *Artificial-intelligence-based electrical machines and drives: application of fuzzy, neural, fuzzy-neural, and genetic-algorithm-based techniques.* Oxford University Press, Oxford.

Winston, W. and Venkataramanan, M. (2003). *Introduction to mathematical programming*, volume 1 of *Operations Research*. Brooks/Cole - Thomson Learning, Pacific Grove, CA, 4^{th} edition.

Xie, X. L. and Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847.

Xu, Y. and Brereton, R. G. (2005). A comparative study of cluster validation indices applied to genotyping data. *Chemometrics and intelligent laboratory systems*, 78(1–2):30–40.

Yamada, T. and Taniguchi, E. (2005). Modelling the effects of urban freight transport schemes. In Taniguchi, E. and Thompson, R. G., editors, *City Logistics*, pages 75–89, Kyoto, Japan. Institute for City Logistics, Institute for City Logistics.

Zhu, K. Q. (2003). A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows. In *IEEE International Conference on Tools with Artificial Intelligence*, pages 176–183. IEEE.

Appendix A

Initial solution sample

```
Output for the data set: r2_2_3

Tour:  1, v-type:  1, v-cap:  300
41-23-18-96-88-43-19-59-92-47-77-61-75-67-99-85-70-80

Tour:  2, v-type:  1, v-cap:  300
34-74-33-62-36-63-3-12-4-87-25-73-57-5-45-78
56-65-9-49-81-51-37-1-72-39

Tour:  3, v-type:  1, v-cap:  300
30-83-50-26-11-91-66-64-79-17-89-58-16-14-100-82-2-94-35-28

Tour:  4, v-type:  1, v-cap:  300
60-68-21-40-6-27-32-13-95-38-8-90-84-98-93-10-46

Tour:  5, v-type:  1, v-cap:  300
42-31-15-71-44-48-69-7-55-22-54-52

Tour:  6, v-type:  1, v-cap:  300
29-76-20-24-86-53

Orphans:
```

Appendix B

TS results for benchmark data sets

Table B.1: TS results for benchmark data sets

(a) Problem set $c1$

	Run					Average
Problem	**1**	**2**	**3**	**4**	**Average**	**time (sec)**
$c1\_2\_1$	285935	288999	286108	289065	287526	1630
$c1\_2\_2$	176921	176463	176594	176399	176594	2022
$c1\_2\_3$	172851	172833	172639	169703	172006	2543
$c1\_2\_4$	169016	169138	168894	168966	169003	3167
$c1\_2\_5$	282248	282289	281969	282177	282170	1644
$c1\_2\_6$	79155	79053	79115	79097	79105	1610
$c1\_2\_7$	79694	79625	79383	79290	79498	1697
$c1\_2\_8$	75902	76057	76131	76178	76067	1842
$c1\_2\_9$	72323	69569	72533	72458	71720	2252
$c1\_210$	72151	72053	72101	71813	72029	2871

(b) Problem set $c2$

	Run					Average
Problem	**1**	**2**	**3**	**4**	**Average**	**time (sec)**
$c2\_2\_1$	913477	923921	923694	913411	918625	1559
$c2\_2\_2$	702879	702790	702816	702734	702804	1980
$c2\_2\_3$	592152	592167	592208	592229	592189	2457
$c2\_2\_4$	581243	581579	581364	581137	581330	3072
$c2\_2\_5$	813035	813296	813107	813383	813205	1576
$c2\_2\_6$	213958	224152	214001	213512	216405	1706
$c2\_2\_7$	313295	313603	303183	313426	310876	1775
$c2\_2\_8$	213557	213589	213725	213413	213571	1823
$c2\_2\_9$	302713	292452	302834	302695	300173	1960
$c2\_210$	202794	202801	203140	202838	202893	2102

Table B.1: TS results for benchmark data sets (continued)

(c) Problem set $r1$

	Run					Average
Problem	**1**	**2**	**3**	**4**	**Average**	**time (sec)**
$r1\_2\_1$	26151	26426	26412	26032	26255	1284
$r1\_2\_2$	23877	23526	22399	22581	23095	1609
$r1\_2\_3$	21496	21152	22026	21549	21555	1909
$r1\_2\_4$	20592	20334	20603	20724	20563	2392
$r1\_2\_5$	26200	24840	25698	25621	25589	1333
$r1\_2\_6$	23569	22398	22877	23272	23029	1630
$r1\_2\_7$	22182	21690	20577	22390	21709	1983
$r1\_2\_8$	19778	20256	20011	19646	19922	2524
$r1\_2\_9$	26807	25207	27937	25603	26388	1646
$r1_2 10$	27977	29090	28277	28287	28407	1659

(d) Problem set $r2$

	Run					Average
Problem	**1**	**2**	**3**	**4**	**Average**	**time (sec)**
$r2\_2\_1$	45730	41216	41307	41216	42367	670
$r2\_2\_2$	40831	40808	41089	40920	40912	1106
$r2\_2\_3$	36260	36284	36370	36216	36282	1680
$r2\_2\_4$	30522	30531	30605	30629	30571	2142
$r2\_2\_5$	40858	36465	36540	36465	37582	766
$r2\_2\_6$	31367	36359	36178	36324	35057	1219
$r2\_2\_7$	31097	31143	35720	35880	33460	1536
$r2\_2\_8$	30468	30634	30359	30466	30481	2010
$r2\_2\_9$	35981	35929	36033	36000	35985	880
$r2\_210$	33511	35765	35864	35773	35228	1047

Table B.1: TS results for benchmark data sets (continued)

(e) Problem set $rc1$

	Run					Average
Problem	**1**	**2**	**3**	**4**	**Average**	**time (sec)**
$rc1\_2\_1$	23671	23352	23579	23447	23512	1618
$rc1\_2\_2$	21786	21932	21610	21641	21742	1880
$rc1\_2\_3$	18613	19753	19827	19414	19401	3160
$rc1\_2\_4$	16317	16151	16250	16314	16258	4762
$rc1\_2\_5$	24564	24887	23695	26082	24807	3241
$rc1\_2\_6$	26466	26122	27462	26680	26682	3395
$rc1\_2\_7$	25750	26178	25615	24741	25571	3802
$rc1\_2\_8$	26063	26354	26619	25451	26121	4243
$rc1\_2\_9$	25769	24579	23043	25931	24830	3291
$rc1\_210$	23398	23541	25508	24306	24188	3697

(f) Problem set $rc2$

	Run					Average
Problem	**1**	**2**	**3**	**4**	**Average**	**time (sec)**
$rc2\_2\_1$	23912	24225	24349	23949	24108	1912
$rc2\_2\_2$	20480	20641	20712	20544	20594	2638
$rc2\_2\_3$	18644	18846	18610	18799	18724	3275
$rc2\_2\_4$	17039	17181	17279	17026	17131	4605
$rc2\_2\_5$	23916	23577	23622	23601	23679	2302
$rc2\_2\_6$	24000	24054	23998	23782	23958	2413
$rc2\_2\_7$	25284	23595	25133	24284	24574	2770
$rc2\_2\_8$	28422	26886	26769	27451	27382	3583
$rc2\_2\_9$	29097	28729	28206	28478	28627	3573
$rc2\_210$	26818	25362	26627	25273	26020	4169

Appendix C

GA crossover performance

Table C.1a: Summary of GA results

Set	EER			MX on L_i		
	Min	**Mean**	**Time**	**Min**	**Mean**	**Time**
$c1\_2\_8$	90026	92803	26478	88895	89317	25708
$c2\_2\_3$	213070	215864	24498	212736	212820	23783
$r1\_2\_1$	37147	37441	20348	36779	36898	19746
$r2\_2\_5$	51472	57614	17685	50715	50911	17160
$rc1\_2\_8$	42833	43066	61481	42310	42450	60192
$rc2\_2\_8$	37348	37766	46954	34763	36356	46193

Table C.1b: Summary of GA results (continued)

Set	MX on e_i			MX on $\boldsymbol{TWC}$		
	Min	**Mean**	**Time**	**Min**	**Mean**	**Time**
$c1\_2\_8$	88510	88715	25747	85272	85467	25776
$c2\_2\_3$	212572	212663	23695	212318	212398	23416
$r1\_2\_1$	36358	36493	19736	36144	36206	19697
$r2\_2\_5$	50392	50528	17279	50147	50224	17374
$rc1\_2\_8$	41693	41972	60236	41327	41441	60007
$rc2\_2\_8$	34433	34477	46561	34294	34328	46686

Table C.1c: Summary of GA results (continued)

Set	MX on angles			PMX		
	Min	Mean	Time	Min	Mean	Time
$c1\_2\_8$	85043	85146	25560	84607	84649	25361
$c2\_2\_3$	212261	212272	23595	212078	212132	23657
$r1\_2\_1$	35610	35904	19658	34350	34856	19441
$r2\_2\_5$	49975	50072	17612	50277	53077	17003
$rc1\_2\_8$	41257	41284	59988	40772	41040	60215
$rc2\_2\_8$	34271	34274	46605	34045	34102	47711

Appendix D

Cluster validation results for test sets

Table D.1: Cluster validation results for test set with two clusters

m	V_{PC}	V_{PE}	V_{XB}	V_{XB}^{+}	V_{FS}	K	CWB
1.1	$\boxed{5}$	$\boxed{5}$	2	2	$\boxed{10}$	2	2
1.2	$\boxed{7}$	$\boxed{7}$	2	2	$\boxed{10}$	2	2
1.3	$\boxed{9}$	$\boxed{9}$	2	2	$\boxed{10}$	2	2
1.4	2	2	2	2	$\boxed{10}$	2	2
1.5	2	2	2	2	$\boxed{9}$	2	2
1.6	2	2	2	2	$\boxed{10}$	2	2
1.7	2	2	2	2	$\boxed{10}$	2	2
1.8	2	2	2	2	$\boxed{10}$	2	2
1.9	2	2	2	2	$\boxed{10}$	2	2
2.0	2	2	2	2	$\boxed{10}$	2	2
2.1	2	2	2	2	$\boxed{10}$	2	2
2.2	2	2	2	2	$\boxed{10}$	2	2
2.3	2	2	2	2	$\boxed{10}$	2	2
2.4	2	2	2	2	$\boxed{8}$	2	2
2.5	2	2	2	2	$\boxed{10}$	2	2
2.6	2	2	2	2	$\boxed{10}$	2	2
2.7	2	2	2	2	$\boxed{10}$	2	2
2.8	2	2	2	2	$\boxed{10}$	2	2
2.9	2	2	2	2	$\boxed{10}$	2	2
3.0	2	2	2	2	$\boxed{10}$	2	2
5.0	$\boxed{10}$	2	2	$\boxed{10}$	$\boxed{10}$	2	2
10.0	$\boxed{10}$	2	2	$\boxed{10}$	$\boxed{10}$	2	2
20.0	2	2	2	$\boxed{10}$	$\boxed{4}$	2	2

Table D.2: Cluster validation results for test set with three clusters

m	V_{PC}	V_{PE}	V_{XB}	V_{XB}^{+}	V_{FS}	K	CWB
1.1	$\boxed{7}$	$\boxed{7}$	3	3	$\boxed{10}$	3	3
1.2	$\boxed{8}$	$\boxed{8}$	3	3	$\boxed{10}$	3	3
1.3	$\boxed{10}$	$\boxed{1}0$	3	3	$\boxed{10}$	3	3
1.4	$\boxed{4}$	3	3	3	$\boxed{10}$	3	3
1.5	3	3	3	3	$\boxed{10}$	3	3
1.6	3	3	3	3	$\boxed{10}$	3	3
1.7	3	3	3	3	$\boxed{10}$	3	3
1.8	3	3	3	3	$\boxed{10}$	3	3
1.9	3	3	3	3	$\boxed{10}$	3	3
2.0	3	3	3	3	$\boxed{10}$	3	3
2.1	3	$\boxed{2}$	3	3	$\boxed{10}$	3	3
2.2	3	$\boxed{2}$	3	3	$\boxed{9}$	3	3
2.3	3	$\boxed{2}$	3	3	$\boxed{10}$	3	3
2.4	3	$\boxed{2}$	3	3	$\boxed{10}$	3	3
2.5	3	$\boxed{2}$	3	3	$\boxed{10}$	3	3
2.6	$\boxed{10}$	$\boxed{2}$	3	3	$\boxed{9}$	3	3
2.7	$\boxed{10}$	$\boxed{2}$	3	3	9	3	3
2.8	$\boxed{10}$	$\boxed{2}$	3	3	$\boxed{10}$	3	3
2.9	$\boxed{10}$	$\boxed{2}$	3	3	$\boxed{9}$	3	3
3.0	$\boxed{10}$	$\boxed{2}$	3	3	$\boxed{9}$	3	3
5.0	$\boxed{10}$	$\boxed{2}$	3	$\boxed{10}$	$\boxed{10}$	3	3
10.0	$\boxed{10}$	$\boxed{2}$	3	$\boxed{10}$	$\boxed{10}$	3	3
20.0	$\boxed{2}$	$\boxed{2}$	3	$\boxed{9}$	$\boxed{4}$	3	3

Table D.3: Cluster validation results for test set with four clusters

m	V_{PC}	V_{PE}	V_{XB}	V_{XB}^{+}	V_{FS}	K	CWB
1.1	4	4	4	4	$\boxed{10}$	4	$\boxed{3}$
1.2	4	4	4	4	$\boxed{10}$	4	$\boxed{3}$
1.3	4	4	4	4	$\boxed{10}$	4	4
1.4	4	4	4	4	$\boxed{10}$	4	4
1.5	4	4	4	4	$\boxed{10}$	4	4
1.6	4	4	4	4	$\boxed{10}$	4	4
1.7	4	4	4	4	$\boxed{10}$	4	4
1.8	4	4	4	4	$\boxed{10}$	4	4
1.9	4	4	4	4	$\boxed{10}$	4	4
2.0	4	4	4	4	4	4	4
2.1	4	$\boxed{2}$	4	4	$\boxed{10}$	4	4
2.2	4	$\boxed{2}$	4	4	$\boxed{10}$	4	4
2.3	4	$\boxed{2}$	4	4	4	4	4
2.4	$\boxed{2}$	$\boxed{2}$	4	4	4	4	4
2.5	$\boxed{2}$	$\boxed{2}$	4	4	4	4	4
2.6	$\boxed{2}$	$\boxed{2}$	4	4	$\boxed{10}$	4	4
2.7	$\boxed{2}$	$\boxed{2}$	4	4	$\boxed{10}$	4	4
2.8	$\boxed{10}$	$\boxed{2}$	4	4	$\boxed{10}$	4	$\boxed{2}$
2.9	$\boxed{10}$	$\boxed{2}$	4	4	$\boxed{10}$	4	$\boxed{2}$
3.0	$\boxed{10}$	$\boxed{2}$	4	4	$\boxed{10}$	4	$\boxed{2}$
5.0	$\boxed{10}$	$\boxed{2}$	4	4	$\boxed{10}$	4	$\boxed{3}$
10.0	$\boxed{10}$	$\boxed{2}$	4	4	$\boxed{10}$	4	$\boxed{3}$
20.0	$\boxed{2}$	$\boxed{2}$	4	$\boxed{8}$	4	4	$\boxed{3}$

Table D.4: Cluster validation results for test set with five clusters

m	V_{PC}	V_{PE}	V_{XB}	V_{XB}^{+}	V_{FS}	K	CWB
1.1	5	5	5	5	$\boxed{28}$	5	$\boxed{3}$
1.2	5	5	5	5	$\boxed{29}$	5	$\boxed{3}$
1.3	5	5	5	5	$\boxed{30}$	5	$\boxed{4}$
1.4	5	5	5	5	$\boxed{29}$	5	$\boxed{4}$
1.5	5	5	5	5	$\boxed{30}$	5	$\boxed{4}$
1.6	5	5	5	5	5	5	$\boxed{4}$
1.7	5	5	5	5	5	5	$\boxed{4}$
1.8	5	5	5	5	5	5	$\boxed{4}$
1.9	5	$\boxed{2}$	5	5	5	5	$\boxed{4}$
2.0	5	$\boxed{2}$	5	5	5	5	$\boxed{4}$
2.1	5	$\boxed{2}$	5	5	5	5	$\boxed{4}$
2.2	$\boxed{30}$	$\boxed{2}$	5	5	5	5	$\boxed{4}$
2.3	$\boxed{30}$	$\boxed{2}$	5	5	$\boxed{30}$	5	$\boxed{4}$
2.4	$\boxed{30}$	$\boxed{2}$	5	5	$\boxed{30}$	5	$\boxed{4}$
2.5	$\boxed{30}$	$\boxed{2}$	5	5	$\boxed{29}$	5	$\boxed{4}$
2.6	$\boxed{30}$	$\boxed{2}$	5	5	$\boxed{30}$	$\boxed{4}$	$\boxed{4}$
2.7	$\boxed{30}$	$\boxed{2}$	$\boxed{4}$	5	$\boxed{30}$	$\boxed{4}$	$\boxed{4}$
2.8	$\boxed{30}$	$\boxed{2}$	$\boxed{4}$	5	$\boxed{28}$	$\boxed{4}$	$\boxed{4}$
2.9	$\boxed{30}$	$\boxed{2}$	$\boxed{4}$	5	$\boxed{29}$	$\boxed{4}$	$\boxed{4}$
3.0	$\boxed{30}$	$\boxed{2}$	5	5	$\boxed{30}$	$\boxed{4}$	$\boxed{4}$
5.0	$\boxed{30}$	$\boxed{2}$	$\boxed{4}$	$\boxed{30}$	$\boxed{30}$	$\boxed{4}$	$\boxed{4}$
10.0	$\boxed{2}$	$\boxed{2}$	5	$\boxed{20}$	$\boxed{24}$	5	$\boxed{2}$
20.0	$\boxed{2}$	$\boxed{2}$	$\boxed{3}$	$\boxed{30}$	$\boxed{4}$	$\boxed{3}$	$\boxed{2}$

Appendix E

Cluster validation results for benchmark problem sets

Table E.1: Cluster validation results for benchmark problem sets

(a) Problem set $C1$

Fuzzy factor (m)	**Number of clusters (c)**	V_{XB} (10^{-5})	**Number of clusters (c)**	V_{XB}^{+} (10^{-5})
1.5	4	1.6	4	1.8
1.6	4	1.5	4	1.7
1.7	4	1.4	4	1.6
1.8	4	1.3	4	1.5
1.9	4	1.3	4	1.4
2.0	4	1.3	4	1.3

(b) Problem set $C2$

Fuzzy factor (m)	**Number of clusters (c)**	V_{XB} (10^{-5})	**Number of clusters (c)**	V_{XB}^{+} (10^{-5})
1.5	2	4.6	2	5.3
1.6	2	4.5	2	5.1
1.7	2	4.4	2	4.9
1.8	2	4.4	2	4.8
1.9	2	4.4	2	4.6
2.0	2	4.4	2	4.4

(c) Problem set $R1$

Fuzzy factor (m)	**Number of clusters (c)**	V_{XB} (10^{-5})	**Number of clusters (c)**	V_{XB}^{+} (10^{-5})
1.5	4	2.4	4	2.8
1.6	4	2.2	4	2.7
1.7	4	2.1	4	2.5
1.8	4	2.1	4	2.4
1.9	4	2.1	4	2.3
2.0	4	2.1	4	2.1

Table E.1: Cluster validation results for benchmark problem sets (continued)

(d) Problem set $R2$

Fuzzy factor (m)	**Number of clusters (c)**	V_{XB} (10^{-5})	**Number of clusters (c)**	V_{XB}^{+} (10^{-5})
1.5	4	2.4	4	2.8
1.6	4	2.2	4	2.7
1.7	4	2.1	4	2.5
1.8	4	2.1	4	2.4
1.9	4	2.1	4	2.3
2.0	4	2.1	4	2.1

(e) Problem set $RC1$

Fuzzy factor (m)	**Number of clusters (c)**	V_{XB} (10^{-5})	**Number of clusters (c)**	V_{XB}^{+} (10^{-5})
1.5	4	2.0	4	2.4
1.6	4	1.9	4	2.2
1.7	4	1.8	4	2.1
1.8	4	1.7	4	1.9
1.9	4	1.7	4	1.8
2.0	4	1.7	4	1.7

(f) Problem set $RC2$

Fuzzy factor (m)	**Number of clusters (c)**	V_{XB} (10^{-5})	**Number of clusters (c)**	V_{XB}^{+} (10^{-5})
1.5	4	2.0	4	2.4
1.6	4	1.9	4	2.2
1.7	4	1.8	4	2.1
1.8	4	1.7	4	1.9
1.9	4	1.7	4	1.8
2.0	4	1.7	4	1.7

Appendix F

Training set

Problem	Clusters	V_{XB} ($\times 10^{-5}$)	Time window width (mean)	Time window width (stdev)	Demand (mean)	Demand (stdev)
$c1\_2\_1$	4	13	0.0442	0.0079	18.00	7.91
$c1\_2\_2$	4	13	0.2648	0.3734	18.00	7.91
$c1\_2\_3$	4	13	0.4855	0.4268	18.00	7.91
$c1\_2\_4$	4	13	0.6803	0.3698	18.00	7.91
$c1\_2\_5$	4	13	0.0894	0.0145	18.00	7.91
$c1\_2\_6$	4	13	0.1144	0.0644	18.00	7.91
$c1\_2\_7$	4	13	0.1333	0.0003	18.00	7.91
$c1\_2\_8$	4	13	0.1791	0.0275	18.00	7.91
$c1\_2\_9$	4	13	0.2665	0.0005	18.00	7.91
$c1\_210$	4	13	0.3576	0.0639	18.00	7.91
$c2\_2\_1$	2	44	0.0445	0.0001	19.20	8.13
$c2\_2\_2$	2	44	0.2738	0.3992	19.20	8.13
$c2\_2\_3$	2	44	0.4940	0.4610	19.20	8.13
$c2\_2\_4$	2	44	0.7225	0.4040	19.20	8.13
$c2\_2\_5$	2	44	0.0889	0.0002	19.20	8.13
$c2\_2\_6$	2	44	0.1396	0.0236	19.20	8.13
$c2\_2\_7$	2	44	0.1721	0.0777	19.20	8.13
$c2\_2\_8$	2	44	0.1779	0.0003	19.20	8.13
$c2\_2\_9$	2	44	0.2443	0.1002	19.20	8.13

Problem	Clusters	V_{XB} ($\times 10^{-5}$)	Time window width (mean)	(stdev)	Demand (mean)	(stdev)
$c2$_210	2	44	0.2446	0.0004	19.20	8.13
$r1$_2_1	4	21	0.0158	0.0007	19.04	9.66
$r1$_2_2	4	21	0.2207	0.3771	19.04	9.66
$r1$_2_3	4	21	0.4590	0.4460	19.04	9.66
$r1$_2_4	4	21	0.7074	0.3702	19.04	9.66
$r1$_2_5	4	21	0.0474	0.0004	19.04	9.66
$r1$_2_6	4	21	0.2450	0.3638	19.04	9.66
$r1$_2_7	4	21	0.4748	0.4302	19.04	9.66
$r1$_2_8	4	21	0.7143	0.3571	19.04	9.66
$r1$_2_9	4	21	0.0990	0.0303	19.04	9.66
$r1$_210	4	21	0.1935	0.0441	19.04	9.66
$r2$_2_1	4	21	0.0480	0.0160	17.08	8.48
$r2$_2_2	4	21	0.2604	0.3937	17.08	8.48
$r2$_2_3	4	21	0.4920	0.4674	17.08	8.48
$r2$_2_4	4	21	0.7417	0.4057	17.08	8.48
$r2$_2_5	4	21	0.0948	0.0006	17.08	8.48
$r2$_2_6	4	21	0.2976	0.3730	17.08	8.48
$r2$_2_7	4	21	0.5177	0.4424	17.08	8.48
$r2$_2_8	4	21	0.7547	0.3830	17.08	8.48
$r2$_2_9	4	21	0.1511	0.0682	17.08	8.48
$r2$_210	4	21	0.1877	0.0245	17.08	8.48
$rc1$_2_1	4	17	0.0473	0.0003	17.24	8.50
$rc1$_2_2	4	17	0.2089	0.3355	17.24	8.50
$rc1$_2_3	4	17	0.4394	0.4274	17.24	8.50
$rc1$_2_4	4	17	0.6533	0.3901	17.24	8.50
$rc1$_2_5	4	17	0.1051	0.0484	17.24	8.50
$rc1$_2_6	4	17	0.0946	0.0005	17.24	8.50
$rc1$_2_7	4	17	0.1446	0.0478	17.24	8.50

Problem	Clusters	V_{XB} ($\times 10^{-5}$)	Time window width (mean)	Time window width (stdev)	Demand (mean)	Demand (stdev)
$rc1\_2\_8$	4	17	0.1875	0.0516	17.24	8.50
$rc1\_2\_9$	4	17	0.1893	0.0006	17.24	8.50
$rc1\_210$	4	17	0.2366	0.0007	17.24	8.50
$rc2\_2\_1$	4	17	0.0473	0.0003	17.24	8.50
$rc2\_2\_2$	4	17	0.2789	0.4033	17.24	8.50
$rc2\_2\_3$	4	17	0.5108	0.4659	17.24	8.50
$rc2\_2\_4$	4	17	0.7337	0.4090	17.24	8.50
$rc2\_2\_5$	4	17	0.1114	0.0585	17.24	8.50
$rc2\_2\_6$	4	17	0.0947	0.0005	17.24	8.50
$rc2\_2\_7$	4	17	0.1421	0.0658	17.24	8.50
$rc2\_2\_8$	4	17	0.1995	0.0691	17.24	8.50
$rc2\_2\_9$	4	17	0.1893	0.0007	17.24	8.50
$rc2\_210$	4	17	0.2366	0.0007	17.24	8.50