Université de Montréal

Heuristic solution methods for multi-attribute vehicle routing problems

par

Alireza Rahimi Vahed

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences

en vue de l'obtention du grade de

Doctorat (Ph.D.)

en informatique option recherche opérationnelle

Septembre

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée:

Heuristic solution methods for multi-attribute vehicle routing problems

présenté par:

Alireza Rahimi Vahed

a été évalué par un jury composé des personnes suivantes:

Jacques Ferland
(président-rapporteur)

Michel Gendreau
(directeur de recherche)

Teodor Gabriel Crainic, Walter Rei
(co-directeur)

Gilbert Laporte
(membre du jury)

Manuel Laguna
(examinateur externe)

Thèse acceptée le:

18 / 12 / 2012

Sommaire

Le Problème de Tournées de Véhicules (PTV) est une clé importante pour gérér efficacement des systèmes logistiques, ce qui peut entraîner une amélioration du niveau de satisfaction de la clientèle. Ceci est fait en servant plus de clients dans un temps plus court. En terme général, il implique la planification des tournées d'une flotte de véhicules de capacité donnée basée à un ou plusieurs dépôts. Le but est de livrer ou collecter une certain quantité de marchandises à un ensemble des clients géographiquement dispersés, tout en respectant les contraintes de capacité des véhicules.

Le PTV, comme classe de problèmes d'optimisation discrète et de grande complexité, a été étudié par de nombreux chercheurs au cours des dernières décennies. Étant donné son importance pratique, des chercheurs dans les domaines de l'informatique, de la recherche opérationnelle et du génie industrielle ont mis au point des algorithmes très efficaces, de nature exacte ou heuristique, pour faire face aux différents types du PTV. Toutefois, les approches proposées pour le PTV ont souvent été accusées d'être trop concentrées sur des versions simplistes des problémes de tournées de véhicules rencontrés dans des applications réelles. Par conséquent, les chercheurs sont récemment tournés vers des variantes du PTV qui auparavant étaient considérées trop difficiles à résoudre. Ces variantes incluent les attributs et les contraintes complexes observés dans les cas réels et fournissent des solutions qui sont exécutables dans la pratique. Ces extensions du PTV s'appellent Problème de Tournées de Véhicules Multi-Attributs (PTVMA).

Le but principal de cette thèse est d'étudier les différents aspects pratiques de trois types de problèmes de tournées de véhicules multi-attributs qui seront modélisés dans celle-ci. En plus, puisque pour le PTV, comme pour la plupart des problèmes NP-complets, il est difficile de résoudre des instances de grande taille de façon optimale et dans un temps d'exécution raisonnable, nous nous tournons vers des méthodes approchées à base d'heuristiques.

Mots-clés. Problème de tournées de véhicules, Optimisation discrète, Problème de tournées de véhicules multi-attributs, Heuristique.

Summary

The Vehicle Routing Problem (VRP) is an important key to efficient logistics system management, which can result in higher level of customer satisfaction because more customers can be served in a shorter time. In broad terms, it deals with designing optimal delivery or collection routes from one or several depot(s) to a number of geographically scattered customers subject to side constraints.

The VRP is a discrete optimization and computationally hard problem and has been extensively studied by researchers and practitioners during the past decades. Being complex problems with numerous and relevant potential applications, researchers from the fields of computer science, operations research and industrial engineering have developed very efficient algorithms, both of exact and heuristic nature, to deal with different types of VRPs. However, VRP research has often been criticized for being too focused on oversimplified versions of the routing problems encountered in real-life applications. Consequently, researchers have recently turned to variants of the VRP which before were considered too difficult to solve. These variants include those attributes and constraints observed in real-life planning and lead to solutions that are executable in practice. These extended problems are called Multi-Attribute Vehicle Routing Problems (MAVRPs).

The main purpose of this thesis is to study different practical aspects of three multi-attribute vehicle routing problems which will be modeled in it. Besides that, since the VRP has been proved to be NP-hard in the strong sense such that it is impossible to optimally solve the large-sized problems in a reasonable computational time by means of traditional optimization approaches, novel heuristics will be designed to efficiently tackle the created models.

Keywords. Vehicle routing problem, Discrete optimization, Multi-attribute vehicle routing, Heuristics.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I am heartily thankful to my supervisors, Dr. Michel Gendreau, Dr. Teodor Gabriel Crainic and Dr. Walter Rei, who have supported me throughout my thesis with their patience, invaluable assistance and guidance. Completing a PhD, as a truly marathon event, would not have been possible without their aid and support over the past five years.

I also thank my colleagues and great friends I have worked with at University of Montreal. In particular, I would like to mention Mostafa Nasri, Iman Dayarian and Vahid Famildardashti. Their insights and comments were precious over the years, and I look forward to a continuing collaboration with them in the future.

I would be remiss if I didn't pause to thank a few extraordinary people who have touched my life and have inspired me to accomplish this goal. To Tanaz Kanani, Mostafa Dangchi, Dr. Masoud Rabbani and Dr. Reza Tavakkoli Moghaddam for pushing me beyond my limits and for teaching me valuable life lessons.

Last but not the least, I would like to express my gratitude to my family members, especially my mother, for their understanding and endless love, through the duration of my studies.

Chapter 1

Introduction

The Vehicle Routing Problem (VRP) is one of the most important combinatorial optimization and computationally hard problems in the field of Operations Research. It was introduced by Dantzig and Ramser (1959) and its basic version can be described as follows: a set of customers having deterministic demands have to be satisfied from a central depot with a fleet of homogeneous delivery vehicles of known capacity. Usually, the objective of VRPs is to minimize the total distance traveled by the vehicle fleet, but it is also common to minimize other objectives like the total transportation costs and the number of used vehicles. Effectively solving VRPs in a distribution network can result in higher level of customer satisfaction and substantial savings in the global transportation costs. In other words, better routing decisions improve the capability of a distribution network, thereby enhancing market competitiveness. Toth and Vigo (2002) reported that the use of appropriate solution methodologies to efficiently solve vehicle routing problems in distribution processes often results in savings ranging from 5% to 20% in transportation costs.

The VRP has been a challenging subject for many researchers and a large variety of different optimization methods have been proposed and studied. However, VRP research has often been criticized for being too focused on idealized models with non-realistic assumptions for practical applications. As a result, researchers have turned to variants of the VRP which before were considered too difficult to solve. The variants include aspects of the VRP that are essential to the routing of vehicles in real problems. These extended problems are called Multi-Attribute Vehicle Routing Problems (Rieck and Zimmermann (2006)).

Multi-Attribute Vehicle Routing Problems (MAVRPs) incorporate various complicating attributes and constraints found in real-life applications. Among the real-life requirements are capacity and travel time constraints, time window restrictions, a heterogeneous vehicle fleet with different travel costs, order/vehicle compatibility constraints, multi-dimensional capacity constraints, orders with multiple pickup and delivery, different start and end locations for vehicles, and route restrictions for vehicles.

The most common approach when solving such multi-attribute vehicle routing problems is to either simplify them, or to sequentially solve a series of simpler problems, that are obtained by fixing or ignoring certain parts of the overall problem. This procedure usually leads to suboptimal solutions. The current literature is very scarce to offer a satisfactory answer to this challenge in terms of algorithms able to efficiently tackle multi-attribute vehicle routing problems. The main goal of this thesis is to contribute toward addressing this challenge.

This thesis consists of three papers, each concerning the development of structured heuristics to efficiently tackle a class of multi-attribute vehicle routing problems. In the first paper, we study a Multi-Depot Periodic VRP (MDPVRP) as a well-known variant of the multi-attribute vehicle routing problems. In the considered MDPVRP, a daily plan is computed for a homogeneous fleet of vehicles that depart from different depots and must visit a set of customers for delivery operations over a planning horizon. In this multi-attribute vehicle routing problem, two kinds of constraints, i.e., maximum route length constraint and an upper limit of the number of goods that each vehicle can transport, are considered. Moreover, the cost of each vehicle route is computed through a system of fees depending on the distance that is traveled. Since the VRP, as a generalization of the famous travelling salesman problem, is NP-hard and does not admit high-quality polynomial time approximations, a new Path Relinking Algorithm (PRA) is proposed to find the best possible solutions to this problem. The designed PRA includes different exploitation and exploration strategies that permit the algorithm to solve the problem in two different settings: 1) As a stand-alone algorithm, and 2) As a part of a parallel co-operative search algorithm known as Integrative Cooperative Search (ICS). On the other hand, in the second paper, we address the problem of determining the optimal fleet size for three different MAVRPs, i.e., multi-depot VRP, periodic VRP and multi-depot periodic VRP. Each of these multi-attribute VRPs incorporates three kinds of constraints that are often found in reality, i.e., vehicle capacity, route duration and budget constraints. To solve the problems, we propose a new Modular Heuristic Algorithm (MHA) whose exploration and exploitation strategies enable the algorithm to produce promising results. Finally, in the third part, we study a bi-criteria Multi-Depot Periodic VRP where two contradicting objective functions, i.e., total number of used vehicles and total traveled distance, are to be simultaneously minimized. To solve the problem, a parallel cooperative search approach called Integrative Cooperative Search (ICS) is designed to find locally Pareto-optimal frontier of the problem. The developed ICS is designed based on a new hierarchical decomposition procedure to decompose the problem into more tractable sub-problems, the integration of elite solutions yielded by the sub-problems, and an adaptive guidance mechanism.

The remainder of this thesis is organized as follows. The literature survey relevant to the topic of this thesis is presented in Chapter 2. Chapters 3-5 respectively show the details of the three papers done in this thesis. Finally, Chapter 6 provides conclusions and the evaluation of the work.

Chapter 2

Literature review

In this chapter, the literature relevant to the topic of this thesis is reviewed. This chapter consists of two main sections. Section 2.1 recalls the traditional Capacitated Vehicle Routing Problem (CVRP) and briefly reviews the most prominent heuristics developed to solve it. Most of these heuristics are also found in the next section when analysing heuristics for multi-attribute vehicle routing problems. Section 2.2 focuses on the attribute classification system introduced by Vidal et al. (2012b) and presents the selected MAVRPs and the corresponding subset of chosen well-reported heuristics.

2.1 Heuristics for the CVRP

The CVRP, as a fundamental problem in combinatorial optimization with wide-ranging applications in practice, was introduced by Dantzig and Ramser (1959) under the name ”Truck Dispatching problem”. We initiate the following section by recalling the CVRP formulation. We then review the main categories of heuristic solution methods proposed in the past decades to solve the CVRP.

2.1.1 Problem statement

In general, the CVRP is defined on a directed complete graph $G = (V, A)$ with one depot (node 0) and n customers indexed from 1 to n. A fleet of K identical vehicles with limited capacity Q is based at the depot. Each customer has a known demand q_i and a non-negative travel cost c_{ij} is associated with each arc $(i,j) \in A$. If the matrix c is asymmetric, the corresponding problem is called the asymmetric CVRP. Otherwise, we have $c_{ij} = c_{ji}$ for all $(i,j) \in A$ and the problem is called the symmetric CVRP. The goal is to design a set of routes of minimum total cost to supply all customers. A route is a cycle performed by one vehicle, starting and ending at the depot and visiting a subset of customers. Its total load must not exceed the vehicle capacity. Each customer must be visited by one single route, i.e., split deliveries are not allowed. In general, the number of routes or vehicles used is not imposed, it is a decision variable. The mathematical model of the asymmetric CVRP can be written as follows (Toth and Vigo (2002)).

$$min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.1}$$

S.T

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}; \tag{2.2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}; \tag{2.3}$$

$$\sum_{i \in V} x_{i0} = K; \tag{2.4}$$

$$\sum_{j \in V} x_{0j} = K; \tag{2.5}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset; \tag{2.6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V; \tag{2.7}$$

Given a set $S \subseteq V \setminus \{0\}$, we denote by $r(S)$ the minimum number of vehicles needed to serve all customers in S, i.e., the optimal solution value of the Bin Packing Problem (BPP) associated with item set *S*. The degree constraints (2.2) and (2.3) impose that exactly one arc enters and leave each vertex associated with a customer, respectively. Similarly, constraints (2.4) and (2.5) impose the degree requirements for the depot vertex. The capacity-cut constraints (2.6) impose both the connectivity of the solution and the vehicle capacity requirements. Finally, constraints (2.7) represent binary variable x_{ij} which takes value 1 if arc $(i,j) \in A$ belongs to the optimal solution and takes value 0 otherwise.

The asymmetric CVRP mathematical formulation can be easily adopted to the symmetric problem. In the symmetric CVRP, the arc set *A* is generally replaced by a set of undirected edges *E*. Toth and Vigo (2002) represented the mathematical model of the symmetric CVRP as follows.

$$min \sum_{e \in E} c_e x_e \tag{2.8}$$

subject to

$$\sum_{e \in \delta(i)} x_e = 2 \qquad \forall i \in V \setminus \{0\}; \tag{2.9}$$

$$\sum_{e \in \delta(0)} x_e = 2K; \tag{2.10}$$

$$\sum_{e \in \delta(S)} x_e \geq 2r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \oslash; \tag{2.11}$$

$$x_e \in \{0,1\} \qquad \forall e \notin \delta(0); \tag{2.12}$$

$$x_e \in \{0,1,2\} \qquad \forall e \in \delta(0); \tag{2.13}$$

In the model described above, $\delta(S)$ is the set of edges $e \in E$ that have only one endpoint in *S*. The degree constraints (2.9) and (2.10) impose that exactly two edges are incident to each node associated with a customer and that 2*K* edges are incident to the depot vertex, respectively. The capacity-cut constraints (2.11) impose both the connectivity of the solution and the vehicle capacity requirements by forcing that a sufficient number of edges enter each subset of vertices. Finally, constraints (2.12) and (2.13) state the domain of decision variables. If single-customer routes are not allowed, all used variables are binary; otherwise, if $e \notin \delta(0)$, then $x_e \in \{0,1\}$, whereas if $e \in \delta(0)$, then $x_e \in \{0, 1, 2\}$.

A large amount of research results regarding the CVRP have been reported in terms of theories, methods and algorithms. Generally, solution approaches presented for the CVRP can be classified into exact methods, heuristics and meta-heuristics. Today, the best exact methods for the CVRP are still limited to relatively small problems. Consequently, the development of heuristics and meta-heuristics continues a very active domain in the literature (Vidal et al. (2012b)). The main categories of such methods are reviewed in the following sections.

2.1.2 Route construction methods

Route construction methods were among the first heuristics implemented to solve the CVRP. These algorithms gradually build a feasible solution while attempting to keep solution cost as low as possible. Construction algorithms are divided into sequential and parallel methods. Sequential algorithms expand only one route at a time, whereas parallel methods consider more than one route simultaneously. Route construction algorithms are specified by three main ingredients, namely an initialization criterion, a selection criterion specifying which customers are chosen for insertion at the current iteration, and an insertion criterion to decide where to insert the chosen customers into the current routes.

The most prominent heuristic of this group was proposed by Clarke and Wright (1964). Their suggested algorithm starts with routes only formed by the depot and each node. At every step of the algorithm, two routes are merged if there is a saving in distance. The procedure is repeated until a complete feasible solution is achieved. Another classical route construction heuristic is the sequential insertion algorithm of Mole and Jameson (1976). Their proposed algorithm generalizes the definition of the savings function (Clarke and Wright (1964)), introducing two parameters for controlling the savings behaviour. A two-step insertion heuristic was suggested by Christofides et al. (1979). In the first step, a sequential insertion algorithm is used to determine a set of feasible routes. The second step is a parallel insertion approach. For each route determined in the first step, a customer is selected and a set of single-customer routes is constructed with these customers. The remaining unrouted customers are then inserted using a regret criterion, where the difference between the best and the second-best insertion cost is taken into account, and partial routes are improved by means of a 3-opt procedure.

2.1.3 Two-phase methods

Two-phase methods are based on the decomposition of the CVRP solution process into two separate phases, i.e., clustering and routing. In the clustering phase, a partition of the customers into subsets, each corresponding to a route, is determined, whereas in the routing phase, the sequence of the customers on each subset is obtained.

In cluster-first-route-second methods, customers are first partitioned into different subsets and the routes are then determined by sequencing the customers within each subset. Gillett and Miller (1974) proposed the first cluster-first-route-second method, namely the sweep algorithm. The algorithm is applied to a polar coordinate and the depot is considered to be the centre of the coordinate. The depot is first joined with an arbitrarily chosen node. All other nodes are sequentially joined to the depot and then aligned by increasing the angles which are formed by the segment that connects the nodes to the depot. As soon as the current node cannot be feasibly assigned to the depot, a new subset is initialized with it. Once all nodes are assigned to subsets, the sequence of the nodes on each subset is separately defined by solving a TSP. Another early cluster-first-route-second method was suggested by Fisher and Jaikumar (1981). Their algorithm solves the clustering phase using a Generalized Assignment Problem (GAP) which determines a minimum cost assignment of items to a given set of bins of capacity equal to the vehicle's capacity. Each vehicle is assigned a customer, namely the seed, and the assignment cost of a customer to a vehicle is equal to its distance to the seed. The GAP is then solved, by means of either exact or heuristic algorithms, and the final routes are determined by solving a TSP on each defined cluster. Bramel and Simchi-levi (1995) proposed another two-phase method in which the number of vehicles is to be considered fix and equal to m. This algorithm determines route seeds by solving a capacitated location problem, where m customers are selected by minimizing the total distance between each customer and its closest seed, and by imposing that the total demand associated with each seed be at most equal to the vehicle's capacity. Once seeds have been determined and the single-customer routes are initialized, the remaining customers are inserted in the current routes by minimizing insertion costs.

A different family of cluster-first-route-second methods is petal algorithms. These methods generate a large set of feasible routes, namely petals, and select the final subset by solving a set partitioning model as follows (Balinski and Quandt (1964)).

$$min \sum_{k \in S} c_k x_k \tag{2.14}$$

S.T

$$\sum_{k \in S} a_{i_k} x_k = 1 \qquad \forall i = 1, ..., n; \tag{2.15}$$

$$x_k \in \{0, 1\} \qquad \forall k \in S; \tag{2.16}$$

where *S* is the set of routes, binary variable x_k takes value 1 if and only if *k* belongs to the solution, a_{i_k} is the binary parameter equal to 1 if vertex *i* belongs to route *k* and the c_k is the cost of petal *k*. Foster and Ryan (1976) and Ryan et al. (1993) proposed heuristic rules, called 1-petals, for determining the set of routes to be selected, while Renaud et al. (1996b) described an extension that considers more involved configurations, called 2-petals, consisting of two embedded or intersecting routes.

Finally, in route-first-cluster-second methods, a giant TSP tour over all customers is constructed in a first phase and subdivided into feasible routes in the routing phase. Examples of such algorithms are given by Beasley (1983), Haimovich and Kan (1985), and Bertsimas and Simchi-Levi (1996).

2.1.4 Route improvement methods

Route improvement heuristics are often local search algorithms used to improve initial solutions usually generated by construction heuristics. Starting from a given initial solution, a local search method applies simple structural modifications to obtain neighbour solutions of possibly better quality. These local search algorithms may be divided into intra-route neighbourhoods which operate on a single route at a time, or inter-route neighbourhoods which consider more than one route simultaneously. The most well-known local search algorithm studied in the literature is the λ-opt heuristic proposed by Lin (1965) for the TSP, where λ edges are removed from the current solution and replaced by λ others. As an alternative, restricted neighbourhoods characterized by subsets of moves associated with larger λ values can also be used. For example, Or-exchanges (Or (1976)) or the 4-opt* neighbourhood of Renaud et al. (1996a) which considers only a subset of all potential 4-opt exchanges. More complex inter-route neighbourhoods are analysed by Thompson and Psaraftis (1993).

2.1.5 Meta-heuristics

Meta-heuristics are a powerful tool to solve combinatorial optimization problems. Different meta-heuristics have been proposed to solve the CVRP. When compared to the classical heuristics mentioned in the previous section, meta-heuristics perform a more thorough search of the solution space and are less likely to get trapped in a local optimum. These methods are divided into two classes, i.e., neighbourhood search and population-based methods.

Neighbourhood search method

Neighbourhood search algorithms explore the solution space by iteratively moving from a solution x_t at iteration t to a solution x_{t+1} in the Neighbourhood $N(x_t)$ of x_t until a stopping criterion is met. If $f(x)$ represents the cost of solution x, then $f(x_{t+1})$ is not necessarily smaller than $f(x_t)$. As a result, mechanisms must be implemented to avoid cycling. A large number of neighbourhood search heuristics have been produced over the past decades. The first of these methods is the simulated annealing algorithm, which has been studied by a little number of researchers in the early of 1990s. In simulated annealing, a solution x is drawn, either randomly or based some principles, from $N(x_t)$. If $f(x) < f(x_t)$, then $x_{t+1} := x$. Otherwise, $x_{t+1} := x$ with probability p_t or $x_{t+1} := x_t$ with probability 1 - p_t, where p_t is a decreasing function of t and of $f(x)$-$f(x_t)$. The most famous simulated annealing algorithm was developed by Osman (1993). The most interesting features of the proposed algorithm are defining neighbourhoods by means of a 2-interchange scheme and applying a different rule of temperature changes. This algorithm produced good solutions but was not competitive enough with the best tabu search implementations available at the same period. Another well-known simulated annealing-based algorithm proposed to solve the CVRP is the deterministic annealing strategy. Deterministic annealing operates in a way that is similar to the simulated annealing algorithm, except that a deterministic rule is applied for acceptance of a move (Toth and Vigo (2002)). The deterministic annealing algorithm was first developed for the CVRP by Golden et al. (1998). The proposed method is a descent algorithm which works based on the record-to-record travel heuristic suggested by Dueck (1993). In record-to-record travel, a record is the best solution x^* found during the optimization procedure. At iteration *t*, solution x_{t+1} is accepted if $f(x_{t+1}) < \theta f(x_t)$, where θ is a user-controlled parameter.

Another neighbourhood search method considered in this section is the tabu search algorithm. Tabu search starts from an initial solution and moves at each iteration from the current solution to the best one in its neighbourhood, even this leads to a deterioration of the objective function value. To avoid cycling, attributes of recently visited solutions are declared tabu for a certain number of iterations. This process is repeated until a a stopping criterion is satisfied (Glover and Laguna (2003)). Contrary to simulated annealing, a large variety of tabu search algorithms have been developed to solve the CVRP. One of the first attempts to apply tabu search to the CVRP is due to Willard (1989). In the proposed algorithm, the initial solution is first transformed into a giant tour by replication of the depot, and neighbourhoods are then defined as all feasible solutions that can be reached from the current solution using 2-opt or 3-opt exchanges. The next solution is determined by the best non-tabu move. Another tabu search algorithm which is one of the most successful tabu search implementation for the CVRP was proposed by Taillard (1993). The algorithm uses a 1-interchange mechanism without local reoptimization, and without allowing infeasibilities. Periodically, routes are reoptimized using an exact TSP algorithm. The search procedure developed by Taillard employs a decomposition scheme that lends itself to the use of parallel computing. In planar instances, the set of customers is first partitioned into sectors centered at the depot, and also into concentric circles. Search is performed in each subregion by a different processor. The subregion boundaries are updated periodically to provide a diversification effect. In non-planar problems, regions are defined through the computation of shortest spanning arborescences rooted at the depot.

In Taburoute (Gendreau et al. (1994)), neighbour solutions are obtained by moving a vertex from its current route to another route containing one of its closest neighbours. Insertions are performed simultaneously with a local reoptimization of the route, based on the GENI procedure (Gendreau et al. (1992)). To limit the neighbourhood size, only a randomly selected subset of vertices are considered for reinsertion in other routes. Taburoute also uses a continuous diversification mechanism. During the course of the search, infeasible solutions are penalized using a function whose parameters are progressively modified during the optimization process. Other features of Taburoute include the use of random tabu durations, periodic route reoptimizations by means of the US procedure of Gendreau et al. (1992), false starts to initialize the search, and a final intensification phase around the best known solution.

Rego and Roucairol (1996) developed another tabu search algorithm based on an ejection chains method involving l levels, or routes, to define neighbourhoods. Ejection Chains are variable depth methods that generate a sequence of interrelated simple (component) moves to create a more complex compound move. There are several types of ejection chains, some structured to induce successive changes in problem variables and others structured to induce changes in particular types of model components (such as nodes and edges of a graph) (Glover (1992)). In the proposed algorithm, an ejection chain is considered only if no arc (edge) appears more than once in the solution, but routes that violate capacity or duration constraints are accepted. A parallel version of this algorithm was implemented by the authors. Another ejection chains-based tabu search algorithm was proposed by Xu and Kelly (1996). The suggested method oscillates between ejection chains and vertex swaps between two routes. The ejection chains are obtained by solving an auxiliary network flow problem. This method obtains several good CVRP solutions on benchmark instances but is rather time consuming.

Another tabu search implemented to solve the CVRP was proposed by Bachem et al. (1996). The proposed heuristic is based on the procedures of trading. The clustering of customers into tours is determined by finding matches in a leveled bipartite graph, namely trading graph. The nodes correspond to either an insertion (buy) of a customer into a tour or a deletion (sell). The edges represent possible exchanges and the weight of each edge is the gain that is obtained by the corresponding action. Thus, every matching of the trading graph corresponds to a number of interchanges of customers. At each iteration, tours are shuffled by choosing some permutation at random. Then for each tour either a sell or buy action is selected and finally possible trading matches are evaluated and the best one selected. The approach allows infeasibilities given certain penalty factors, as well as trading matchings with negative weights causing deterioration. Because of this deterioration a tabu list is also added to prevent cycling. The approach was implemented using two different kinds of parallelizations.

Toth and Vigo (2003) proposed a very interesting tabu search, namely Granular Tabu Search (GTS). The idea of this algorithm is to permanently remove from consideration long edges that have only a small likelihood of belonging to an optimal solution. More specifically, a threshold is defined and the search is performed on the restricted edge set *E(V)* $= \{(v_i, v_j) \in E : c_{ij} \leq v\} \cup$ *I*, where *I* is a set of important edges defined as those incident to the depot. The value of *v* is set equal to $\beta\bar{c}$, where β is called a sparsification parameter, and $\bar{c}$ is the average edge cost in a good feasible solution quickly obtained, for example, by the Clarke and Wright (1964) algorithm.

Population-based methods

This category of meta-heuristics are based on the essence of natural evolution processes, which involve the reproduction, random variation, competition, and selection of contending individuals in a population.

The first type of the population-based meta-heuristics is Genetic Algorithms (GAs). GAs are an approach to optimization and learning based loosely on principles of biological evolution. Genetic algorithms maintain a population of possible solutions to a problem, encoded as chromosomes based on a particular representation scheme. After generating an initial population, new individuals for this population are generated via the process of reproduction. Parents are randomly selected from the current population for reproduction with the better ones (according to the evaluation criteria) more likely to be selected. The genetic operators of mutation and crossover generate children (i.e., new individuals) by random changes to a single parent or combining the information from two parents respectively (Holland (1975)). Prins (2004) developed an effective GA combining the two important features of evolutionary algorithms, i.e., crossover and mutation operations. In the proposed algorithm, solutions are represented as a giant tour without trip delimiters. To create an offspring from two parents, a chain $(1, ..., j)$ is first selected from the first parent and the vertices of the second parent are scanned from position *j*+1 by skipping those of the chain $(1, ..., j)$. A second offspring is generated in a similar way by reversing the roles of the two parents. Offspring are improved by applying a combination of vertex and edge reinsertions, vertex swaps, combined vertex and edge swaps. Baker and Ayechew (2003) proposed a GA for the CVRP. The representation of this algorithm has been influenced by the non-binary representation of Chu and Beasley (1997) for the Generalised Assignment Problem (GAP). Without specifying explicitly the exact route which each vehicle should follow but only the assignment of n customers to m vehicles known, the chromosome for an individual solution has the form of a string of length n, with each gene value in the range $[1,m]$. Then, a TSP algorithm is applied to each route and a fitness value is associated with each population member. Specifically, the initial population consists of structured solutions generated by employing either the sweep approach of Gillett and Miller (1974) or the generalized approach of Fisher and Jaikumar (1981).

The second type of the population-based algorithms is Ant Colony Optimization (ACO). ACO is based on the behavior of real ants and possesses enhanced abilities such as memory of past actions and knowledge about the distance to other locations. In nature, ants communicate using a chemical substance called pheromone. As an ant travels, it deposits a constant amount of pheromone that other ants can follow. Each ant moves in a somewhat random fashion, but when an ant encounters a pheromone trail, it must decide whether to follow it. If it follows the trail, the ant's own pheromone reinforces the existing trail, and the increase in pheromone increases the probability of the next ant selecting the path. Additionally, an ant using a short route to a food source will return to the nest sooner and therefore, mark its path twice, before other ants return. This directly influences the selection probability for the next ant leaving the nest. Trail selection by ants is a pseudo-random proportional process and is a key element of the simulation algorithm of ant colony optimization (Dorigo et al. (1999)). The early ant colony optimization heuristics could not compete with respect to the best available methods. However, Reimann et al. (2002) developed recently an ant colony optimization algorithm based on the transformation of the simultaneous tour construction mechanism proposed by Clarke and Wright (1964) into a rank-based ant colony optimization algorithm. The first step of the proposed algorithm starts with the generation of a list of attractiveness values, sorted in decreasing order. Then, the probability of visiting vertex u_j after vertex u_i is computed based on the attractiveness values computed in the previous step. Each solution is then improved using the 2-opt algorithm, separately for each route constructed by the ants. Reimann et al. (2004) proposed an ant colony optimization approach, namely D-Ants, built on the algorithm developed by the same authors in (2002). The main advantage of the D-Ants approach is the concept of divide and conquer, according to which, a decomposition of the set of tours that constitute the complete CVRP into a number of smaller sets of tours with geographically proximity is proposed. The resulting sub-problems are solved by the algorithm of Reimann et al. (2002).

2.1.6 Parallel algorithms

Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently. Parallelism is divided into two main categories, i.e., functional and data parallelism. In functional parallelism, different tasks are allocated to different processors and run in parallel, possibly exchanging information, while in data parallelism, the feasible domain of the problem is partitioned and a particular solution approach is used to tackle the problem of each component of the problem. According to how large the tasks are, a few instructions or a sizeable part of the algorithm, the parallelization is called fine- or coarse-grained, respectively (Crainic (2007)).

In parallel algorithms, information must be exchanged among tasks to provide the necessary data for computations or the estimation of the global status of the search. Communications may be performed synchronously or asynchronously. In the former case, all concerned tasks have to stop and engage in some form of communication, whereas in the latter case, each task is in charge of establishing communications with other tasks, according to its internal logic. There are a large variety of different parallel algorithms implemented to efficiently tackle hard optimization problems. In this section, we focus our attention to one special type of these parallel algorithms successfully implemented to solve the general VRP, i.e., parallel cooperative search mechanisms.

Cooperative search is a parallelization strategy increasingly used for search methods. Contrary to conventional parallelization strategies, the parallel tasks of cooperative procedures are not obtained from the decomposition of the data or the search program. Most often, these tasks are obtained by executing concurrently several search programs using the same data set. These parallel tasks are said to cooperate with each other because, at run time, search programs exchange information on previously explored regions of the solution space. By providing shared information among search programs, cooperative design aims to improve the performance of the individual search programs through the reuse of information collected by other programs (Bouthillier et al. (2005)).

The most famous cooperative search algorithm proposed to solve the CVRP is the Adaptive Memory Procedure (AMP). An adaptive memory is a pool of good solutions which is updated by replacing its worst elements with better ones. In order to generate a new solution, several solutions are selected from the pool and recombined. Rochat and Taillard (1995) introduced for the first time the concept of AMP for the CVRP, according to which a set of high quality VRP solutions is stored in a pool that is dynamically updated throughout the optimization process. Routes of the solutions in the set are extracted periodically, giving a larger weight to those routes belonging to the best solutions, and are combined to construct a new solution. Since the result is almost always a partial solution, a constructive heuristic is then used to complete it. Tarantilis and Kiranoudis (2002) also developed another adaptive memory-based algorithm called BoneRoute. There are two main differences between the Rochat and Taillard (1995) and the Tarantilis and Kiranoudis (2002) algorithms. First, BoneRoute provides new starting solutions through the selection and combination of sequences of nodes, called bones (rather than vehicle routes). Second, the Rochat and Taillard algorithm selects routes from the pool probabilistically, while BoneRoute chooses the extracted bones systematically considering the number of nodes.

2.2 MAVRP classification and state-of-the-art heuristics

The majority of VRP attributes are directly derived from the requirements encountered in real-life applications. They are the subject of a huge amount of studies, grouping various thousands of scientific papers. Vidal et al. (2012b) quite recently distinguished three main groups of attributes: the *Assignment of customers and routes to resources* (ASSIGN), the *Sequence choices* (SEQ). and the *Evaluation of fixed sequences* (EVAL).

ASSIGN attributes impact assignment of limited resources, e.g., depots, vehicles, and vehicle types, to customer services and routes. Most common ASSIGN attributes consist of multiple depots, periodic, heterogeneous fleets and split deliveries. On the other hand, SEQ attributes impact directly the structure of the routes. In a backhaul setting, for example, the route is made up of two sequences of linehaul and backhaul services, respectively. Finally, EVAL attributes impact a large variety of evaluations and constraints checks that must be performed once the route contents and orders are selected. The literature is very large on this type of attributes, some of the most common being time windows and time-dependent route durations or costs.

Since each of the problems, considered in this thesis, includes a subset of ASSIGN attributes, we focus our attention on reviewing the most prominent heuristics and meta-heuristics proposed to solve the vehicle routing problems of this group.

2.2.1 Heuristics for VRP variants with ASSIGN attributes

Multi-Depot VRP (MDVRP)- Consider a distribution company with multiple depots. The number and locations of the depots are predetermined. Each depot is large enough to store all the products ordered by the customers. A fleet of vehicles with limited capacity is used to transport the products from depots to customers. Each vehicle starts and finishes at the same depot. The location and demand of each customer is also known in advance. Each customer is visited by a vehicle exactly once. This practical distribution problem is denoted the MDVRP, in which there are three decisions. The decision makers first need to cluster a set of customers to be served by the same depot, that is, the grouping problem. They then have to assign customers of the same depot to vehicles such that capacity requirements are enforced. At last, the decision on delivery sequence of each route is made. Generally, the objective of the MDVRP is to minimize the total delivery distance or time spent in serving all customers. Shorter delivery time results in higher level of customer satisfaction. Besides, the objective can also be the minimization of the number of vehicles needed. Fewer vehicles imply that the total operation cost is reduced. No matter which type of objective is defined, the ultimate goal of the MDVRP is to increase the efficiency of the delivery (Salhi and Sari (1997)).

Several heuristics have been put forward for the MDVRP. Early heuristics, performing based on simple construction and improvement procedures, have been developed by Tillman (1969), Tillman and Hering (1971), Tillman and Cain (1972), Gillett and Johnson (1976), Golden et al. (1977), and Raft (1982).

Cassidy and Bennett (1972) proposed an iterative heuristic for the multi-depot vehicle routing problem. The proposed method progressively improves the routing arrangements starting from an initial solution. An interesting feature of the algorithm is the method of data storage, which is designed to facilitate the alteration of route configurations. The suggested heuristic is divided in three main steps. In the first step, an initial solution is generated by assigning each customer to its nearest depot. In the second step, the initial solution obtained from the previous step is improved by taking each customer in turn and trying to fit it into another position. Finally in the last step, the algorithm examines all depots in the routes to see if any of them can be replaced by any of those still having enough capacity. Several years later, Chao et al. (1993) proposed a search procedure combining the record-to-record local search method for the reassignment of customers to different vehicle routes, followed by a 2-opt procedure for the improvement of individual routes. Salhi and Sari (1997) suggested a multi-level construction-based composite heuristic for solving a multi-depot fleet mix vehicle routing problem in which allocating customers to depots, finding the delivery routes and determining the vehicle fleet composition are simultaneously considered. The main purpose of that paper was to minimize the total traveled cost where both the vehicle capacity (the largest vehicle in case there are different types of vehicles) and the maximum distance traveled on any route must not be violated. The proposed heuristic consists of three levels. In the first level, a starting solution is found as follows: the vehicle fleet mix problem is first solved within each depot with certain customers left unassigned (borderline customers). Then each of these customers is inserted into an existing route or an empty route by using a selection-insertion procedure. In the second level, a composite heuristic, which attempts to improve on the solution found for each depot when taken separately, is introduced. Finally in the third level, a composite heuristic which considers all depots is implemented.

Various meta-heuristics have also been developed to tackle the MDVRP. Renaud et al. (1996c) described a tabu search heuristic in which an initial solution is built by first assigning every customer to its nearest depot. A petal algorithm is then used for the solution of the VRP associated with each depot. The algorithm is completed by an improvement phase using either a subset of the 4-opt exchanges to improve individual routes, swapping customers between routes from the same or different depots, or exchanging customers between three routes. The tabu search approach of Cordeau et al. (1997) is probably the best known algorithm for the MDVRP. An initial solution is obtained by assigning each customer to its nearest depot and a VRP solution is generated for each depot by means of a sweep algorithm. Improvements are performed by transferring a customer between two routes incident to the same depot, or by relocating a customer in a route incident to another depot. Reinsertions are performed by means of the GENI heuristic (Gendreau et al. (1992)). One of the main characteristics of this algorithm is that infeasible solutions are allowed throughout the search. Continuous diversification is achieved through the penalization of frequent moves. Dondo and Cerdà (2007) studied the multi-depot vehicle routing problem with time windows. To solve it, they presented a model-based large-scale neighbourhood search algorithm that steadily improves an initial solution generated through the three-phase cluster-based hybrid approach. At each iteration, a sequence of two evolutionary steps is executed. First, a neighbourhood around the starting solution is generated by using a mixed-integer linear problem that permits the algorithm to exchange multiple nodes between neighbouring trips. Next, a different neighbourhood is defined by just allowing relocations of nodes on the same tour. Lau et al. (2010) addressed an MDVRP in which the objective is to simultaneously optimize both the cost due to the total traveling distance and that due to the total traveling time. To solve the problem, a genetic algorithm with fuzzy logic adjusting the crossover rate and mutation rate after ten consecutive generations was proposed. Finally, Yu et al. (2011) designed a parallel ant colony optimization algorithm for the MDVRP. In the proposed algorithm, three improved strategies: the coarse-grain parallel strategy, the ant weight strategy and the local search strategy, were applied.

Periodic VRP (PVRP)- In the PVRP, as in the CVRP, customer locations each with a certain demand function are given, as well as a set of capacitated vehicles. In addition, the PVRP has a planning horizon of T periods, a service frequency for each customer, stating how often within these T periods the customer must be visited, and a list of possible visit-period combinations. A solution to the PVRP consists of T sets of routes that jointly satisfy the demand constraints and the frequency constraints. The objective is to minimize the sum of the costs of all routes over the planning horizon. Obviously, this problem is at least as hard as the CVRP. Several variants of the PVRP are described in the literature. A classification of the different variants of the PVRP can be found in a survey by Mourgaya and Vanderbeck (2006). Different objective functions are distinguished, such as minimizing the distance traveled, the driving time, or total transportation cost; also regionalization of routes, an even spread of workload over the vehicles, the number of vehicles, and service quality can be part of an optimization function. Differences also occur in the constraints which can be divided in three categories: constraints concerning (i) the planning of visits (different frequencies, restrictions on certain periods, etc.), (ii) the type of demand (constant or variable), and (iii) the fleet of vehicles (homogeneous or heterogeneous).

Solution algorithms proposed to solve the PVRP can be categorized into two main groups, i.e., classical heuristics, and meta-heuristics. Heuristics have been extensively studied to solve the PVRP. The majority of these heuristics are multi-phase optimization approaches which try to solve the problem at hand in a sequential manner. Russell and Gribbin (1991) presented a multi-phase approach to solve the PVRP. The first phase of the proposed method consists of a procedure which generates initial solutions by using a generalized network approximation method. The second phase involves an interchange heuristic that reduces the total traveled cost through a surrogate traveling salesman problem. In the third phase, the total traveled cost is further reduced by addressing the actual routes. Finally, a proposed 0-1 integer model is used to attempt further improvements. Chao et al. (1993) provided a two-phase heuristic. To obtain an initial solution they solve an integer linear program to assign visit day combinations to the customers. In a second phase, they use several improvement operators while they relax the capacity of the vehicles. When getting stuck, re-initializations are performed. Bertazzi et al. (2004) suggested a heuristic algorithm for a special case of the PVRP namely the periodic traveling salesman problem, in which a single vehicle is used in each period. The algorithm is a construction type with an embedded improvement procedure. At each iteration, a procedure selects a not yet processed city, assigns to it a combination of visit days and, for each day of the combination day, inserts the city to the best position of the current partial tour. The iteration process is temporarily interrupted after a predefined number of iterations and an iterative improvement procedure tries to improve the current solution.

These early heuristics are outperformed by more recent meta-heuristic approaches, including tabu search, scatter search, and variable neighbourhood search. Cordeau et al. (1997) proposed a tabu search heuristic for the PVRP that can also be used to solve the Multi-Depot Vehicle Routing Problem and the Periodic Traveling Salesman Problem (PTSP). The neighbourhood consists of moving a customer from one route to another route of the same day or assigning a new visit combination to a customer. Insertions and removals of customers are performed using the GENI operator (Gendreau et al. (1992)). The tabu search algorithm allows for infeasible solutions during the search process using an adaptive penalty function. This paper presents an asynchronous parallel metaheuristic for the period vehicle routing problem (PVRP). Drummond et al. (2001) designed an island-based parallel meta-heuristic for the PVRP. The proposed algorithm was based on concepts used in parallel genetic algorithms and local search heuristics. Angelelli and Speranza (2002) presented a tabu search algorithm for an extension of the periodic vehicle routing problem where the homogeneous vehicles have the possibility of renewing their capacity at some intermediate facilities. The initial solution of the proposed tabu search is generated by using a procedure similar to the sweep algorithm (Gillett and Miller (1974)). Then, the initial solution is improved via an improvement procedure which consists of four move operators, i.e., relocation, changing the visiting schedule of a customer, redistribution, intersection. To enhance the performance of the proposed algorithm, the tabu search is permitted to search the solution space by using a tunneling strategy. Besides that, a diversification mechanism is also used. Recently, a scatter search procedure was developed by Alegre et al. (2007) for solving a problem of periodic pick-up of raw materials for a manufacturer of auto parts. They use a two-phase approach, that first assigns orders to days and then constructs the routes of each day. Alonso et al. (2008) proposed a tabu search for an extension of the periodic vehicle routing problem where each vehicle can service more than one route per day as long as the maximum delay operation time in not exceeded. Besides that, there exist some accessibility constraints of the vehicles to the customers in the sense that not every vehicle can visit every customer. The efficiency of the implemented tabu search is proved based on some existing and randomly generated test problems. Hemmelmayr et al. (2009) implemented a variable neighbourhood search for the periodic vehicle routing problem. First, for obtaining an initial solution each customer is randomly assigned a visit day combination. Routes are constructed by solving a vehicle routing problem for each day using Clarke and Wright savings algorithm (Clarke and Wright (1964)). Then, for the shaking phase two popular and effective neighbourhoods, i.e., move and cross-exchange, are proposed in order to enhance the quality of the starting solution in each iteration. Finally, the solution obtained through shaking is further improved by using a local search procedure based on the 3-opt operator. Pirkwieser and Raidl (2010) proposed a variable neighbourhood search for the periodic vehicle routing problem with time windows. In that paper, the authors claimed that using a random VNS often yielded significantly better results than a VNS using a reasonable fixed ordering of the shaking neighbourhoods. Furthermore, a selectively applied simple inter-route improvement procedure, 2-opt*, was shown to considerably improve both VNS variants at nearly no computational cost at all. Gulczynski et al. (2011) developed a new heuristic for the PVRP that combined integer programming and the record-to-record travel algorithm. The proposed heuristic produced very high-quality results on standard benchmark instances. The authors also extended the heuristic to two new variants of the PVRP that involve reassigning customers to new routes and balancing the workload among drivers across routes.

Multi-Depot Periodic VRP (MDPVRP)- Another MAVRP is the MDPVRP which combines the two above problem settings, asking for the selection of a depot and a visit pattern for each customer, with services in different periods to the same customer being required to originate at the same depot (Vidal et al. (2012a)).

The majority of solution methods, targeting the MDPVRP, are divided into two main groups: 1) Classical heuristics which often solve the problem in a sequential manner, and 2) Sophisticated meta-heuristics and parallel algorithms which tackle the problem by simultaneously optimizing all the involved attributes.

We are aware of three heuristics of the first group. Hadjiconstantinou and Baldacci (1998) formulated the problem of providing maintenance services to a set of customers as the MDPVRP with Time Windows (MDPVRPTW). The authors proposed a multi-phase optimization problem and solved it using a four-phase algorithm. In the developed algorithm, all customers are first assigned to particular depot. Then, customer visits are successively inserted among available periods to obtain feasible visit combinations. At the third phase, each of the depot-period VRP sub-problems is separately solved using a tabu search algorithm. Finally, at the last phase, solutions obtained during the optimization process are improved by modifying the period or depot assignments through a 3-opt procedure. Kang et al. (2005) designed a two-phase heuristic method to solve the problem considered by Hadjiconstantinou and Baldacci (1998). In the proposed method, all feasible schedules are first generated from each depot for each period and, then, the set of routes are determined through using the shortest path problem. Parthanadee and Logendran (2006) proposed a tabu search heuristic to tackle the problem considered by Hadjiconstantinou and Baldacci (1998)). In this algorithm, all the initial assignments are built by cheapest insertion, where each customer is assigned to its nearest depot and is given its most preferred visit pattern. At the improvement phase, a neighbourhood search is defined by depot and visit pattern interchanges.

We are also aware of two contributions belonging to the second group. The first contribution was the evolutionary meta-heuristic proposed by Vidal et al. (2012a). The authors developed a hybrid Genetic Algorithm (GA) to tackle the MDPVRP and two of its special cases, i.e., the Multi-depot VRP (MDVRP) and the Periodic VRP (PVRP). The most interesting feature of the proposed GA is a new population-diversity management mechanism which allows a broader access to reproduction, while preserving the memory of what characterizes good solutions represented by the elite individuals of the population. The second contribution was the cooperative parallel algorithm designed by Lahrichi et al. (2012). The authors proposed a well structured cooperative parallel search method, denoted Integrative Co-operative Search (ICS), to solve highly complex combinatorial optimization problems. The proposed ICS framework involves problem decomposition by decision sets, integration of elite partial solutions yielded by the sub-problems, and adaptive guiding mechanism. The authors used the MDPVRPTW to present the applicability of the developed methodology.

Heterogeneous Fleet VRP (HFVRP)-The Heterogeneous Fleet Vehicle Routing Problem (HFVRP) is another variant of the MAVRPs where the vehicles do not necessary have the same capacity, vehicle fixed cost and unit variable cost. We are also given a set of customers, *N*, a certain number of vehicle types, *M*, each of which has a vehicle capacity Q_m, a fixed cost F_m and a unit variable cost a_m (*m* = 1,...,*M*). As in the CVRP, each customer must be served by one vehicle only, each vehicle must start and finish its journey at a central depot and the capacity of a vehicle and the maximum length of a route must not be exceeded. The objective of the HFVRP is to minimize the total cost which includes both the vehicle variable and fixed costs. The idea is not only to consider the routing of the vehicles, but also the composition of the vehicle fleet. A special case of HFVRP is the fleet size and mix vehicle routing problem (Golden et al. (1984)) also called the fleet size and composition VRP or the vehicle fleet mix. The goal of this problem is to determine a fleet of vehicles such that the sum of fixed costs and travel costs is minimized. This problem is a particular HFVRP for which the number of vehicles of each type is not limited.

The existing papers in the literature proposed for solving all the HFVRP variants have focused on developing heuristic algorithms instead of exact solution methods. They can be broadly grouped into two categories: classical heuristics mostly derived from the classical CVRP heuristics, and meta-heuristics.

Golden et al. (1984) proposed for the first time a heuristic algorithm to solve the HFVRP. They created several heuristics based on the savings method of Clarke and Wright (1964), as well as on the giant tour algorithm of Gillett and Miller (1974). Gheysens et al. (1986) adapted the generalised assignment heuristic of Fisher and Jaikumar (1981). Desrochers and Verhoog (1991) proposed a matching based savings heuristic initially proposed by themselves for the VRP. The most recent heuristic is due to Renaud and Boctor (2002) and is, to date, the one that produces better quality solutions, but also the one that requires more computing time. This heuristic starts by generating a large set of routes using different procedures and afterwards chooses those that satisfy the constraints of the problem at the lowest cost using an exact, but polynomial, set partitioning algorithm. Among those procedures of producing the initial set of routes is the petal method used by several authors, including Renaud et al. (1996b), for the VRP.

More recently, meta-heuristics have been proposed for solving the HFVRP. Gendreau et al. (1999) and Wassan and Osman (2002), which will be designated from now on as OS, GLMT and WO, respectively. All these algorithms are based on tabu search. The OS algorithm takes an initial solution produced by the heuristic of Salhi and Rand (1993), which is then improved by a tabu search method with short term memory and with moves defined by a 1-interchange mechanism. The GLMT algorithm is rather complex and requires the use of GENIUS, developed by Gendreau et al. (1992) for the travelling salesman problem (TSP), as well as an adaptive memory procedure developed by Rochat and Taillard (1995). The WO algorithm comprises several variants obtained from the selection of different neighbourhood mechanisms, tabu restrictions and tabu tenure schemes. Very recently, Paraskevopoulos et al. (2008) considered the heterogeneous fleet routing problem with time windows where the total distribution costs are to be minimized. To solve the problem a two-stage algorithm based on a hybridized tabu search within a novel reactive variable neighbourhood search algorithm is implemented. In the first stage, several initial solutions are generated by using a semi-parallel construction heuristic. Then, a route elimination procedure is applied to reduce the number of vehicles. Finally, a subset of high quality solutions is selected to further improve the obtained solution. In the second stage, a reactive variable neighbourhood tabu search is proposed to minimize the total distribution costs of the selected initial solutions generated in the first stage. Choi and Tcha (2007) developed a mathematically based resolution method for the HFVRP. In this method, the HFVRP is formulated as a set covering problem as it is done for VRP with time windows. Then, its linear programming relaxation is solved by the column generation technique. This method could not solve exactly any of the examples tested, but it generated good lower bounds and upper bounds and, therefore, can be used as a good heuristic method. Imran et al. (2009) proposed a variable neighbourhood search (VNS) for a heterogeneous vehicle routing problem. The initial solution is obtained in three steps. First, a giant tour is constructed using the sweep algorithm (Gillett and Miller (1974)). Then, the constructed tour is improved using the 2-opt of Lin (1965). Finally, the cost network is constructed and the Dijkstra algorithm is applied to find the corresponding optimal fleet size. The proposed VNS uses several neighbourhood structures, i.e. 1-1 interchange, 2-0 shift, 2-1 interchange, and a perturbation mechanism. It also applies six different local searches, i.e.1-insertion (inter-route and intra-route), 2-insertion (intra-route), 2-opt (inter-route and intra-route), and swap (intra-route), to improve current solution. A Dijkstra-based method and a diversification procedure are also implemented after the local search phase to enhance the quality of the solution obtained and explore other unvisited regions of the solution space. Liu et al. (2009) suggested an effective genetic algorithm for the fleet and mix vehicle routing problem, in which the fleet is heterogeneous and its composition is to be determined. In the proposed algorithm, first, a set of initial solutions is generated by using two different construction methods, i.e. savings method and sweep algorithm. Each generated initial solution is converted to its corresponding chromosome representation. Then, for a predetermined number of iterations, two solutions as parents are randomly selected by using the tournament selection method to generate offsprings. The desired number of offsrpings is constructed through the reproduction phase by using two different crossovers, namely order crossover and single parent crossover. The generated offsprings are then mutated via a series of local searches, i.e. string relocation, string cross, and string exchange.

Split Delivery VRP (SDVRP)- Another extension of the MAVRPs is the Split Delivery Vehicle Routing Problem (SDVRP) where a fleet of capacitated homogeneous vehicles has to serve a set of customers. Each customer can be visited more than once, contrary to what is usually assumed in the CVRP, and the demand of each customer may be greater than the capacity of the vehicles. There is a single depot for the vehicles and each vehicle has to start and end its tour at the depot. The problem consists of finding a set of vehicle routes that serve all the customers such that the sum of the quantities delivered in each tour does not exceed the capacity of a vehicle, the demand of each customer is fully satisfied and the total distance traveled is minimized. In other words, in the SDVRP the restriction that each customer is visited once is removed. The SDVRP is NP-hard, even under restricted conditions on the costs, when all vehicles have a capacity greater than two, while it is solvable in polynomial time when the vehicles have a maximum capacity of two (Dror and Trudeau (1989)).

Like other variants of the VRP, heuristics have been extensively applied to the SDVRP. Dror and Trudeau (1989) proposed a local search to solve the routing problem with split deliveries. This is a two-stage algorithm that first constructs a feasible VRP solution and from this generates a feasible SDVRP solution if split deliveries improve the initial VRP solution. The first stage uses three subroutines: (i) an initial route generator based on the algorithm of Clarke and Wright (1964), (ii) a node interchange based on a one-node and two-node swap, and (iii) a route improvement based on a 2-opt procedure. The second stage uses: (i) a 2-split interchange and (ii) a route addition routine. Given a demand point, the 2-split creates a neighborhood with all the possible alternatives that remove the demand point and insert it into two other routes whose combined spare capacity is sufficient for the demand. At each iteration, the candidate with the highest saving is selected and the search terminates when improvements cease. After this local search, a route addition routine creates new routes to eliminate split deliveries as long as a reduction in the total routing cost is obtained. Frizzell and Giffin (1992) developed construction and improvement heuristics for the VRPSD with grid network distances. Mullaseril et al. (1997) described a feed distribution problem encountered on a cattle ranch in Arizona. The problem is cast as a collection of capacitated rural postman problem with time windows and split deliveries. They presented an adaptation of the heuristics proposed by Dror and Trudeau (1989).

Meta-heuristics have been also proposed to solve the SDVRP. Ho and Haugland (2004) developed a tabu search to solve instances of the SDVRP with time windows. They constructed an initial solution by checking customers in sequence and appending the nearest unrouted customer to the latest routed customer in feasible routes. If the customer demand exceeds the capacity, the current route is deemed full loaded, the demand is then split, and a new route is created to supply the remaining demand. Once the route schedule is constructed, the TS commences. Each iteration, the best feasible candidate among four neighbourhoods is selected and the neighbouring solution is evaluated for improvements. The neighbourhoods examined are: relocating a customer between routes, eliminating a split delivery between two routes and introducing a new delivery between the same two routes, exchanging two customers between two routes, and performing a 2-opt operation between two routes. Archetti et al. (2006) proposed a tabu search to solve the SDVRP where a customer is removed from a set of routes serving it and inserted into a new route or into an existing route that has spare capacity. The scheme of the procedure employs an initial solution procedure, a tabu search, and an improvement phase.

The literature survey presented in this chapter supports the general statement made in Chapter 1 that the MAVRPs are among the VRP variants which did not receive an adequate degree of attention and the solution algorithms proposed to solve the MAVRPs are scarce. Moreover, solution methodologies which solve the MAPVRPs as a whole by simultaneously considering all its attributes are scarcer. To contribute toward addressing these two challenges, we prepared three different papers, each studying a class of MAVRPs often found in reality and proposing a sophisticated solution method to efficiently solve it.

Chapter 3

A Path Relinking Algorithm for a multi-depot periodic vehicle routing problem

Abstract

In this paper, we consider a variant of vehicle routing problems which is characterized by the presence of a homogeneous fleet of vehicles, multiple depots, multiple periods and two kinds of constraints that are often found in reality, i.e., vehicle capacity and route duration constraints. The objective is to minimize total travel costs. Since the Vehicle Routing Problem has been proved to be NP-hard in the strong sense, an effective Path Relinking Algorithm (PRA) is designed for finding the best possible solutions to this problem. The proposed PRA incorporates several purposeful exploitation and exploration strategies that enable the algorithm to tackle the problem in two different settings: 1) As a stand-alone algorithm, and 2) As a part of a co-operative search algorithm called Integrative Co-operative Search (ICS). The performance of the proposed Path Relinking Algorithm is evaluated, in each of the above ways, based on various test problems. The computational results show that the developed PRA performs well, in both solution quality and computational efficiency.

Keywords: Vehicle routing problems, Path Relinking, Integrative co-operative search.

3.1 Introduction

The vehicle routing problem (VRP), introduced by Dantzig and Ramser (1959), is one of the most important and widely studied combinatorial optimization problems, with many real-life applications in distribution and transportation logistics. In the classical VRP, a homogeneous fleet of vehicles services a set of customers from a single distribution depot or terminal. Each vehicle has a fixed capacity that cannot be exceeded and each customer has a known demand that must be fully satisfied. Each customer must be serviced by exactly one visit of a single vehicle and each vehicle must depart from the depot and return to the depot (Toth and Vigo (2002)).

Several variations and specializations of the vehicle routing problem, each reflecting various real-life applications, exist. However, surveying the literature, one can notice that not all VRP variants have been studied with the same degree of attention in the past five decades. This is the case for the problem considered in this study. Moreover, most of the methodological developments target a special problem variant, the Capacitated VRP (CVRP) or the VRP with Time Windows (VRPTW), despite the fact that the majority of the problems encountered in real-life applications display more complicating attributes and constraints. This also applies to the problem addressed in this paper.

Our objective is to contribute toward addressing these two challenges. In this paper, we address a variant of the VRP in which a daily plan is computed for a homogeneous fleet of vehicles that depart from different depots and must visit a set of customers for delivery operations in a planning horizon. In this VRP, we consider maximum route duration constraint and an upper limit of the quantity of goods that each vehicle can transport. Moreover, the cost of each vehicle route is computed through a system of fees depending on the distance that is traveled. This type of vehicle routing problem is typically called the Multi-depot Periodic Vehicle Routing Problem (MDPVRP).

To tackle the MDPVRP, we propose a new Path Relinking Algorithm, which incorporates exploitation and exploration strategies allowing the algorithm to solve the considered problem in two different manners: 1) As a stand-alone algorithm, and 2) As a part of a cooperative search method named as Integrative Cooperative Search (ICS).

The remainder of this paper is organized as follows. The problem statement is introduced in Section 3.2. The literature survey relevant to the topic of this paper is presented in Section 3.3. Section 3.4 deals with the proposed Path Relinking Algorithm. The experimental results are reported in Section 3.5. Finally, Section 3.6 provides conclusions and the evaluation of the work.

3.2 Problem statement

In this section, we formally state the MDPVRP, introducing the notations used throughout this paper. The MDPVRP can be defined as follows (Vidal et al. (2012a)): Consider an undirected graph $G(V,E)$. The node set V is the union of two subsets $V = V_C \cup V_D$, where $V_C = \{v_1,...,v_n\}$ represents the customers and $V_D = \{v_{n+1},...,v_{n+m}\}$ includes the depots. With each node $i \in V_C$ is associated a deterministic demand q_i. The edge set E contains an edge for each pair of customers and for each depot-customer combination. There are no edges between depots. With each edge $(v_i, v_j) \in E$ is associated a travel cost c_{ij}. The travel time for arriving to node j from node i (t_{ij}) is considered equal to c_{ij}. A limited number (K) of homogeneous vehicles of known capacity Q is available at each depot. Moreover, the MDPVRP has a planning horizon, say T periods. Each customer i is characterized by a service frequency f_i, stating how often within these T periods this customer must be visited and a list L_i of possible visit-period combinations, called patterns. Each vehicle performs only one route per period and each vehicle route must start and finish at the same depot while the travel duration of the route should not exceed D. The MDPVRP aims to design a set of vehicle routes servicing all customers, such that vehicle-capacity and route-duration constraints are respected, and the total travel cost is minimized.

3.3 Literature review

In this section, we focus our attention on reviewing papers previously published in the literature to address the MDPVRP. The objective of this review is first to present what types of solution methodologies have been proposed to solve the considered problem and second, to distinguish leading solution approaches that have been proved to be efficient to tackle the MDPVRP.

By surveying the literature, one notices that the most common solution approach for solving this type of the VRP is to apply a successive-optimization approach which sequentially solves a series of particular cases instead of considering the problem as a whole. This procedure usually leads to suboptimal solutions. Solution algorithms, belonging to this category, can be divided into two groups, i.e., exact methods and heuristics. To the best of our knowledge, the only exact method used to solve the MDPVRP was the one designed by Mingozzi (2005). In the proposed method, first, an integer programming model which is an extension of the set partitioning formulation of the CVRP is described. Then, an exact method for solving the problem, which uses variable pricing in order to reduce the set of variables to more practical proportions, is proposed. The pricing model is based on the bounding procedure for finding near optimal solutions of the dual problem of the LP relaxation of the proposed integer programming model. The bounding procedure is an additive procedure that determines a lower bound on the MDPVRP as the sum of the dual solution costs obtained by a sequence of five different heuristics for solving the dual problem, where each heuristic explores a different structure of the MDPVRP. Three of these heuristics are based on relaxations, whereas the two others combine subgradient optimization with column generation. We are also aware of three heuristic algorithms in this category. Hadjiconstantinou and Baldacci (1998) addressed the Multi-Depot Periodic VRP with Time Windows (MDPVRPTW). The authors proposed a multi-phase optimization problem and solved it using a four-phase algorithm. They developed a tabu search algorithm which solves the VRPTW and improved the solutions obtained during the optimization process using a 3-opt procedure. The last phase is the only one that modifies the depot and visit combination pattern assignments. Kang et al. (2005) studied the problem considered by Hadjiconstantinou and Baldacci (1998). The authors developed a two-phase solution method in which all feasible schedules are generated from each depot for each period and the set of routes are determined by solving the shortest path problem. Parthanadee and Logendran (2006) also solved the problem considered by Hadjiconstantinou and Baldacci (1998) using a tabu search. In this algorithm, all the initial assignments are built by cheapest insertion. At the improvement phase, depot and delivery pattern interchanges are used.

Another type of solution approaches more recently used to solve the MDPVRP target the problem as a whole by simultaneously considering all its characteristics. Crainic et al. (2009) proposed a well structured parallel cooperative search method, called Integrative Co-operative Search (ICS), to solve combinatorial optimization problems. The proposed ICS framework relies on an attribute decomposition approach and its structure is similar to a self-adaptive evolutionary meta-heuristic evolving several independent populations, where one population corresponds to the solutions of the main problem whereas the others consist of the solutions addressing specific dimensions of the problem. The authors used the MDPVRP with time windows to illustrate the applicability of the developed methodology. Vidal et al. (2012a) proposed a hybrid Genetic Algorithm (GA) to solve the MDPVRP and two of its special cases, i.e., the Multi-depot VRP (MDVRP) and the Periodic VRP (PVRP). The most interesting feature of the proposed GA is a new population-diversity management mechanism which allows a broader access to reproduction, while preserving the memory of what characterizes good solutions represented by the elite individuals of the population.

This brief review supports the general statement made in Section 3.1 that the MDPVRP is among the VRP variants which did not not receive an adequate degree of attention and the solution algorithms proposed to solve the MDPVRP are scarce. Moreover, solution methodologies which solve the MDPVRP as a whole by simultaneously considering all its characteristics are scarcer. To contribute toward addressing these two challenges, we develop a Path Relinking Algorithm to efficiently address the MDPVRP as a whole. The proposed Path Relinking Algorithm is described in the next section.

3.4 The Path Relinking Algorithm (PRA)

In recent years, meta-heuristic algorithms, especially population-based ones, have been applied with success to a variety of hard optimization problems. Among the population-based meta-heuristics, PRA is known as a powerful solution methodology which solves a given problem using purposeful and non-random exploration and exploitation strategies (Glover et al. (2000)). The general concepts and principles of a Path Relinking are first described in Section 3.4.1. Then, the main components of PRA proposed to solve the MDPVRP are explained in details in Section 3.4.2.

3.4.1 The Path Relinking Algorithm in general

The Path Relinking Algorithm has been suggested as an approach to integrate intensification and diversification strategies in the context of tabu search (Glover et al. (2000)). PRA can be considered as an evolutionary algorithm where solutions are generated by combining elements from other solutions. Unlike other evolutionary methods, such as genetic algorithms, where randomness is a key factor in the creation of offsprings from parent solutions, Path Relinking systematically generates new solutions by exploring paths that connect elite solutions. To generate the desired paths, an initial solution and a guiding solution are selected from a so-called reference list of elite solutions to represent the starting and the ending points of the path. Attributes from the guiding solution are gradually introduced into the intermediate solutions, so that these solutions contain less characteristics from the initial solution and more from the guiding solution as one moves along the path.

Based on the description mentioned above, the main components of the general Path Relinking Algorithm are summarized as follows:

1. Rules for building the reference set
2. Rules for choosing the initial and guiding solutions
3. A neighbourhood structure for moving along paths

Algorithm 1 shows a simple Path Relinking procedure presenting how the above-mentioned different components interact.

Algorithm 1 The general Path Relinking Algorithm

```
1: Generate a starting set of solutions.
2: Designate a subset of solutions to be included in the reference list.
   While the cardinality of the reference list $> 1$
     • Select two solutions from the reference list;
     • Identify the initial and guiding solutions;
     • Remove the initial solution from the reference list;
     • Move from the initial toward the guiding solution, generating intermediate solutions.
     • Update the reference list;
3: Verify stopping criterion: Stop or go to 1.
```

3.4.2 The proposed Path Relinking Algorithm

General idea

The Path Relinking Algorithm proposed in this paper relies on an easy-to-build and efficient reference list evolving several independent subsets, where one subset, called complete set, corresponds to elite solutions of the main problem while the others, named as partial sets, consist of elite solutions addressing specific dimensions of the problem. The cooperation between the sets of the reference list is set up by means of information exchange, through the searching mechanism of PRA.

To construct such a reference list, the MDPVRP is first decomposed into two VRPs with fewer attributes, i.e., PVRP and MDVRP, by respectively fixing the attributes ”multiple depots” and ”multiple periods”. Each of the constructed sub-problems is then solved by a dedicated solution algorithm which is called partial solver. The main advantage of applying such a decomposition procedure is that working on selected attribute subsets, instead of considering all attributes at a time, provides relatively high-quality solutions rapidly. Furthermore, well-known solution methodologies found in the literature may be used to solve sub-problems.

Elite solutions obtained by each partial solver are sent to a partial set of the reference list.

The partial sets can be either kept unchanged in the course of PRA or iteratively updated in order to include better solutions, in terms of both solution quality and diversification level, as the algorithm reaches the termination criterion. We respectively call these two possibilities as static and dynamic scenarios. Challenges, advantages and deficiencies of each scenario are thoroughly discussed in Section 3.5.

The construction of the reference list is finalized by considering the complete set as an empty list at the beginning of the optimization procedure. The complete set is subject to be repeatedly updated by high-quality and diverse solutions found through the searching mechanism of PRA.

After constructing the initial reference list, the proposed Path Relinking Algorithm starts to construct high-quality solutions of the main problem by exploring trajectories that connect solutions selected from different subsets, partial and/or complete, of the reference list. Towards this end, several selection strategies, each choosing initial and guiding solutions using a different strategy, are first implemented. Then, for each selected initial and guiding solutions, a neighbourhood search generates a sequence of high-quality complete solutions using the information shared by the selected solutions.

Two special variants of the proposed Path Relinking Algorithm explained above can be obtained by respectively ignoring complete and partial sets. In the former case, PRA generates complete solutions only based on the information gathered from partial solutions, while, in the latter case, the developed algorithm is converted to a general Path Relinking whose main characteristics have been described in Section 3.4.1.

Note that, throughout this paper, we use the term ”partial” for the solutions obtained by the partial solvers only in order to distinguish between these solutions and the solutions generated in the Path Relinking Algorithm and it does not imply that these solutions are not complete and feasible for the main problem, i.e., MDPVRP. Different components of the proposed Path Relinking Algorithm are described in the following subsections.

Search space

It is well known in the meta-heuristic literature that allowing the search to enter infeasible regions may result in generating high-quality and diverse feasible solutions. One of the main characteristics of the proposed PRA is that infeasible solutions are allowed throughout the search. Let us assume that x denotes the new solution generated by the searching mechanism. Moreover, let $c(x)$ denote the travel cost of solution x, and let $q(x)$ and $t(x)$ denote the total violation of the load and duration constraints, respectively. Solution x is evaluated by a cost function $z(x) = c(x) + \alpha q(x) + \beta t(x)$, where α and β are self-adjusting positive parameters. By dynamically adjusting the values of these two parameters, this relaxation mechanism facilitates the exploration of the search space and is particularly useful for tightly constrained instances. Parameter α is adjusted as follows: if there is no violation of the capacity constraints, the value of α is divided by $1+\gamma$, otherwise it is multiplied by $1+\gamma$, where γ is a positive parameter. A similar rule applies also to β with respect to route duration constraint.

Solution representation

One of the most important decisions in designing a meta-heuristic lies in deciding how to represent solutions and relate them in an efficient way to the searching space. Representation should be easy to decode to reduce the cost of the algorithm. In the proposed Path Relinking Algorithm, a path representation based on the method proposed by Kytöjoki et al. (2007) is used to encode the solution of the MDPVRP. The idea of the path representation is that the customers are listed in the order in which they are visited. To explain this solution representation, let us consider the following example: Suppose that there are four customers numbered 1-4 which have to be visited by two depots in two periods. Moreover, let us assume that the two first customers are served by the first depot, whereas the two last ones are visited by the second depot. Besides that, all customers need to be visited in each period. Figure 3.1 shows how a solution of the problem described above is represented.

As depicted in Figure 3.1, in this kind of representation, a single row array of the size equal to $n+1$ is generated for each depot in each period. Note that n is the number of customers to

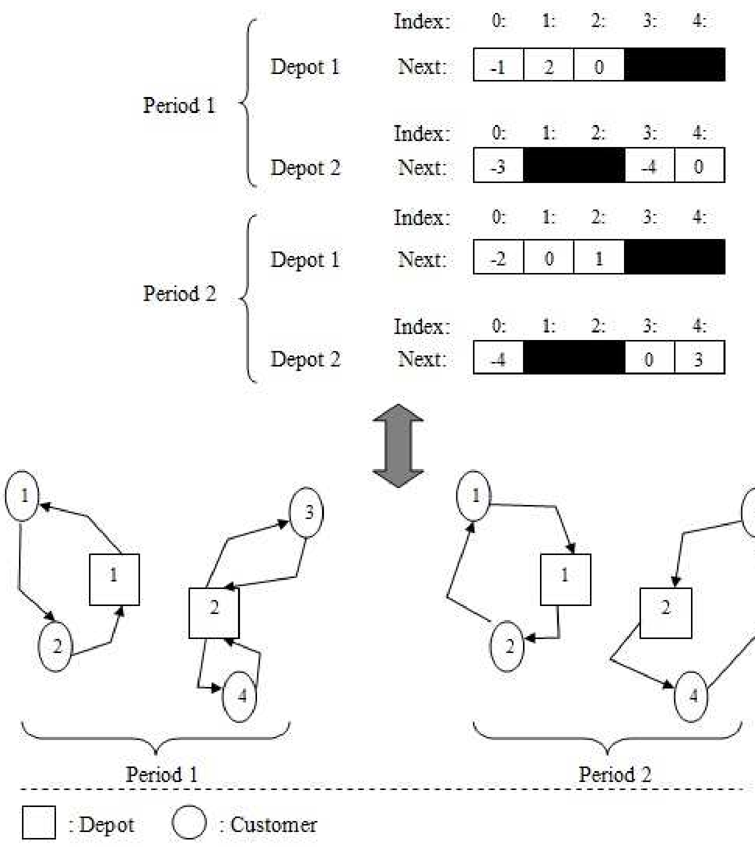


Figure 3.1: An example of the solution representation

be visited. The first position of the array (index 0) is related to the corresponding depot, while each of the other positions (index i; $1 \leq i \leq n$) represents a customer. The value assigned to a position of the array represents which customer should be immediately visited after the customer or depot related to that position. For example, in Figure 3.1, the value ”2” has been assigned to the second position (index 1) of the first array. It means that the second customer is immediately visited after the first customer by a vehicle departed from the first depot. In this path representation, negative values indicate the beginning of a new route, 0 refers to the end of the routes and a vacant position (drawn as a black box in Figure 3.1) reveals that the customer corresponding to that position is not served by the depot to which the array belongs. Using this representation, changes to the solution can be performed very quickly. For example, the insertion of a new customer k between two adjacent customers a and b is done simply by changing the ”next-values” of k to b and a to k. Similarly, one can delete a customer or reverse part of a route very quickly (Kytöjoki et al. (2007)).

Constructing the initial reference list

The reference list is a collection of high-quality solutions that are used to generate new solutions by way of applying the searching mechanism of the Path Relinking Algorithm. What solutions are included in the reference list, how good and how diversified they are, have a major impact on the quality of the new generated solutions (Ghamlouche et al. (2004)). Based on the descriptions mentioned at the beginning of this section, the reference list implemented in PRA consists of three different subsets where the first two subsets are the partial sets, each keeping elite partial solutions generated by a dedicated partial solver, while the last subset is the complete set consisting of elite solutions of the main problem. Note that, in the proposed algorithm, the maximum size of each subset is fixed to a predetermined value shown by R_{max}.

For the sake of the following descriptions, let us define first the following notations:

- Φ^i: the set of partial solutions added to the *i*th partial set of the reference list,
- Ψ^i: the set of whole partial solutions generated by the *i*th partial solver,
- ϕ^i_j: the *j*th partial solution of Φ^i,
- ψ^i_k: the *k*th solution of Ψ^i.

The construction of the initial reference list starts by adding R_{max} elite partial solutions existing in Ψ^i $(i = 1, 2)$ to the *i*th partial set of the reference list using the following strategy whose main aim is to ensure both the quality and diversity of the preserved solutions:

1. First, fill partially the *i*th partial set $(i = 1, 2)$ with $\lceil R_{max}/2 \rceil$ partial solutions of Ψ^i which have the best objective function values. Then, delete the added solutions from Ψ^i. We call this part of the partial set as B_1.
2. Define $\Delta(\phi^i_j, \psi^i_k)(\phi^i_j \in \Phi^i, \psi^i_k \in \Psi^i; \forall i, j, k)$ as the Hamming distance of the *j*th partial solution existing in Φ^i to the *k*th remaining partial solution of Ψ^i.
3. Calculate $d^{\Phi^i}_k = \min_{\phi^i_j \in \Phi^i} \Delta(\phi^i_j, \psi^i_k)(\forall \psi^i_k \in \Psi^i)$.
4. Sort the solutions of Ψ^i $(i = 1, 2)$ in descending order of $d^{\Phi^i}_k$.
5. Extend the *i*th partial set of the reference list $(i = 1, 2)$ with the first $\lfloor R_{max}/2 \rfloor$ solutions of Ψ^i. This part of the partial set is named as B_2.

As mentioned at the beginning of Section 3.4.2, the construction of the reference list is done by considering its last subset (the complete set) as an empty list which is gradually filled up by elite complete solutions generated during the Path Relinking Algorithm.

The reference list update method

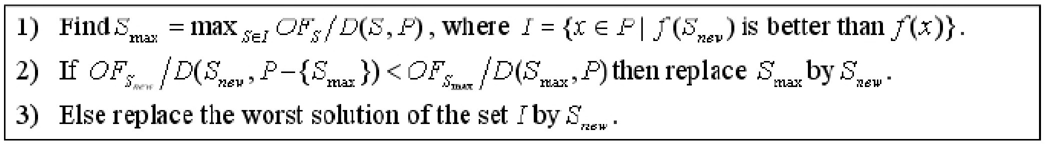
The constructed reference list is iteratively updated during the Path Relinking Algorithm in order not to lose high-quality and diverse solutions. Unlike the general Path Relinking Algorithm in which the reference list is updated only when a new solution is generated, in the proposed PRA, two different kinds of updating method are independently applied as follows: The first type of updating method, called Internal Update Method (IUM), occurs whenever a high-quality complete solution is generated by the searching mechanism of the Path Relinking Algorithm. In IUM, once a feasible complete solution, S_{new}, is generated, it is directly added to the complete set of the reference list if the number of elite complete solutions preserved in this set is less than R_{max}; otherwise, the following replacement strategy is implemented. We first define the diversity contribution of the complete solution S_{new} to the complete set of the reference list shown by *P*, $D(S_{new}, P)$, as the similarity between itself and its nearest neighbour in the complete set, that is:

$$D(S_{new}, P) = \min_{X \in P, X \neq S_{new}} \Delta(S_{new}, X)$$

where $\Delta(S_{new}, X)$, as mentioned in the previous section, is the Hamming distance. Moreover, let us define $OF_{S_{new}}$ as the objective function value of S_{new}. The replacement strategy schematically shown by Figure 3.2 is implemented in three phases as follows: Firstly, the replacement strategy considers all the complete solutions of the complete set with poorer objective function values than S_{new} and finds the one, S_{max}, which maximizes the ratio of (*objective function value*)/(*diversity contribution*) (Step 1). Then, the new generated solution, S_{new}, replaces S_{max} if the following inequality holds (Step 2):

$$OF_{S_{new}}/D(S_{new}, P - S_{max}) < OF_{S_{max}}/D(S_{max}, P)$$

In this way, we introduce into the complete set a solution with better objective function value and possibly higher contribution of diversity. If the inequality mentioned in the second step does not hold, the worst solution of the set determined in the first step is replaced by S_{new} (Step 3).



1) Find $S_{\max} = \max_{S \in I} OF_S / D(S, P)$, where $I = \{x \in P \mid f(S_{new}) \text{ is better than } f(x)\}$.
2) If $OF_{S_{new}} / D(S_{new}, P - \{S_{\max}\}) < OF_{S_{\max}} / D(S_{\max}, P)$ then replace $S_{\max}$ by S_{new}.
3) Else replace the worst solution of the set I by S_{new}.

Figure 3.2: Proposed replacement strategy

On the other hand, the second type of the updating method, called External Update Method (EUM), occurs for the *i*th partial set of the reference list ($i = 1, 2$) whenever a new partial solution is obtained by the *i*th dedicated partial solver. As previously mentioned, the *i*th partial set of the reference list ($i = 1, 2$) consists of a set of high-quality solutions B_1 and a set of diverse solutions B_2. Suppose a new partial solution, x_{new}, is obtained by the *i*th partial solver. EUM updates the corresponding subset of the reference list as follows: First, x_{new} is examined in terms of solution quality. If it is better than the worst existing solution in B_1, the latter is replaced by the former. Otherwise, x_{new} is assessed in terms of solution diversification. In this case, x_{new} is added to the list if it increases the distance of B_2 to B_1. In other words, if the minimum Hamming distance of x_{new} to any solution in B_1 is greater than the minimum Hamming distance of the last existing solution in B_2 to any solution in B_1, the last solution in B_2 is replaced by x_{new} and all the existing solutions of B_2, including x_{new}, are sorted again.

The main purpose of implementing two different update methods is to simultaneously maintain the elite partial and complete solutions generated respectively by the partial solvers and Path Relinking Algorithm.

Choosing the initial and guiding solutions

The performance of the Path Relinking Algorithm is highly dependent on how the initial and guiding solutions are selected from the reference list (Ghamlouche et al. (2004)). In the proposed Path Relinking Algorithm, four different strategies, each following a different purpose, are used to choose the initial and guiding solutions.

The first strategy called Partial Relinking Strategy (PRS) selects two partial solutions, each from a different partial set of the reference list, and sends them to the neighbourhood search phase to generate high-quality complete solutions. The main idea involved in implementing such a selection strategy is to produce complete solutions by integrating the best characteristics of the chosen partial solutions. Towards this end, the effect of four different sub-strategies, each generating R_{max} pairs of partial solutions, is investigated in order to choose the one having the most positive impact on the performance of PRA. These four sub-strategies are described as follows:

PRS_1 : The *i*th pair of the first sub-strategy is constructed by defining the guiding and initial solutions as the *i*th best solution of the jth ($j = 1, 2$) and kth ($k = 1, 2,\ k \neq j$) partial sets, respectively. This sub-strategy is motivated by the idea that high-quality solutions share some common characteristics with optimum solutions. One then hopes that linking such solutions yields improved new ones.

PRS_2 : The *i*th pair of the second sub-strategy is generated by determining the guiding solution as the *i*th best solution of the jth ($j = 1, 2$) partial set, while the initial solution is defined as the *i*th worst solution of the kth ($k = 1, 2,\ k \neq j$) partial set. The purpose of this sub-strategy is to improve the worst partial solution of a partial set based on the appropriate characteristics of a high-quality partial solution of the other partial set.

PRS_3 : The *i*th pair of the third sub-strategy is constructed by randomly choosing the guiding and initial solutions from the jth ($j = 1, 2$) and kth ($k = 1, 2,\ k \neq j$) partial sets, respectively. The aim of this sub-strategy is simply to select the initial and guiding solutions in a random manner with the hope of choosing those pairs of elite partial solutions which are not selected using the other sub-strategies explained in this section.

PRS_4 : The *i*th pair of the fourth sub-strategy is generated by defining the guiding solution as the *i*th best solution of the jth ($j = 1, 2$) partial set, whereas the initial solution is chosen as the solution of the kth ($k = 1, 2,\ k \neq j$) partial set with maximum Hamming distance from the guiding solution. The aim of the fourth sub-strategy is to select the initial and guiding solutions not only according to the objective function value but also according to a diversity, or dissimilarity criterion.

On the other hand, in the second strategy called Complete Relinking Strategy (CRS), two different high-quality complete solutions are selected from the complete set of the reference list as the source for constructing a path of new solutions. In other words, in CRS, trajectories that connect complete solutions generated by the Path Relinking Algorithm are explored to obtain other high-quality complete solutions. The main purpose of this strategy is to prevent losing good complete solutions which can be obtained by searching paths constructed between other complete solutions previously generated by the algorithm. Suppose that the number of existing complete solutions in the complete set is equal to Ω ($\Omega \leq R_{max}$). In CRS, the effect of the following three sub-strategies, each generating Ω pairs of complete solutions, is investigated.

CRS_1 : The *i*th pair of the first sub-strategy is constructed by defining the guiding and initial solutions as the best and *i*th complete solutions of the complete set, respectively. The main idea involved in this sub-strategy is to improve each of the existing complete solution based on appropriate characteristics of the best complete solution found by the Path Relinking Algorithm.

CRS_2 : The *i*th pair of the second sub-strategy is generated by determining the guiding and initial solutions as the *i*th and (*i*+1)th best solutions of the complete set, respectively. The idea behind this sub-strategy is exactly the same as the idea of implementing the first sub-strategy of PRS.

CRS_3 : The *i*th pair of the last sub-strategy is generated as follows: The guiding solution is selected as the *i*th best solution of the complete set, whereas the initial solution is chosen as the solution of the same set with maximum Hamming distance from the selected guiding solution.

The third strategy called Mixed Strategy (MS) selects two distinct partial and complete solutions as the inputs of the moving mechanism phase. Using this selection strategy, we hope to improve the selected partial solution based on good features of the chosen complete solution. In MS, the effect of two different sub-strategies is investigated. These two sub-strategies are explained as follows:

MS_1 : The *i*th pair of the first sub-strategy is constructed by defining the guiding and initial solutions as the *i*th best solution of the *j*th ($j = 1, 2$) and complete sets of the reference list, respectively.

MS_2 : The *i*th pair of the second sub-strategy is generated as follows: The guiding solution is selected as the best solution of the complete set, whereas the initial solution is chosen as the solution of the *j*th ($j = 1, 2$) partial set of the reference list with maximum Hamming distance from the selected guiding solution.

The last strategy is called Ideal Point Strategy (IPS). For the sake of the following description, let us first consider the following definition:

Definition 1. Ideal Point (IP) is a virtual point whose *i*th coordinate ($i = 1, 2$) is made by the objective function value of the best partial solution of the *i*th partial set ($i = 1, 2$).

IPS first selects two different guiding solutions so that the *i*th guiding solution is the solution kept in the *i*th coordinate of ideal point. Then, each of the solutions preserved in the reference list (partial or complete) serves respectively as the initial solution. The main purpose of choosing multiple guiding solutions is that promising regions may be searched more thoroughly in Path Relinking by simultaneously considering appropriate characteristics of multiple high-quality guiding solutions.

Neighbourhood structure and guiding attributes

In the proposed algorithm, unlike a general Path Relinking, two neighbourhood searches, each targeting a different goal, are implemented in parallel.

The first neighbourhood search is a memory-based searching mechanism which is done on each pair of partial solutions selected from the reference list using the partial relinking strategy. The aim of implementing such a neighbourhood search is to generate a sequence of high-quality complete solutions through integrating appropriate characteristics shared by the selected partial solutions.

As previously mentioned, the partial relinking strategy selects a solution (A) from the first partial set of the reference list, as either initial or guiding solution, while the other solution (B) is chosen from the second partial set. Each of the selected partial solutions shares two important kinds of information: 1) A depot assignment pattern which shows that each customer is assigned to what depot, and 2) A visit pattern which reflects that each customer is serviced in what periods of the horizon. Without loss of generality, let us suppose that the first partial set of the reference list contains elite partial solutions of the MDVRP, whereas the second partial set is made up of elite partial solutions of the PVRP. Consequently, the selected solution (A) is a solution that the partial solver obtained by fixing the attribute ”multiple periods” and by optimizing based on the attribute ”multiple depots”. Hence, it is reasonable to claim that in such a solution, each customer is assigned to a good depot, while there is no guarantee that the customers are visited based on good visit patterns. On the other hand, the chosen solution (B) is a solution that the other partial solver attained by fixing the attribute ”multiple depots” and by optimizing based on the attribute ”multiple periods”. Therefore, each customer in this solution is visited through a good visit pattern, while there is no guarantee that the customers are served by good depots.

Based on the descriptions mentioned above, we can deduce that the good characteristic of the selected solution (A) is that each customer is served by a good depot, while the appropriate characteristic of the chosen solution (B) is that each customer is served based on a good visit pattern. The following definitions reveal the major idea involved in the proposed neighbourhood search:

Definition 2. A customer is called eligible if it is visited: 1) by the depot to which that customer is assigned in the solution selected from the first partial set, and 2) based on the visit pattern through which that customer is served in the solution chosen from the second partial set.

Definition 3. A good complete solution generated by the neighbourhood search is a solution in which all the customers are eligible.

Therefore, the main purpose of the neighbourhood search is to progressively introduce the properties mentioned in Definition 2 to all the customers of the selected initial solution. Towards this end, we define an algorithm which is repeated θ iterations where θ is a predetermined positive value. At the *i*th iteration of the algorithm, a customer of the initial solution is randomly selected and its eligibility is investigated based on the properties of Definition 2. Note that, depending on the partial set from which the initial solution is selected, one of the criteria mentioned in Definition 2 is always met. For example, if the initial solution is selected from the first partial set, each of the customers is assigned to a good depot. Consequently, the first property is always satisfied for all the customers and, thus, the second property should only be verified for the eligibility of the chosen customer. If the second property is not met and the selected customer is served by a visit pattern different from its corresponding visit pattern in the guiding solution, it is considered as an ineligible customer. The neighbourhood search follows then one of the following situations:

1. **Eligible customer**: If the customer is eligible, the following operators are successively applied:
    - **Intra-route relocate operator**- In this operator, the eligible customer is first removed, on each period of its visit pattern, from the route by which it is visited. It is then re-inserted to the best position, based on the penalty function described at the beginning of this section, of the same route.
    - **Inter-route relocate operator**- In this operator, the chosen customer is first removed, on each period of its visit pattern, from its current route and, then, it is re-inserted to the best position of the other routes assigned to the depot by which the customer is served.
2. **Ineligible customer**: If the selected customer is ineligible, a neighbourhood search, based on the relocate operator, is applied to the solution in order to overcome the ineligibility of the customer. To implement the relocate operator-based neighbourhood search, the following four steps are done in a sequential manner:
    (a) The depot to which the selected customer is currently assigned is changed to the depot by which that customer is served in the solution selected from the first partial set.
    (b) The current visit pattern of the selected customer is changed to the visit pattern through which the customer is visited in the solution chosen from the second partial set.

(c) The customer is removed from the routes by which it is visited.

(d) Finally, on each period of the new visit pattern, the removed customer is re-inserted to one of the routes assigned to the new depot. Once again, the position to which the customer is inserted is the position in which the penalty function has the least value.

The neighbourhood search described above is equipped by a virtual memory whose aim is to enable the algorithm to search promising regions more thoroughly. Each element preserved in the implemented memory is represented by three indices (i, D^*, P^*), where i $(i = 1, 2...n)$ shows the customer's index, D^* and P^* represent, respectively, the depot and visit pattern based on which the *i*th customer is visited in the best solution generated so far by the Path Relinking. Suppose, in the course of the neighbourhood search, we select the *i*th customer which is an ineligible customer. To describe how the proposed memory works, let us consider the two following cases:

1. **The initial solution has been selected from the first partial set**: In this case, if the visit pattern through which the chosen customer is served is equal to P^*, the current visit pattern remains unchanged; otherwise, the visit pattern is changed to the one through which the customer is visited in the guiding solution.
2. **The initial solution has been chosen from the second partial set**: In this case, if the depot to which the selected customer is assigned is equal to D^*, the current depot is not changed; otherwise, the depot is changed to the one by which the customer is serviced in the guiding solution.

The main purpose of applying such a mechanism is to keep the structure of the selected solution as near as possible to the structure of the best solution obtained so far by the algorithm. This memory is updated when a new best solution is found and, to diversify search directions, the above rule is broken if the current best solution is not changed for ϵ iterations. Note that ϵ is a predetermined positive value.

The second neighbourhood search is another memory-based searching mechanism which explores trajectories connecting initial and guiding solutions selected through one of the other selection strategies, i.e., complete relinking, mixed or ideal point strategy. Like various neighbourhood searches implemented for the general Path Relinking Algorithm, the second neighbourhood search tries to gradually introduce best characteristics of either a single or multiple guiding solutions (based on the strategy used to select initial and guiding solutions) to new solutions obtained by moving away from the chosen initial solution. Similar to the neighbourhood search proposed above, the second neighbourhood is iterated θ times so that at each iteration, the eligibility of a randomly selected customer is investigated.

Definition of an eligible customer is different form what was given in the first neighbourhood search and is dependent on the strategy used to select initial and guiding solutions. Definition 4 represents the properties of an eligible customer in the cases where initial and single guiding solutions are selected using either partial relinking or mixed strategy.

Definition 4. A customer is called eligible if it is served based on the depot and visit pattern through which that customer is visited in the guiding solution.

On the other hand, Definition 5 shows the conditions under which a customer is called eligible if initial and multiple guiding solutions are chosen using ideal point strategy. Note that, in Definition 5, without loss of generality, we suppose that the first and second guiding solutions are respectively selected as the best solutions of the first and second partial sets.

Definition 5. A customer is called eligible if it is served: 1) by the depot to which that customer is assigned in the first guiding solution, and 2) based on the visit pattern through which that customer is served in the second guiding solution.

If the chosen customer is considered eligible, two operators described in the first neighbourhood search, i.e., inter- and intra-route relocate operators, are respectively implemented. Otherwise, to overcome the ineligibility of the chosen customer, a relocate operator-based neighbourhood search is applied. The proposed neighbourhood search removes first the customer from all the routes through which it is currently served. Then, one of the two following situations occurs:

- If the initial solution has been selected using either complete relinking or mixed strategy, the depot and visit pattern of the removed customer are respectively replaced by the depot and visit pattern based on which the customer is visited in the guiding solution.

- If the initial solution has been chosen using ideal point strategy, the depot and visit pattern of the removed customer are respectively changed to the depot and visit pattern of that customer in the first and second guiding solutions.

Finally, on each period of new visit pattern, the removed customer is inserted to one of the existing routes of new depot. Like the first neighbourhood search, the position to which the customer is inserted is the one in which the penalty function takes the least value.

Termination criterion

It is a condition that terminates the search process. In this paper, the two following stopping criteria are simultaneously considered:

- The algorithm is stopped if no improving solution is found for μ successive iterations. μ is a positive value which is determined at the beginning of the algorithm. Or,
- The algorithm is terminated if it passes a maximum allowable running time.

Skeleton of the proposed PRA

Algorithm 2 represents the skeleton of the Path Relinking Algorithm proposed for the MD-PVRP.

In Algorithm 2, C^* is defined as a list representing the best order of selection strategies to choose initial and guiding solutions at a given iteration. For example, if C^* is made up as $PRS_2 \rightarrow CRS_3 \rightarrow MS_1 \rightarrow IPS$, it means that, at a given iteration, initial and guiding solutions should be selected using the following order:

1. The second partial relinking sub-strategy.

2. The third complete relinking sub-strategy.

Algorithm 2 Path Relinking Algorithm

```
Initialize the search parameters.
Construct the initial reference list.
while the termination criterion is not met do
    Set $\alpha$=1, $\beta$=3.
    Set $\rho$=0.
    Update the reference list using the External Update Method (EUM).
    for $i$=1...$\|C^*\|$ do
        Set $j$= The $i$th element of $C^*$.
        repeat
            Select one initial solution, $S$, and one or multiple guiding solutions according to
            selection strategy $j$.
            Set $x$=$S$.
            Set $\upsilon$=0.
            repeat
                Select randomly a customer of $x$.
                Verify the eligibility of the selected customer.
                Generate a solution $\bar{x}$ using the neighbourhood search corresponding to the
                chosen selection strategy.
                If $\bar{x}$ is feasible, update the reference list using the Internal Update Method
                (IUM).
                Compute $q(.)$ and $t(.)$ and update $\alpha$ and $\beta$.
                Set $x$=$\bar{x}$.
                Increment $\upsilon$ by 1.
            until $\upsilon \leq \theta$.
            Increment $\rho$ by 1.
        until $\rho \leq R_{max}$.
    end for
end while
```

3. The first mixed sub-strategy.
4. The ideal point strategy.

Note that, in Section 3.5.2, we thoroughly describe how C^* is built.

3.5 Experimental results

In this section, the performance of the proposed Path Relinking Algorithm is investigated based on different test problems. The only problem instances existing in the literature are those proposed by Vidal et al. (2012a). The authors generated 10 problems whose characteristics are shown by Table 3.1.

Table 3.1: Problem instances

Instance	n	K	m	T
pr01	48	1	4	4
pr02	96	1	4	4
pr03	144	2	4	4
pr04	192	2	4	4
pr05	240	3	4	4
pr06	288	3	4	4
pr07	72	1	6	6
pr08	144	1	6	6
pr09	216	2	6	6
pr10	288	3	6	6

To prove the efficiency of the proposed PRA, two different scenarios, each investigating one special aspect of the algorithm, are independently followed. In the first scenario, called static scenario, the partial sets of the reference list, initially filled up by the dedicated partial solvers, remain unchanged during the algorithm. In such a scenario, we aim to study how PRA performs as a pure stand-alone algorithm without benefiting of the information that are shared by the partial solvers through updating the partial sets of the reference list. Towards this end, in each of the above problem instances, a feasible solution is first generated using the local search proposed by Vidal et al. (2012a). Let us denote the constructed solution by A. Then, the problem in hand is decomposed into two vehicle routing problems with exactly one less attribute, i.e., PVRP and MDVRP. In the PVRP, the attribute ”multiple depots” is considered fixed by assigning each customer to the depot by which it is served in solution A. On the other hand, in the MDVRP, the other attribute ”multiple periods” is set to be fixed by allocating each customer to the visit pattern through which it is visited in solution A. Thereafter, each of the above sub-problems is solved using the hybrid genetic algorithm proposed by Vidal et al. (2012a) to generate the required number of partial solutions. Finally, the obtained partial solutions are sent to the Path Relinking Algorithm in order to generate solutions of the main problem. Note that, the number of partial solutions fed to PRA is set to 20 and the above procedure is repeated in 10 different runs for each of the problem instances.

On the other hand, in Scenario 2, called dynamic scenario, the partial sets of the reference list are updated in the course of the optimization by partial solutions generated through the Integrative Cooperative Search (ICS) method designed by Crainic et al. (2009). To more precisely understand how this scenario is built, let us briefly describe the solution methodology used in the ICS. In the ICS approach, three fundamental questions are carefully answered: how to decompose the problem at hand to define sub-problems; how to integrate partial solutions obtained from the decomposition phase to construct and improve solutions of the main problem and, finally, how to perform and guide the search. In the decomposition phase, the main problem is first decomposed into several sub-problems by fixing the values of given sets of attributes. The constructed sub-problems are then simultaneously solved by partial solvers which can be well-known constructive methods, heuristics, meta-heuristics or exact methods. The elite partial solutions obtained are sent to the central memory accompanied with context information (measures, indicators, and memories). Then, in order to construct whole solutions, integrators play their important role. integrators, which could be either exact methods or meta-heuristics, construct, and possibly improve, solutions to the main problem using solutions from the different partial solution sets. Finally, in order to repeatedly control the evolution of partial solvers and integrators implemented in the ICS approach, a guiding and controlling mechanism, namely global search coordinator, guides the global search by sending appropriate instructions to partial solvers and, eventually, integrators.

In the dynamic scenario, the proposed PRA, in fact, plays the same role as an integrator which works based on partial solutions generated during the optimization procedure of the ICS.

Towards this end, a modified version of the ICS method proposed by Lahrichi et al. (2012) is executed on each problem instance in 10 different runs. In each of the runs, the ICS is interrupted in four different snapshots, i.e., 5, 10, 15 and 30 minutes, and partial solutions obtained at each snapshot are served to update the partial sets of the reference list using the External Update Method (EUM). Note that, in this scenario, each partial set remains unchanged between two successive snapshots. The most distinguishable difference between these two scenarios is that, in the dynamic scenario, we examine how the quality of the proposed Path Relinking Algorithm is affected when better and more diversified partial solutions are eventually fed to the algorithm by the ICS solution methodology.

The proposed algorithm has been coded in C++ and executed on a Pentium 4, 2.8 GHz, and Windows XP using 256 MB of RAM. Different aspects of the experimental results are discussed as follows: In Section 3.5.1, we first use a well-structured algorithm to calibrate all the parameters involved in PRA, Then, in Section 3.5.2, we explore the impact of different combination of selection strategies, mentioned in Section 3.4.2, on the performance of PRA. Finally, experimental results, on the two considered scenarios, are given in Section 3.5.3.

3.5.1 Parameter setting

Like the most meta-heuristic algorithms, the proposed PRA relies on a set of correlated parameters. Table 3.2 provides a summary of all PRA parameters.

Table 3.2: Parameters of PRA

Symbol	Description
R_{max}	Maximum size of each subset of the reference list
α, β	Self-adjusting parameters in the penalty function
γ	Factor involved in updating the self-adjusting parameters
θ	Number of times that each neighbourhood search is iterated
ϵ	Number of iterations after which the memory rule is broken
μ	Maximum allowable number of non-improving iterations

There are various different methods in the literature to calibrate parameters used in heuristics and meta-heuristics. In this paper, we use the well-reported four-step calibration method of Coy et al. (2000) to tune the parameters used in the heuristic algorithm. Coy et al. (2000)

proposed a procedure based on statistical Design Of Experiments (DOE) and gradient descent to systematically calibrate parameter values. The proposed parameter setting procedure takes a small number of the problem instances from the entire problem set, finds high-quality parameter settings for each problem, and then combines the parameter settings to determine good parameter values for the entire set of problems. This procedure can be summarized in the following four steps that are implemented in a sequential manner:

Step 1- In this step, a subset of problems to analyze (analysis set) is chosen from the entire set of problems. The problems are selected so that most of the structural differences found in the problem set are represented in the analysis set.

Step 2- In this step, computational experience is used to select the starting level of each parameter, the range over which each parameter will be varied, and the amount by which each parameter should be changed. Towards this end, in this paper, a robust calibration method called Relevance Estimation and VAlue Calibration (REVAC) (Smith and Eiben (2010)) is used. Technically, REVAC is a heuristic generate-and-test method that is iteratively searching for the set of parameter vectors of a given EA with a maximum performance. In each iteration, a new parameter vector is generated and its performance is tested. Testing a parameter vector is done by executing the EA with the given parameter values and measuring the EA performance. A detailed explanation of REVAC can be found in Smith and Eiben (2010). 3.3 summarizes the results obtained using REVAC.

Table 3.3: Calibration results of REVAC

Symbol	Starting level	Range	Changing step
R_{max}	20	[20,40]	5
α, β	1,1	[1,5], [1,5]	1,1
γ	1	[1,4]	1
θ	n	[n, 5*n]	n/2
ϵ	5000	[5000,10000]	2500
μ	200000	[200000, 600000]	50000

Step 3- In this step, good parameter settings are selected for each problem in the analysis set using fractional factorial design and response surface optimization. In this paper, the half-factorial design is implemented for each selected analysis set.

Step 4- Finally, in this step, the parameter values obtained in the third step are averaged to obtain high-quality parameter values. 3.4 represents the selected value of each parameter as follows:

Table 3.4: Calibration results

Symbol	Description
R_{max}	30
α, β	1,3
γ	1
θ	4*n
ϵ	5000
μ	500000

3.5.2 Path Relinking selection strategies

We tested all combinations of selection strategies, mentioned in Section 3.4, in order to identify the best way to select initial and guiding solutions. The best combination is then used for the extensive experimental analysis of the Path Relinking Algorithm.

The same 10 problem instances used to calibrate the parameter settings are also used here. Moreover, each run is repeated 5 times. Thus, since there are 24 possible combinations of selection strategies (4 partial relinking strategies $\times$ 3 complete relinking strategies $\times$ 2 mixed strategies), a total of 1200 runs are performed. The performance of each combination of selection strategies is measured, in both the static and dynamic scenarios, as the average improvement in solution quality, compared to the best partial solution initially fed to the partial sets of the reference list. Note that, in the dynamic scenario, the best partial solution found at the first snapshot, 5 minutes, is used to compare the efficiency of all combinations. The comparative performances of all combinations of selection strategies, in the static and dynamic scenarios, are respectively presented in Tables 3.5 and 3.6.

Both of Tables 3.5 and 3.6 identify the combination of strategies PRS_4 (The forth Partial Relinking Sub-strategy), CRS_3 (The third Complete Relinking Sub-strategy) and MS_2 (The second Mixed Sub-Strategy) as offering the best results. This set of selection strategies is therefore retained for our experimental analyses. The choice of this combination confirms the

Table 3.5: Average improvement in the static scenario (%)

	PRS_1	PRS_2	PRS_3	PRS_4
(CRS_1, MS_1)	6.88	6.52	6.41	7.16
(CRS_1, MS_2)	7.50	7.31	7.12	7.82
(CRS_2, MS_1)	7.22	7.15	7.01	7.65
(CRS_2, MS_2)	7.71	7.44	7.29	8.07
(CRS_3, MS_1)	7.04	6.72	6.60	7.31
(CRS_3, MS_2)	7.64	7.50	7.34	8.49

Table 3.6: Average improvement in the dynamic scenario (%)

	PRS_1	PRS_2	PRS_3	PRS_4
(CRS_1, MS_1)	1.21	1.16	0.97	1.44
(CRS_1, MS_2)	1.60	1.39	1.30	1.79
(CRS_2, MS_1)	1.47	1.30	1.18	1.72
(CRS_2, MS_2)	1.71	1.58	1.41	1.90
(CRS_3, MS_1)	1.31	1.22	1.03	1.63
(CRS_3, MS_2)	1.80	1.59	1.44	2.12

importance of selecting initial and guiding solutions non-randomly and also not only according to the objective function value but also according to a diversity criterion.

3.5.3 Results on MDPVRP instances

Static scenario

We tested PRA on the problem instances described at the beginning of this section. For solving these problems, the maximum running time is set to 30 minutes. Table 3.7 summarizes the characteristics of partial solutions fed to PRA using the hybrid genetic algorithm.

In Table 3.7, SP_1 and SP_2 represent partial solutions sets generated for the MDVRP and the PVRP, respectively. $SP_1 + SP_2$ is the union of all partial solutions obtained by the hybrid genetic algorithm. Moreover, For each set, worse, average and best partial solutions on 10 runs are shown. Finally, the last column reveals the Best Known Solution (BKS) reported by Vidal et al. (2012a).

In each of the problem instances, we carefully answer to the following questions:

1. What percentage of the gap is there between the PRA's output and the BKS?

Table 3.7: Characteristics of partial solutions in the static scenario

Instance	SP_1			SP_2			SP_1+SP_2			BKS
	Worst	Average	Best	Worst	Average	Best	Worst	Average	Best	
pr01	2152.38	2121.03	2118.84	2371.34	2253.45	2247.3	2371.34	2238.96	2118.84	2019.07
pr02	3784.13	3764.12	3747.75	4743.42	4742.1	4741.48	4743.42	4253.11	3747.75	3547.45
pr03	4943.99	4879.25	4856.39	6882.52	6439.64	6338.26	6882.52	5659.45	4856.39	4480.87
pr04	5714.78	5629.4	5575.78	8804.4	8799.12	8794.91	8804.4	7214.26	5575.78	5134.17
pr05	6059.12	6033.76	5998.22	8329.88	8324.96	8323.32	8329.88	7179.36	5998.22	5570.45
pr06	7196.13	7243.17	7130.2	8406.22	8402.05	8372.31	8406.22	7822.61	7130.2	6524.92
pr07	4820.72	4802.57	4788.51	5422.49	5421.88	5421.77	5422.49	5112.23	4788.51	4502.02
pr08	6620.41	6610.04	6594.6	8902.65	8733.29	8421.24	8902.65	7671.67	6594.6	6023.98
pr09	8750.52	8726.48	8689.47	11790.6	11779.4	11770.3	11790.6	10252.94	8689.47	8257.80
pr10	10678.9	10653.9	10617.8	14476.3	14465.1	14459.8	14476.3	12559.5	10617.8	9818.42

Table 3.8: Average results on MDPVRP instances in the static scenario

	PRA				HGA			
Instance	Worst	Average	Best	Time (sec)	Average	Time (sec)	Gap to HGA (%)	Gap to BKS (%)
pr01	**2019.07**	**2019.07**	**2019.07**	30	**2019.07**	21	0	0
pr02	**3547.45**	**3547.45**	**3547.45**	124	**3547.45**	89	0	0
pr03	**4480.87**	**4480.87**	**4480.87**	495	4491.08	463.2	-22.73	0
pr04	5155.12	5149.64	**5134.17**	1389	5151.73	1326	-0.04	0.30
pr05	5672.71	5598.32	5579.43	1800	5605.60	1800	-0.12	0.50
pr06	6618.38	6568.79	6540.66	1800	6570.28	1800	-0.02	0.67
pr07	**4502.02**	**4502.02**	**4502.02**	203	4502.06	131	-0.01	0
pr08	6038.44	6027.51	**6023.98**	547	6029.58	478	-0.03	0.05
pr09	8341.09	8304.26	8268.88	1800	8310.19	1667	-0.07	0.56
pr10	9987.16	9963.55	9852.30	1800	9972.35	1800	-0.09	1.48
ASD of PRA		0.22%						
ASD of HGA		0.26%						

2. Can the proposed PRA compete with the state-of-the-art Hybrid Genetic Algorithm (HGA) of Vidal et al. (2012a), which has been proved as one of the most powerful solution methodologies to solve the MDPVRP?

3. How much is PRA capable of improving the gap between initial partial solutions and the BKS?

Table 3.8 shows the results dealing with the two first questions. In this table, the average results of 10 runs of PRA and HGA, in terms of solution quality and computational time, are reported in columns 2-6. Moreover, the average error gaps of PRA compared to the HGA and BKS are respectively shown in the last two columns. Finally, ASD is the Average Standard Deviation per instance that an algorithm has obtained. Note that, If each of the considered solution methods (PRA and HGA) give a result equal to the BKS, we indicate the corresponding value in boldface.

The results shown by Table 3.8 can be interpreted as follows:

1. The average error gap of PRA to the BKS is +0.36% which is very reasonable considering the problem complexity. PRA results vary clearly depending on the problem difficulty so that the average gap ranges from 0.00% to 1.18%. On four problems (pr01, pr02, pr03 and pr04), the algorithm seems to always converge toward the BKS, whereas problems pr08 to pr10, with larger number of depots and periods, seem particularly difficult

Table 3.9: An example of the Friedman test table

Instance	The average results obtained by PRA	The average results obtained by HGA
pr01	X_{11}	X_{12}
pr02	X_{21}	X_{22}
pr03	X_{31}	X_{32}
.	.	.
.	.	.
.	.	.

to tackle. Generally, the proposed Path Relinking Algorithm performs well compared to the BKS even for more challenging instances including a larger number of customers, depots and periods.

2. The average error gap existing between PRA and HGA is -2.31% which shows the superiority of PRA to produce better results. To statistically prove this superiority, the well-known Friedman test is used. The Friedman test is a non-parametric statistical test developed by the U.S. economist Milton Friedman. Similar to the parametric repeated measures ANOVA, it is used to detect differences in treatments across multiple test attempts. The procedure involves ranking each row (or block) together, then considering the values of ranks by columns. In this paper, the Friedman test table is presented as Table 3.9. In this table, the first column display instance identifier, while the two other columns show the average results respectively generated by PRA and HGA.

   The characteristics of Friedman test, implemented in this paper, are as follows:

   - **Assumptions**:
     - The results over problem instances are mutually independent (i.e., the results within one instance do not influence the results within other instance)
     - Within each problem instance, the objective functions can be ranked.
   - **Hypotheses**:
     - H_0: There is no significant difference between the outputs of PRA and HGA.
     - H_1: PRA performs better than HGA.
   - **Procedure**:

Table 3.10: Friedman test results

Friedman test's variable	Value
A_2	49
B_2	48.2
T_2	36> $f_{0.99,1,9}$

(a) Rank the results of the both algorithms (PRA and HGA) within each problem instance, giving 1 to the best and 2 to the worst. Let us define $R(X_{ij})$ as the rank assigned to Row i and column j of Table 3.9.

(b) Calculate the total summation of squared ranks, A_2, using the following formula:

$$A_2 = \sum_{i=1}^{10} \sum_{j=1}^{2} [R(X_{ij})]^2$$

(c) Compute the summation of the rank for each algorithm, $R_j = \sum_{i=1}^{10} R(X_{ij})$ for $j = 1, 2$ and calculate B_2:

$$B_2 = \frac{1}{10} \sum_{j=1}^{2} R_j^2$$

(d) The test statistic is given by:

$$T_2 = \frac{9(B_2-45)}{A_2-B_2}$$

(e) Reject H_0, at the level of significance 0.01, if T_2 is greater than the quantile of the F distribution with $K_1 = 1$ and $K_2 = 9$ degrees of freedom. Table 3.10 summarizes the results of the above five-step procedure.

As shown in Table 3.10, the test static (T_2) is greater than $f_{0.99,1,9}$. This result justifies that PRA performs significantly better than HGA to produce good results, in terms of solution quality and computational time.

On the other hand, Table 3.11 represents the results concerning with the third question. This table indicates how much PRA is able to improve the gap between partial solutions and the BKS.

As shown in Table 3.11, PRA is considerably powerful to decrease the gap existing between the BKS and partial solutions of all the sets. This fact, besides the results shown in Table 3.8, reveals that the proposed algorithm plays very well its role as a stand-alone algorithm to generate high-quality solutions of the considered MDPVRP.

Table 3.11: Average gap improvement to BKS in the static scenario (%)

	SP_1			SP_2			SP_1+SP_2		
Instance	Worst	Average	Best	Worst	Average	Best	Worst	Average	Best
pr01	6.60	5.05	4.94	17.45	11.61	11.30	17.45	8.08	4.94
pr02	6.67	6.11	5.65	33.71	33.68	33.66	33.71	19.89	5.65
pr03	10.33	8.89	8.38	53.60	43.71	41.45	53.60	26.03	8.38
pr04	10.90	9.34	8.60	71.08	71.08	71.30	71.08	40.21	8.60
pr05	6.94	7.82	7.52	47.70	48.95	49.26	47.70	28.39	7.52
pr06	8.85	10.33	9.04	27.40	28.10	28.07	27.40	19.21	9.04
pr07	7.08	6.68	6.36	20.45	20.43	20.43	20.45	13.56	6.36
pr08	9.66	9.67	9.47	47.54	44.91	39.80	47.54	27.29	9.47
pr09	4.96	5.11	5.09	41.77	42.08	42.40	41.77	23.60	5.09
pr10	7.04	7.03	7.80	45.72	45.84	46.92	45.72	26.43	7.80

Dynamic scenario

In the dynamic scenario, we try to properly answer to the same questions as mentioned in the previous section. Table 3.12 indicates the main characteristics of partial solutions generated by the ICS in different snapshots. In each of the problem instances, PRA is executed on partial solutions of each snapshot and the obtained results on 10 runs is reported in Table 3.13

The average error gap to the BKS is +0.25%, +0.20%, +0.17% and +0.12% at 5, 10, 15 and 30-min snapshot, respectively. These average error gaps reveal that the quality of the proposed PRA increases by gradually feeding better and more diversified partial solutions by the ICS. On the other hand, in all the snapshots, the values of error gaps seem reasonable considering the problem difficulty. On four problem instance (pr01, pr02, pr07 and pr08), PRA always traps, in all snapshots, on the best partial solution fed by the ICS. This phenomenon seems inevitable because, in each of these problems, there exists apparently no better solution than the BKS which is initially sent as a partial solution to PRA by the ICS. On two problems (pr03 and pr10), PRA obtained new best known solutions which are shown as boldface starred values in the table.

Moreover, the average error gap between PRA and HGA is -2.41%, -2.47%, -2.84% and -3.15% at 5, 10, 15 and 30-min snapshot, respectively. These average error gaps prove, once again, better performance of PRA to generate promising results, in terms of solution quality and computational efficiency. Note that, in this scenario, the average result generated by PRA,

Table 3.12: Characteristics of partial solutions in the dynamic scenario

		SP_1			SP_2			SP_1+SP_2		
Instance	Snapshot	Worst	Average	Best	Worst	Average	Best	Worst	Average	Best
pr01	5 min.	2053.43	2028.14	2019.17	2112.51	2044.05	2019.17	2112.51	2036.09	2019.17
	10 min.	2043.01	2020.19	2019.17	2044.07	2026.42	2019.17	2044.07	2032.27	2019.17
	15 min.	2033.19	2026.38	2019.17	2027.48	2024.26	2019.17	2027.48	2025.61	2019.17
	30 min.	2022.88	2121.83	2019.17	2023.97	2021.75	2019.17	2023.97	2021.92	2019.17
pr02	5 min.	3608.77	3558.96	3547.45	3611.28	3562.31	3547.45	3611.28	3567.14	3547.45
	10 min.	3595.14	3552.16	3547.45	3595.14	3550.51	3547.45	3595.14	3553.22	3547.45
	15 min.	3588.42	3550.68	3547.45	3559.66	3549.14	3547.45	3588.42	3550.09	3547.45
	30 min.	3565.31	3549.66	3547.45	3554.04	3548.23	3547.45	3565.31	3549.53	3547.45
pr03	5 min.	4721.48	4537.04	4481.94	4853.33	4486.88	4481.94	4853.33	4507.53	4481.94
	10 min.	4700.62	4503.37	4481.94	4850.66	4483.23	4481.94	4850.66	4493.28	4481.94
	15 min.	4564.74	4523.19	4480.87	4486.41	4483.34	4480.87	4564.74	4503.14	4480.87
	30 min.	4538.53	4514.19	4480.87	4484.35	4482.45	4480.87	4538.53	4495.91	4480.87
pr04	5 min.	5201.69	5188.06	5172.76	5248.72	5195.26	5175.77	5248.72	5192.59	5172.76
	10 min.	5170.82	5164.58	5149.05	5162.32	5161.02	5149.05	5170.82	5162.81	5149.05
	15 min.	5177.83	5168.42	5149.05	5239.55	5178.44	5149.05	5239.55	5173.47	5149.05
	30 min.	5442.74	5239.41	5144.45	5152.96	5149.75	5144.45	5442.74	5199.34	5144.45
pr05	5 min.	5958.50	5788.24	5603.28	5958.50	5762.19	5682.16	5958.50	5768.22	5603.28
	10 min.	5720.83	5687.28	5642.99	5683.15	5664.23	5642.99	5720.83	5665.12	5642.99
	15 min.	5714.57	5681.45	5642.99	5958.50	5773.21	5642.99	5958.50	5728.35	5642.99
	30 min.	5706.81	5653.23	5604.95	5717.69	5664.12	5604.95	5717.69	5658.92	5604.95
pr06	5 min.	7085.21	6675.21	6608.98	7047.29	6663.29	6608.98	7085.21	6669.53	6608.98
	10 min.	6772.95	6659.41	6589.88	6735.38	6648.21	6589.88	6772.95	6653.12	6589.88
	15 min.	6744.39	6638.79	6589.81	6735.38	6626.47	6589.81	6744.39	6630.25	6589.81
	30 min.	6594.50	6574.25	6567.66	6590.36	6571.19	6567.66	6594.50	6572.22	6567.66
pr07	5 min.	4638.60	4578.29	4502.02	4778.42	4589.23	4502.02	4778.42	4584.18	4502.02
	10 min.	4577.91	4538.25	4502.02	4517.79	4506.77	4502.02	4577.91	4527.51	4502.02
	15 min.	4577.91	4528.84	4502.02	4509.36	4504.87	4502.02	4577.91	4520.36	4502.02
	30 min.	4509.97	4504.93	4502.02	4504.45	4503.33	4502.02	4509.97	4503.66	4502.02
pr08	5 min.	6246.78	6167.32	6024.24	6577.04	6321.87	6024.24	6577.04	6244.25	6024.24
	10 min.	6246.78	6097.44	6023.98	6485.56	6299.41	6023.98	6485.56	6226.09	6023.98
	15 min.	6246.78	6077.29	6023.98	6069.12	6044.65	6023.98	6246.78	6060.43	6023.98
	30 min.	6246.78	6054.38	6023.98	6025.21	6024.46	6023.98	6246.78	6044.61	6023.98
pr09	5 min.	8570.64	8433.12	8326.58	8531.55	8417.17	8316.95	8570.64	8425.62	8316.95
	10 min.	8312.40	8304.99	8296.42	8305.65	8301.44	8296.42	8312.40	8302.78	8296.42
	15 min.	8307.94	8301.45	8296.09	8305.65	8299.52	8296.09	8307.94	8300.27	8296.09
	30 min.	8424.49	8349.51	8293.33	8300.34	8297.44	8293.33	8424.49	8324.66	8293.33
pr10	5 min.	12626.90	10857.44	10128.8	13340.10	11190.51	10128.8	13340.10	13152.27	10128.8
	10 min.	10489.30	10227.44	9993.94	10402.10	10200.36	9993.94	10489.30	10214.77	9993.94
	15 min.	10169.20	10134.98	9993.94	10059.70	10032.46	9993.94	10169.20	10081.33	9993.94
	30 min.	10091.90	10049.77	9993.94	12192.90	10104.71	9993.94	12192.90	10070.14	9993.94

Table 3.13: Average results on MDPVRP instances in the dynamic scenario

		PRA					
		Worst	Average	Best	Time	Gap to	Gap to
Instance	Snapshot				(sec)	HGA (%)	BKS (%)
pr01	5 min.	**2019.07**	**2019.07**	**2019.07**	15	0	0
	10 min.	**2019.07**	**2019.07**	**2019.07**	15	0	0
	15 min.	**2019.07**	**2019.07**	**2019.07**	15	0	0
	30 min.	**2019.07**	**2019.07**	**2019.07**	15	0	0
pr02	5 min.	**3547.45**	**3547.45**	**3547.45**	65	0	0
	10 min.	**3547.45**	**3547.45**	**3547.45**	65	0	0
	15 min.	**3547.45**	**3547.45**	**3547.45**	65	0	0
	30 min.	**3547.45**	**3547.45**	**3547.45**	65	0	0
pr03	5 min.	**4480.87**	**4480.87**	**4480.87**	242	-22.73	0
	10 min.	**4480.87**	**4480.87**	**4480.87**	242	-22.73	0
	15 min.	**4480.87**	**4479.29***	**4472.22***	242	-26.25	-0.03
	30 min.	**4480.87**	**4478.12***	**4472.22***	242	-28.86	-0.06
pr04	5 min.	5155.32	5148.42	**5134.17**	300	-0.07	0.27
	10 min.	5149.05	5148.09	**5134.17**	300	-0.07	0.27
	15 min.	5149.05	5147.81	**5134.17**	300	-0.07	0.27
	30 min.	5148.45	5147.75	**5134.17**	300	-0.07	0.26
pr05	5 min.	5597.12	5595.24	5581.10	300	-0.18	0.44
	10 min.	5603.28	5592.91	5581.10	300	-0.22	0.40
	15 min.	5596.73	5589.04	5581.10	343	-0.29	0.33
	30 min.	5594.94	5585.16	5581.10	343	-0.36	0.26
pr06	5 min.	6573.29	6560.44	6540.66	300	-0.14	0.54
	10 min.	6566.46	6547.08	6540.66	300	-0.35	0.33
	15 min.	6566.46	6546.19	6538.91	512	-0.37	0.32
	30 min.	6549.57	6541.80	6538.60	512	-0.43	0.25
pr07	5 min.	**4502.02**	**4502.02**	**4502.02**	101	-0.009	0
	10 min.	**4502.02**	**4502.02**	**4502.02**	101	-0.009	0
	15 min.	**4502.02**	**4502.02**	**4502.02**	101	-0.009	0
	30 min.	**4502.02**	**4502.02**	**4502.02**	101	-0.009	0
pr08	5 min.	**6023.98**	**6023.98**	**6023.98**	225	-0.09	0
	10 min.	**6023.98**	**6023.98**	**6023.98**	225	-0.09	0
	15 min.	**6023.98**	**6023.98**	**6023.98**	225	-0.09	0
	30 min.	**6023.98**	**6023.98**	**6023.98**	225	-0.09	0
pr09	5 min.	8312.14	8303.89	8268.88	300	-0.07	0.56
	10 min.	8292.23	8291.08	8268.88	300	-0.22	0.40
	15 min.	8292.23	8288.44	8268.88	494	-0.26	0.37
	30 min.	8286.01	8279.55	8268.88	494	-0.37	0.26
pr10	5 min.	9995.29	9890.22	9852.30	300	-0.82	0.73
	10 min.	9886.34	9875.78	9852.30	300	-0.97	0.58
	15 min.	9886.34	9856.30	**9811.3***	750	-1.16	0.39
	30 min.	9871.06	9839.07	**9811.3***	750	-1.36	0.21
	ASD of PRA	22% (5-min)	22% (10-min)	20% (15-min)	19% (30-min)		

in each problem instance and each snapshot, is equal or better than the corresponding value in the static scenario. Consequently, we do not need to use the Friedman test and the significant difference between PRA and HGA is automatically proved.

Table 3.14 reports the improvement percentage on the gap between the BKS and partial solutions initially sent to PRA. Note that, pr01, pr02, pr07 and pr08 are ignored in 3.14 because, as mentioned above, the ICS always sends, in these problems, the BKS as a partial solution to PRA.

Studying the results obtained in the static and dynamic scenarios, we deduce that the proposed PRA can be used as a competitive solution method in the both of its considered settings, i.e., as a stand-alone algorithm and as an integrator in the ICS solution methodology.

3.6 Conclusions

This paper presented a new Path Relinking Algorithm to efficiently tackle the multi-depot periodic vehicle routing problem, for which few efficient algorithms are currently available. The proposed algorithm was designed based on prominent exploration and exploitation strategies which enable the algorithm to solve the problem in two different ways: 1) As a pure stand alone algorithm, and 2) As an integrator in the ICS solution framework.

To validate the efficiency of PRA, different test problems, existing in the literature, were solved. The computational results revealed that the proposed Path Relinking Algorithm performs considerably well, in all the problem instances.

The proposed PRA, as a fairly general-purpose solver, opens the way to experimentations and sensitivity analyses of local search and meta-heuristic components on a wide range of structurally different problems. Finally, perspective of research involve the generalization of the method towards a wider variety of multi-objective and stochastic settings.

Table 3.14: Average gap improvement to BKS in the dynamic scenario (%)

Instance	Snapshot	SP_1			SP_2			SP_1+SP_2		
		Worst	Average	Best	Worst	Average	Best	Worst	Average	Best
pr03	5 min.	5.37	1.25	0.02	8.31	0.13	0.02	8.31	0.59	0.02
	10 min.	4.90	0.50	0.02	8.25	0.05	0.02	8.25	0.28	0.02
	15 min.	1.87	0.98	0.19	0.12	0.09	0.19	1.87	0.53	0.19
	30 min.	1.29	0.80	0.19	0.08	0.09	0.19	1.29	0.44	0.19
pr04	5 min.	0.90	0.77	0.46	1.82	0.86	0.49	1.82	0.81	0.46
	10 min.	0.42	0.32	0.15	0.26	0.28	0.15	0.42	0.30	0.15
	15 min.	0.56	0.41	0.13	1.79	0.62	0.13	1.76	0.51	0.13
	30 min.	5.81	1.82	0.03	2.06	0.05	0.03	5.81	0.93	0.03
pr05	5 min.	6.49	3.50	0.38	6.49	3.02	1.80	6.49	3.26	0.38
	10 min.	2.11	1.74	1.12	1.43	1.33	1.12	2.11	1.53	1.12
	15 min.	2.11	1.69	1.12	6.49	3.34	1.12	6.49	2.51	1.12
	30 min.	2.01	1.28	0.44	2.20	1.46	0.44	2.20	1.37	0.44
pr06	5 min.	7.84	1.80	1.03	7.26	1.59	1.03	7.84	1.70	1.03
	10 min.	3.16	1.74	0.75	2.59	1.61	0.75	3.16	1.68	0.75
	15 min.	2.73	1.44	0.80	2.61	1.26	0.80	2.75	1.35	0.80
	30 min.	0.69	0.50	0.44	0.64	0.48	0.44	0.72	0.49	0.44
pr09	5 min.	3.13	1.60	0.40	2.66	1.41	0.28	3.13	1.50	0.28
	10 min.	0.24	0.21	0.10	0.17	0.18	0.10	0.24	0.20	0.10
	15 min.	0.19	0.17	0.12	0.16	0.15	0.14	0.18	0.16	0.12
	30 min.	1.68	0.87	0.24	0.16	0.24	0.28	1.65	0.56	0.24
pr10	5 min.	26.80	9.12	2.78	34.07	12.51	2.77	34.07	10.81	2.77
	10 min.	6.14	3.60	1.54	5.25	3.32	1.53	6.14	3.46	1.53
	15 min.	2.88	2.86	1.86	1.77	1.83	1.84	2.88	2.34	1.86
	30 min.	2.25	2.18	1.86	23.65	2.76	1.84	23.65	2.47	1.86

Acknowledgements

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair, Collaborative Research and Development, and Discovery Grant programs, by the Fonds de Recherche du Québec - Nature et Technologies (FRQ-NT) through its Team Research Project program, and by the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants

Chapter 4

Fleet-sizing for multi-depot and periodic vehicle routing problems using a modular heuristic algorithm

Abstract

In this paper, we address the problem of determining the optimal fleet size for three vehicle routing problems, i.e., multi-depot VRP, periodic VRP and multi-depot periodic VRP. In each of these problems, we consider three kinds of constraints that are often found in reality, i.e., vehicle capacity, route duration and budget constraints. To tackle the problems, we propose a new Modular Heuristic Algorithm (MHA) whose exploration and exploitation strategies enable the algorithm to produce promising results. Extensive computational experiments show that MHA performs impressively well, in terms of solution quality and computational time, for the three problem classes.

keywords: Multi-depot periodic vehicle routing problem, Fleet-sizing, Modular heuristic algorithm.

4.1 Introduction

In the classical Vehicle Routing Problem (VRP), a homogeneous fleet of vehicles services a set of customers from a single distribution depot or terminal. Each vehicle has a fixed capacity that cannot be exceeded and each customer has a known demand that must be fully satisfied. Each customer must be serviced by exactly one visit of a single vehicle and each vehicle must depart from the depot and return to the depot (Dantzig and Ramser (1959)).

During the past five decades, the vehicle routing problem and its variations have been extensively studied. However, surveying the literature, one can perceive that not all VRP variants have been addressed with the same degree of attention. This is the case for the problem classes considered in this paper. On the other hand, most of the methodological developments target a special problem variant, the Capacitated VRP (CVRP) or the VRP with Time Windows (VRPTW), despite the fact that the majority of the problems encountered in real-life applications display more complicating attributes and constraints. This also applies to the problem addressed in this paper. Moreover, the literature survey underlines that, the problem classes studied in this study have been often set up with objective functions other than the minimization of the fleet size, despite the fact that there are many real-life applications in which it is more crucial to give more importance to the minimization of the fleet size in comparison with other existing objectives as the total traveled distance. This situation may occur when important factors such as high vehicle fixed costs exist.

Our objective is to contribute toward addressing the above three challenges. In this paper, we propose a modular heuristic algorithm capable of successfully dealing with three VRP variants: Multi-depot VRP, MDVRP, Periodic VRP, PVRP, and Multi-depot Periodic VRP, MDPVRP. In each of these problems, the goal is to determine the optimal fleet size where three practical constraints, i.e., vehicle capacity, maximum route duration and budget constraints, should be satisfied. The proposed heuristic algorithm incorporates different exploration and exploitation strategies to produce good results, in terms of solution quality and computational efficiency.

The remainder of this paper is organized as follows: Section 4.2 gives the problem statement. In Section 4.3, the literature survey relevant to the topic of this study is presented. Different aspects of the proposed heuristic algorithm are described in Section 4.4. The experimental results are given in Section 4.5. Finally, Section 4.6 provides conclusions and the evaluation of the work.

4.2 Problem statement

In this section, we formally state each of the problem classes, introducing the notations used throughout this paper.

MDVRP- Consider an undirected graph $G(V, E)$. The node set V is the union of two subsets $V = V_C \cup V_D$, where $V_C = \{C_1, ..., C_N\}$ represents the customers and $V_D = \{D_1, ..., D_M\}$ includes the depots. With each node $i \in V_C$ is associated a deterministic demand q_i. The edge set E contains an edge for each pair of customers and for each depot-customer combination. There are no edges between depots. With each edge $(v_i, v_j) \in E$ is associated a travel cost c_{ij}. The travel distance for arriving to node j from node i (d_{ij}) is considered equal to c_{ij}. Each vehicle performs only one route and each vehicle route must start and finish at the same depot. (Cordeau et al. (1997)).

PVRP- In the PVRP, the undirected graph $G(V, E)$ is modified by fixing the value of M to one and by introducing a planning horizon of T periods. In such a graph, each customer i is characterized by a service frequency f_i, stating how often within these T periods the customer must be visited and a list Ω_i of possible visit-period combinations. Moreover, Ω is defined as the set of subsets of T, giving all the allowable patterns, that is: $\Omega = \cup_{i=1}^{N} \Omega_i$ (Vidal et al. (2012a)).

MDPVRP- Finally, the Multi-Depot Periodic VRP (MDPVRP) combines the two above problem settings, asking for the selection of a depot and a visit pattern for each customer, with services in different periods to the same customer being required to originate at the same depot (Vidal et al. (2012a)).

In each problem class, the goal is to determine the optimal fleet size subject to the following three practical constraints:

1. The vehicle capacity constraint: This constraint states that the total demand of the customers on any route should not exceed the vehicle capacity Q.
2. The route duration constraint: This constraint reveals this fact that the total duration of a route does not exceed a preset value D.
3. The budget constraint: In many logistical systems, one is usually faced with budgetary constraints that come from the fact that a limited investment budget is available for a certain area or a certain period of time. Although it is quite easy to understand the practical aspect of these constraints, budget considerations are almost always ignored when dealing with VRPs. In this paper, we consider a Travel-Distance Budget (TDB) constraint which imposes a threshold on the total distance traveled by vehicles for delivery operations. The TDB constraint is defined using two different rules, each realizing an important managerial challenge in real-life distribution and logistical systems. In the first rule (R_1), we set a bound on the total distance that vehicles are permitted to travel over the planning horizon. On the other hand, the second rule (R_2) aims to reflect the situations in which, due to geographical and operational constraints, the total distance traveled by vehicles assigned to a depot cannot exceed a limit in a period.

Depending on how we model the TDB constraint, each of the above problem classes can be expressed by one of the following mathematical programming models.

$$(R_1)\ min\ \ K \tag{4.1}$$

subject to

$$F(x, K) = b \tag{4.2}$$

$$\tau \leq \epsilon \tag{4.3}$$

In the above model, K is the total number of used vehicles (Fleet size) over the planning horizon which is to be minimized. Constraint (4.2) corresponds to the vehicle-capacity and route-duration restrictions described above. Constraint (4.3) imposes that the total traveled distance (τ) is limited by a positive value ϵ.

$$(R_2)\ min\ \ K \tag{4.4}$$

subject to

$$F(x, K) = b \tag{4.5}$$

$$\tau_{tj} \leq \epsilon_{tj} \qquad \forall t \in T, \forall j \in D; \tag{4.6}$$

where τ_{tj} is the total distance traveled by the vehicles assigned to depot j in period t and ϵ_{tj} is a positive upper bound which is set on τ_{tj}.

Cordeau et al. (1997) showed that the formulation of a generalized PVRP includes the MDVRP as a special case by associating a different period to each depot, such that the ith customer has a frequency f_i=1 and can be visited in any period. Vidal et al. (2012a) extended this result by proving that an MDPVRP with T periods and D depots can be transformed into a generalized PVRP by associating a period to each (period, depot) pair, such that the ith customer, having a list Ω_i of L patterns, is visited f_i times over the planning horizon using one of the $D \times L$ patterns. We rely on these two transformations in the development of the proposed modular heuristic algorithm.

4.3 Literature review

In this section, we focus on reviewing papers formerly published in the literature to solve the PVRP, the MDVRP and the MDPVRP. The aim of this review is first to present the most recently proposed heuristic and meta-heuristic algorithms for the considered problems and, to discern leading solution approaches which have been demonstrated to be impressive to address the three problem settings.

Several heuristics have been put forward for the MDVRP. Early heuristics, performing based on simple construction and improvement procedures, have been developed by Tillman (1969), Tillman and Hering (1971), Tillman and Cain (1972), Golden et al. (1977), and Raft (1982).

Cassidy and Bennett (1972) proposed an iterative heuristic for the multi-depot vehicle routing problem. The proposed method progressively improves the routing arrangements starting from an initial solution. An interesting feature of the algorithm is the method of data storage, which is designed to facilitate the alteration of route configurations. The suggested heuristic is divided in three main steps. In the first step, an initial solution is generated by assigning each customer to its nearest depot. In the second step, the initial solution obtained from the previous step is improved by taking each customer in turn and trying to fit it into another position. Finally in the last step, the algorithm examines all depots in the routes to see if any of them can be replaced by any of those still having enough capacity. Several years later, Chao et al. (1993) proposed a search procedure combining the record-to-record local search method for the reassignment of customers to different vehicle routes, followed by a 2-opt procedure for the improvement of individual routes. Salhi and Sari (1997) suggested a multi-level construction-based composite heuristic for solving a multi-depot fleet mix vehicle routing problem in which allocating customers to depots, finding the delivery routes and determining the vehicle fleet composition are simultaneously considered. The main purpose of that paper was to minimize the total traveled cost where both the vehicle capacity (the largest vehicle in case there are different types of vehicles) and the maximum distance traveled on any route must not be violated. The proposed heuristic consists of three levels. In the first level, a starting solution is found as follows: the vehicle fleet mix problem is first solved within each depot with certain customers left unassigned (borderline customers). Then each of these customers is inserted into an existing route or an empty route by using a selection-insertion procedure. In the second level, a composite heuristic, which attempts to improve on the solution found for each depot when taken separately, is introduced. Finally in the third level, a composite heuristic which considers all depots is implemented.

Various meta-heuristics have also been developed to tackle the MDVRP. Renaud et al. (1996c) described a tabu search heuristic in which an initial solution is built by first assigning every customer to its nearest depot. A petal algorithm is then used for the solution of the VRP associated with each depot. The algorithm is completed by an improvement phase using either a subset of the 4-opt exchanges to improve individual routes, swapping customers between routes from the same or different depots, or exchanging customers between three routes. The tabu search approach of Cordeau et al. (1997) is probably the best known algorithm for the MDVRP. An initial solution is obtained by assigning each customer to its nearest depot and a VRP solution is generated for each depot by means of a sweep algorithm. Improvements are performed by transferring a customer between two routes incident to the same depot, or by relocating a customer in a route incident to another depot. Reinsertions are performed by means of the GENI heuristic (Gendreau et al. (1992)). One of the main characteristics of this algorithm is that infeasible solutions are allowed throughout the search. Continuous diversification is achieved through the penalization of frequent moves. Dondo and Cerdà (2007) studied the multi-depot vehicle routing problem with time windows. To solve it, they presented a model-based large-scale neighbourhood search algorithm that steadily improves an initial solution generated through the three-phase cluster-based hybrid approach. At each iteration, a sequence of two evolutionary steps is executed. First, a neighbourhood around the starting solution is generated by using a mixed-integer linear problem that permits the algorithm to exchange multiple nodes between neighbouring trips. Next, a different neighbourhood is defined by just allowing relocations of nodes on the same tour. Lau et al. (2010) addressed an MDVRP in which the objective is to simultaneously optimize both the cost due to the total traveling distance and that due to the total traveling time. To solve the problem, a genetic algorithm with fuzzy logic adjusting the crossover rate and mutation rate after ten consecutive generations was proposed. Finally, Yu et al. (2011) designed a parallel ant colony optimization algorithm for the MDVRP. In the proposed algorithm, three improved strategies: the coarse-grain parallel strategy, the ant weight strategy and the local search strategy, were applied.

Solution algorithms proposed to solve the PVRP can be categorized into two main groups, i.e., classical heuristics, and meta-heuristics. Heuristics have been extensively studied to solve the PVRP. The majority of these heuristics are multi-phase optimization approaches which try to solve the problem at hand in a sequential manner. Russell and Gribbin (1991) presented a multi-phase approach to solve the PVRP. The first phase of the proposed method consists of a procedure which generates initial solutions by using a generalized network approximation method. The second phase involves an interchange heuristic that reduces the total traveled cost through a surrogate traveling salesman problem. In the third phase, the total traveled cost is further reduced by addressing the actual routes. Finally, a proposed 0-1 integer model is used to attempt further improvements. Chao et al. (1993) rovided a two-phase heuristic. To obtain an initial solution they solve an integer linear program to assign visit day combinations to the customers. In a second phase, they use several improvement operators while they relax the capacity of the vehicles. When getting stuck, re-initializations are performed. Bertazzi et al. (2004) suggested a heuristic algorithm for a special case of the PVRP namely the periodic traveling salesman problem, in which a single vehicle is used in each period. The algorithm is a construction type with an embedded improvement procedure. At each iteration, a procedure selects a not yet processed city, assigns to it a combination of visit days and, for each day of the combination day, inserts the city to the best position of the current partial tour. The iteration process is temporarily interrupted after a predefined number of iterations and an iterative improvement procedure tries to improve the current solution.

These early heuristics are outperformed by more recent meta-heuristic approaches, including tabu search, scatter search, and variable neighbourhood search. Cordeau et al. (1997) proposed a tabu search heuristic for the PVRP that can also be used to solve the Multi-Depot Vehicle Routing Problem and the Periodic Traveling Salesman Problem (PTSP). The neighbourhood consists of moving a customer from one route to another route of the same day or assigning a new visit combination to a customer. Insertions and removals of customers are performed using the GENI operator (Gendreau et al. (1992)). The tabu search algorithm allows for infeasible solutions during the search process using an adaptive penalty function. This paper presents an asynchronous parallel metaheuristic for the period vehicle routing problem (PVRP). Drummond et al. (2001) designed an island-based parallel meta-heuristic for the PVRP. The proposed algorithm was based on concepts used in parallel genetic algorithms and local search heuristics. Angelelli and Speranza (2002) presented a tabu search algorithm for an extension of the periodic vehicle routing problem where the homogeneous vehicles have the possibility of renewing their capacity at some intermediate facilities. The initial solution of the proposed tabu search is generated by using a procedure similar to the sweep algorithm (Gillett and Miller (1974)). Then, the initial solution is improved via an improvement procedure which consists of four move operators, i.e., relocation, changing the visiting schedule of a customer, redistribution, intersection. To enhance the performance of the proposed algorithm, the tabu search is permitted to search the solution space by using a tunneling strategy. Besides that, a diversification mechanism is also used. Recently, a scatter search procedure was developed by Alegre et al. (2007) for solving a problem of periodic pick-up of raw materials for a manufacturer of auto parts. They use a two-phase approach, that first assigns orders to days and then constructs the routes of each day. Alonso et al. (2008) proposed a tabu search for an extension of the periodic vehicle routing problem where each vehicle can service more than one route per day as long as the maximum delay operation time in not exceeded. Besides that, there exist some accessibility constraints of the vehicles to the customers in the sense that not every vehicle can visit every customer. The efficiency of the implemented tabu search is proved based on some existing and randomly generated test problems. Hemmelmayr et al. (2009) implemented a variable neighbourhood search for the periodic vehicle routing problem. First, for obtaining an initial solution each customer is randomly assigned a visit day combination. Routes are constructed by solving a vehicle routing problem for each day using Clarke and Wright savings algorithm (Clarke and Wright (1964)). Then, for the shaking phase two popular and effective neighbourhoods, i.e., move and cross-exchange, are proposed in order to enhance the quality of the starting solution in each iteration. Finally, the solution obtained through shaking is further improved by using 3-opt procedure as a local search. Pirkwieser and Raidl (2010) proposed a variable neighbourhood search for the periodic vehicle routing problem with time windows. In that paper, the authors claimed that using a random VNS often yielded significantly better results than a VNS using a reasonable fixed ordering of the shaking neighbourhoods. Furthermore, a selectively applied simple inter-route improvement procedure, 2-opt*, was shown to considerably improve both VNS variants at nearly no computational cost at all. Gulczynski et al. (2011) developed a new heuristic for the PVRP that combined integer programming and the record-to-record travel algorithm. The proposed heuristic produced very high-quality results on standard benchmark instances. The authors also extended the heuristic to two new variants of the PVRP that involve reassigning customers to new routes and balancing the workload among drivers across routes.

The majority of solution methods, targeting the MDPVRP, are divided into two main groups: 1) Classical heuristics which often solve the problem in a sequential manner, and 2) Sophisticated meta-heuristics and parallel algorithms which tackle the problem by simultaneously optimizing all the involved attributes.

We are aware of three heuristics of the first group. Hadjiconstantinou and Baldacci (1998) formulated the problem of providing maintenance services to a set of customers as the MDPVRP with Time Windows (MDPVRPTW). The authors proposed a multi-phase optimization problem and solved it using a four-phase algorithm. In the developed algorithm, all customers are first assigned to particular depot. Then, customer visits are successively inserted among available periods to obtain feasible visit combinations. At the third phase, each of the depot-period VRP sub-problems is separately solved using a tabu search algorithm. Finally, at the last phase, solutions obtained during the optimization process are improved by modifying the period or depot assignments through a 3-opt procedure. Kang et al. (2005) designed a two-phase heuristic method to solve the problem considered by Hadjiconstantinou and Baldacci (1998). In the proposed method, all feasible schedules are first generated from each depot for each period and, then, the set of routes are determined through using the shortest path problem. Parthanadee and Logendran (2006) proposed a tabu search heuristic to tackle the problem considered by Hadjiconstantinou and Baldacci (1998)). In this algorithm, all the initial assignments are built by cheapest insertion, where each customer is assigned to its nearest depot and is given its most preferred visit pattern. At the improvement phase, a neighbourhood search is defined by depot and visit pattern interchanges.

We are also aware of two contributions belonging to the second group. The first contribution was the evolutionary meta-heuristic proposed by Vidal et al. (2012a). The authors developed a hybrid Genetic Algorithm (GA) to tackle the MDPVRP and two of its special cases, i.e., the Multi-depot VRP (MDVRP) and the Periodic VRP (PVRP). The most interesting feature of the proposed GA is a new population-diversity management mechanism which allows a broader access to reproduction, while preserving the memory of what characterizes good solutions represented by the elite individuals of the population. The second contribution was the cooperative parallel algorithm designed by Crainic et al. (2009). The authors proposed a well structured cooperative parallel search method, denoted Integrative Co-operative Search (ICS), to solve highly complex combinatorial optimization problems. The proposed ICS framework involves problem decomposition by decision sets, integration of elite partial solutions yielded by the sub-problems, and adaptive guiding mechanism. The authors used the MDPVRPTW to present the applicability of the developed methodology.

This brief review supports the general statements made in Section 4.1 that the problem classes of this study, especially the MDPVRP, are among the VRP variants which did not receive an adequate degree of attention and the solution algorithms proposed to solve them are scarce. Moreover, to the best of our knowledge, there is no significant contribution in the literature dealing with the minimization of the fleet size for the problem settings considered in this paper. To contribute toward addressing these three challenges, we develop a Modular Heuristic Algorithm (MHA) to efficiently address the PVRP, the MDVRP and the MDPVRP. The proposed MHA is described in the next section.

4.4 The proposed Modular Heuristic Algorithm (MHA)

In this section, we propose a new modular heuristic algorithm which solves each of the considered problems in three sequential phases, each targeting one special dimension of the problem. The general concepts and structure of the heuristic algorithm are first described in Section 4.4.1. Then, the main components of the MHA are explained in details in Sections 4.4.2-4.4-5.

4.4.1 The general structure of MHA

The Modular Heuristic Algorithm (MHA) that we propose is based on the sequential optimization paradigm, but it includes a number of advanced exploration and exploitation features which contribute to its high performance level, in terms of solution quality and computational efficiency.

The general scheme of the proposed heuristic algorithm is displayed in Algorithm 3. MHA consists of three different steps that sequentially address the decisions to be made. These decisions are: the visit pattern assignment, the depot assignment and the detailed route design.

These three steps can be performed in different orders depending on the problem in question. Our experiments show that the order generating the best results, in terms of solution quality and computational time, is:

1. The visit pattern assignment: In this step, each of the customers is assigned to one of the possible visit patterns.
2. The depot assignment: In this step, customers of each period are assigned to depots.
3. The routing problem: Finally, in this step, the routes are established for each period and depot.

These three steps are iteratively repeated until MHA reaches its pre-defined stopping criteria. In this study, the following two stopping criteria are simultaneously considered:

- MHA is stopped if no improving solution is found for Ψ successive iterations. Ψ is a positive value which is determined at the beginning of the algorithm. Or,
- MHA is terminated if it passes a maximum allowable running time.

One of the most important characteristics of MHA is to use an elitism strategy which enables the algorithm not to lose good and diverse solutions obtained in the course of the optimization. Towards this end, MHA keeps all generated high-quality and diverse solutions in a list called the reference set. The reference set has the size equal to a predetermined positive value γ and consists of two different subsets. The first subset, B_1, preserves $\lceil 3 \times \gamma/4 \rceil$ high quality solutions, while the second subset, B_2, is made up of $\lfloor \gamma/4 \rfloor$ diverse solutions. The reference set is initially set as an empty list and is subject to be iteratively updated. Suppose a new solution, x_{new}, is obtained by the algorithm. The reference set is updated using the following two steps:

1. x_{new} is first investigated in terms of solution quality. In this case, x_{new} is directly added to B_1 if the number of solutions preserved in B_1 is less than $\lceil 3 \times \gamma/4 \rceil$; otherwise, if x_{new} is better than the worst existing solution in B_1, the latter is replaced by the former.

2. If none of the conditions mentioned in Step 1 are not met, x_{new} is assessed in terms of solution diversification. In this case, x_{new} is directly added to B_2 if the number of solutions existing in B_2 is less than $\lfloor \gamma/4 \rfloor$; otherwise, the following replacement strategy is implemented. We first define the contribution to diversity of solution *S* to the first subset of the reference set, $D(S, B_1)$, as the similarity between itself and its nearest neighbour in B_1, that is:

$$D(S, B_1) = \min_{X \in B_1, X \neq S} \Delta(S, X)$$

where $\Delta(S, X)$ is the Hamming distance. Moreover, let us define OF_S as the objective function value of solution *S*. The replacement strategy is implemented in three phases as follows: Firstly, the replacement strategy considers all the solutions of B_2 with poorer objective function values than S_{new} and finds the one, S_{max}, which maximizes the ratio of (*objective function value*)/(*contribution of diversity*) (Step 1). Then, the new generated solution, S_{new}, replaces S_{max} if the following inequality holds (Step 2):

$$\frac{OF_{S_{new}}}{D(S_{new}, B_1 - S_{max})} < \frac{OF_{S_{max}}}{D(S_{max}, B_1)} \tag{4.7}$$

In this way, we introduce into B_2 a solution with better objective function value and possibly higher contribution to diversity. If Inequality (4.7) does not hold, the worst solution of the set determined in the first step is replaced by S_{new} (Step 3).

Algorithm 3 Modular heuristic algorithm

```
-Initialize the search parameters.
-Determine the upper limit of the budget constraint.
-Set the initial reference set as an empty list.
while the termination criterion is not met do
    -Assign a possible visit pattern to each customer.
    -Assign each customer to a depot in each period of the
      selected visit pattern.
    -Design routes visiting customers assigned to the same
      depot using the three-phase heuristic.
    -Update the reference list.
end while
```

In the following sections, each of the steps used in MHA is described in details.

4.4.2 Solution representation

The first step in designing an algorithm for a particular problem is to devise a suitable solution representation scheme. In the proposed heuristic algorithm, the path representation proposed by Rahimi-Vahed et al. (2012a) is used. The idea of this path representation is that the customers are listed in the order in which they are visited. In this kind of representation, a single row array of the size equal to N+1 is generated for each depot in each period, where N is the number of customers to be visited. The first position of the array (index 0) is related to the corresponding depot, while each of the other positions (index i; $1 \leq i \leq N$) represents a customer. The value assigned to a position of the array represents which customer should be immediately visited after the customer or depot related to that position. In this path representation, negative values corresponds to the beginning of the next route index, 0 refers to the end of the routes and a vacant position reveals that the customer corresponding to that position is not served by the depot with which the array is associated. For a detailed description of the above solution representation, readers should refer to Rahimi-Vahed et al. (2012a).

4.4.3 Visit pattern assignment

The modular heuristic algorithm, as depicted in Algorithm 3, first assigns, at each iteration, a possible visit pattern to each customer. There exist a very scarce number of contributions in the literature that use systematic and non-random methods to assign customers to visit patterns. For example, Tan and Beasley (1984) extended the generalized assignment problem of Fisher and Jaikumar (1981) to assign a customer to an allowable visit pattern. Christofides and Beasley (1984) developed a median problem to establish an initial assignment of customers to visit patterns that meets customer service requirements. Russell and Gribbin (1991) designed a generalized network approximation method to assign visit patterns to customers. The proposed method was represented by a tripartite transshipment graph with source nodes (customers), transshipment nodes (allowable visit patterns), and sink nodes (periods of the planning horizon).

In this study, we also propose a non-blind and non-random algorithm to systematically assign customers to their possible visit patterns. In the proposed algorithm, we first locate a single point, called reference point, for each period of the planning horizon and, then, using a new Integer Programming Model (IPM), customers are assigned to visit patterns so as to minimize the sum of customer-to-reference point distances. The proposed IPM, unlike the similar models existing in the literature, is governed by some parameters whose values are dynamically adjusted in the course of the optimization. This feature enables IPM to assign better visit patterns to customers as MHA evolves. The integer programming model is formulated as follows:

$$min \sum_{i=1}^{N} \sum_{k \in \Omega_i} [\sum_{t=1}^{T} a_{kt}\{(x_i - x_t^{\nu})^2 + (y_i - y_t^{\nu})^2\}]u_{ik} \tag{4.8}$$

subject to

$$\sum_{k \in \Omega_i} u_{ik} = 1 \qquad \forall i \in V; \tag{4.9}$$

$$LB_t^{\nu} \leq \sum_{i=1}^{N} \sum_{k \in \Omega_i} a_{kt} q_i u_{ik} \leq UB_t^{\nu} \qquad \forall t \in T; \tag{4.10}$$

$$u_{ik} \in \{0, 1\} \qquad \forall i \in V, \forall k \in \Omega_i; \tag{4.11}$$

In the above model, (x_i, y_i) is the location of customer i and (x_t^{ν}, y_t^{ν}) is the location of a reference point which is generated, in iteration ν of the algorithm, for period t. Moreover, the parameter a_{kt} equals 1 if period t is in pattern k, and 0 otherwise. The decision variable of this model is u_{ik}, which is equal to 1 if customer i is assigned to pattern k, and 0 otherwise. Finally, LB_t^{ν} and UB_t^{ν} are respectively lower and upper limits that bound the total number of demands which should be satisfied in period t and iteration ν. As it can be seen in this mathematical formulation, the aim is to assign possible visit patterns to customers so that the total squared Euclidean distance between the customers and reference points is minimized. Constraints (4.9) impose that each customer is assigned to exactly one feasible pattern. Constraints (4.10) show that the demands serviced in a given period must be within an imposed interval.

One of the most crucial parameters affecting the strength of the above integer programming model is the reference points' locations. In this paper, the reference points' locations are generated using a memory-based algorithm as follows: We first enumerate all the customers that can be serviced in period t (t=1,2...T). That is: I_t=$\{i \in V_c \mid \exists k \in \Omega_i$: $a_{kt} = 1\}$. Then, the coordinates of the reference point is calculated using the two following formulas:

$$x_t^\nu = \frac{\sum_{i \in I_t} w_{it} x_i}{\sum_{i \in I_t} w_{it}} \qquad \forall \nu = 1, 2..., \forall t \in T; \tag{4.12}$$

$$y_t^\nu = \frac{\sum_{i \in I_t} w_{it} y_i}{\sum_{i \in I_t} w_{it}} \qquad \forall \nu = 1, 2..., \forall t \in T; \tag{4.13}$$

In the above equations, w_{it} is defined as a self-adjusting positive weight which reflects the desirability of visiting customer i in period t. The values of these weights are dynamically adjusted, at each iteration, based on the information gathered from the reference set. The adjusting procedure is summarized as follows: If customer i is visited in period t in more than $\theta\%$ of the solutions existing in the reference set, the value of w_{it} is multiplied by 1+φ, otherwise it remains unchanged, where φ is a positive parameter. It should be noted that, this adjusting procedure starts after the modular heuristic passes a preliminary phase called Warming-up Stage (WS). In WS, the algorithm is repeated in λ successive iterations, $\lambda \geq \gamma$, so that, at each iteration, the weights involved in equations (4.12) and (4.13) are randomly generated in the interval (0,1].

The other important parameters having key roles in the integer programming model are the bounds considered in constraints (4.10). In this paper, LB_t^ν and UB_t^ν are considered as two parameters which are iteratively adjusted based on the information obtained from elite solutions kept in the reference set. Towards this end, LB_t^ν and UB_t^ν are respectively defined, in iteration ν, as the minimum and maximum required capacity for period t observed in elite solutions existing in the reference set. It should be noted that, in the Warming-up Stage described above, Constraints (4.10) are relaxed by respectively setting LB_t^ν and UB_t^ν to 0 and ∞.

4.4.4 Depot assignment

The proposed heuristic continues by assigning the customers of each period to depots. The assignment algorithm that we propose in this step belongs to a category of assignment problems which is called assignment by clusters. In this type of assignment problems, a cluster is defined as the set of points consisting of a depot and the customers assigned to it. The algorithms in this class try to build compact clusters of customers for each depot. When a customer is assigned to a cluster it means that this customer is assigned to that cluster's depot. In this study, the way in which customers are incorporated in a cluster is defined by an integer programming model which is mathematically expressed as follows:

$$min \sum_{t=1}^{T} \sum_{i \in \Pi_t} \sum_{j=1}^{M} b_{ijt} \{(x_i - x_{D_j})^2 + (y_i - y_{D_j})^2\} \omega_{ijt} \tag{4.14}$$

subject to

$$\sum_{j=1}^{M} \omega_{ijt} = 1 \qquad \forall t \in T, \forall i \in \Pi_t; \tag{4.15}$$

$$\omega_{ijt} \in \{0,1\} \qquad \forall t \in T, \forall i \in \Pi_t, \forall j \in D; \tag{4.16}$$

In the above model, (x_{D_j}, y_{D_j}) is the location of depot j and Π_t is the set of customers to be visited in period t. The decision variable of this model is ω_{ijt}, which is equal to 1 if customer i is assigned to depot j in period t, and 0 otherwise. Moreover, the parameter b_{ijt} is defined as the penalty of assigning customer i to depot j in period t. In fact, the higher b_{ijt} is, the more desirable customer i is not assign to depot j in period t. The objective of the above integer programming model is to assign customers to depots so that the total weighted squared Euclidean distance between the customers and depots is minimized. Moreover, constraints (4.15) force that each customer to be assigned to exactly one depot in a period.

The strength of the integer programming model is dependent on the penalties involved in the objective function. In this paper, these penalties are considered as self-adjusting parameters which are iteratively updated based on the information obtained from the reference set. The adjusting procedure is summarized as follows: If customer i is assigned to depot j in period t in more than $\theta\%$ of the solutions kept in the reference set, the value of b_{ijt} is divided by 1+μ, otherwise it is multiplied by 1+μ, where μ is a positive parameter. This updating procedure enables the algorithm to assign customers to better depots as the heuristic gets closer to the termination criteria. It should again be noted that, in the Warming-up Stage, the penalties involved in the objective function are randomly generated in interval (0,1].

4.4.5 Route design

In this phase, customers assigned to the same depot, in each period, are divided into different routes. Towards this end, a heuristic algorithm, consisting of the following three phases, is implemented in a sequential manner:

1. **Construction phase**- In the first phase, customers of each depot are assigned to a giant tour using the GENIUS heuristic to solve the corresponding treveling salesman problem. GENIUS, proposed by Gendreau et al. (1992), consists of two phases that are implemented in a sequential manner. In the first phase, called GENI, a Hamiltonian cycle is progressively generated by inserting vertices (i.e., customers) one at a time. More precisely, GENI starts with a partial solution consisting of three arbitrarily chosen vertices and, at each iteration, it includes any given vertex between two of its p closest neighbors on the partial cycle. While making an insertion, GENI performs a local reoptimization of the partial cycle. Once all vertices have been inserted, the second phase, named US, is executed as a post-optimization heuristic which successively removes each vertex from the cycle, and reinserts it using GENI.

2. **Splitting phase**- In the second phase, using the optimal splitting procedure proposed by Prins (2004), each constructed giant tour is split into shorter routes, each satisfying the vehicle capacity constraint. In other words, by relaxing the route duration restriction and budget constraint, each giant tour is split to least possible number of shorter routes.

3. **Improving phase**- Finally, in the third phase, a heuristic consisting of two exchange procedures is implemented on each constructed route in order to reduce the route's length and, accordingly, to improve the total traveled distance. One of the main characteristics of the proposed heuristic method is that infeasible solutions are allowed throughout the search. Let us assume that X denotes the new solution generated by the search mecha-

nism. Moreover, let $\tau(X)$ denote the total traveled distance of solution X, and let $A(X)$ and $B(X)$ denote the total violation of the route-length and ϵ- constraints, respectively. Solution X is evaluated by a function, $z(X) = \tau(X) + \alpha A(X) + \beta B(X)$, where α and β are self-adjusting positive parameters. By dynamically adjusting the values of these two parameters, this relaxation mechanism facilitates the exploration of the search space and is particularly useful for tightly constrained instances. Parameter α is adjusted as follows: if there is no violation of the route-length constraints, the value of α is divided by $1+\Lambda$, otherwise it is multiplied by $1+\Lambda$, where Λ is a positive parameter. A similar rule applies also to β with respect to route duration constraint. The two proposed exchange procedures are described as follows:

- **SWAP procedure**: The first exchange procedure, called SWAP, is performed by choosing two distinct customers i and j, where i= 1,2...n_k-1 and j=i+1, i+2...n_k (n_k is the number of customers that are visited in the kth route), and then exchanging their positions within the route. If the swapping procedure results in a better solution according to the penalty function described above, the positions are exchanged, otherwise the solution remains unchanged. The procedure stops when no more exchanges that result in an improving solution are possible.
- **INSERT procedure**: The second exchange procedure, called INSERT, is based on removing a customer from one route and inserting it into another route. Towards this end, we first randomly select a customer from the longest route of a depot whose length is defined as t_{max}. Then, the selected customer is removed from its current position and is re-inserted to a new route using the two following methods:
  (a) **Inter-depot method**: In this method, the removed customer is re-inserted into either one of the existing routes or a new route of the same depot from which the customer has been removed.
  (b) **Intra-depot method**: In this method, the depot to which the customer is concurrently assigned is changed to another one and the customer is re-inserted into either one of the existing routes or a new route of the new depot.

  Note that, in both cases, an insertion is called feasible if: 1) It does not violate the vehicle's capacity constraint, and 2) It does not produce a route longer than

t_{max}. The position to which the customer is inserted is the one that satisfies the two above conditions and results in the best improvement in the penalty function. This procedure is repeated for a predetermined number of iterations, denoted by σ.

4.5 Experimental results

In this section, the performance of the proposed MHA is investigated based on three different sets of test problems. The first two sets, each including 10 problem instances, have been developed by Cordeau et al. (1997) for the PVRP and the MDVRP, respectively, while the last set includes 10 different test problems which have been designed by Vidal et al. (2012a) for the MDPVRP. Note that, in all the considered problem instances, the number of vehicles assigned to a depot, which was originally set as a limited and fixed parameter, is ignored. Detailed information on these sets are provided in Subsection 4.5.2.

The efficiency of the developed heuristic is tested in two different settings that are defined according to how the budget constraint is formulated: either using Rule R_1 or Rule R_2 defined in Section 4.2. Recall that, in Rule R_1, an upper limit is imposed on the total distance traveled over the planning horizon. As for Rule R_2, an upper bound is enforced on the total distance traveled by the vehicles assigned to each depot in each period.

In both of the above cases, values assigned as the upper bound of the budget constraint may have a major impact on the performance of the proposed modular heuristic algorithm. In this paper, the upper bounds of both rules (R_1 and R_2) are initially set based on the information extracted from the Best Known Solution (BKS) reported by Vidal et al. (2012a) for the considered problems. More precisely, the values of ϵ, in Rule R_1, is set to the total traveled distance of the BKS, whereas the value of ϵ_{td}, in Rule R_2, is fixed to the total distance traveled by the vehicles assigned to depot d in period t of the BKS. Then, we systematically vary the values of the upper bound, set on the budget constraint, to investigate how the performance of MHA is affected by tightening or widening the budget constraint.

The proposed algorithm is ran on each problem instance, for both budget constraint rules, and its efficiency, in terms of solution quality and computational time, is compared to the

Unified Tabu Search (UTS) implemented in Lahrichi et al. (2011). Note, however, that this algorithm was modified to handle the objective considered in this paper, i.e., minimization of the fleet size. Furthermore, the penalty function used in the algorithm was also modified to include budget constraint violations. Both algorithms have been coded in C++ and executed on a Pentium 4, 2.8 GHz, and Windows XP using 256 MB of RAM.

Different aspects of the experimental results are discussed as follows: In Section 4.5.1, we first use a well-structured algorithm to calibrate all the parameters involved in the heuristic algorithm. Then, in Section 4.5.2, computational results are given in details.

4.5.1 Parameter setting

Like most heuristic and meta-heuristic algorithms, the proposed heuristic method has several parameters that need to be tuned before it can reach good results. The problem then turns into finding best parameter setting for the heuristic to solve the considered problems efficiently and timely. Table 4.1 provides a summary of all the parameters involved in the algorithm.

Table 4.1: Parameters of the heuristic algorithm

Symbol	Description
γ	Maximum size of the reference set
θ	Threshold defined in Sections 4.3 and 4.4
φ	Factor involved in updating w_{it}
λ	Number of times that WS is repeated
μ	Factor involved in updating b_{ijt}
α, β	Self-adjusting parameters in the penalty function
Λ	Factor involved in updating α and β
σ	Number of times that INSERT is repeated
Ψ	Maximum allowable number of non-improving iterations

There are various different methods in the literature to calibrate parameters used in a heuristic or meta-heuristic algorithm. Coy et al. (2000) designed a procedure based on statistical Design Of Experiments (DOE) that systematically selects high-quality parameter values. The parameter setting procedure has four steps that are implemented in a sequential manner. In the first step, a subset of problems to analyze is chosen from the entire set of problems. In the second step, computational experience is used to select the starting level of each parameter, the range over which each parameter will be varied, and the amount by which each parameter should be changed. In the third step, good parameter settings are selected for each problem in the analysis set using fractional factorial design and response surface optimization. Finally, in the fourth step, the parameter values obtained in the third step are averaged to obtain high-quality parameter values. The proposed approach does not use higher-order models (such as quadratic) since different response surfaces are averaged over all considered instances. The authors acknowledged that their method will perform poorly if the representative test problems are not chosen correctly or if the problem class is so broad that it requires very different parameter settings. For a detailed description, see Coy et al. (2000).

In this paper, we adopt the above four-step calibration method to tune the parameters used in the heuristic algorithm. The calibration results for each class, along with the final choice of parameter values, are reported in Table 4.2.

Table 4.2: Calibration results

Symbol	PVRP	MDVRP	MDPVRP	Final choice
γ	30	40	50	40
θ	0.2	0.3	0.4	0.3
φ	1	1	1	1
λ	$\lceil \frac{N*M*T}{5} \rceil$	$\lceil \frac{N*M*T}{5} \rceil$	$\lceil \frac{N*M*T}{5} \rceil$	$\lceil \frac{N*M*T}{5} \rceil$
μ	1	1	1	1
α, β	1, 1	1, 1	1, 1	1, 1
Λ	1	1	1	1
σ	N	N	N	N
Ψ	$N*M*T$	$N*M*T$	$N*M*T$	$N*M*T$

4.5.2 Computational results

We tested the proposed modular heuristic algorithm on the problem instances described at the beginning of this section using the two rules described in Section 4.2, Rules R_1 and R_2. Detailed computational results on each fold are given in the following subsections.

Rule R_1

In Rule R_1, both the modular heuristic and UTS are run 10 times on each problem instance. Moreover, the maximum running time of both algorithms for the PVRP and MDVRP instances is set to 20 minutes, while for the MDPVRP instances, due to their greater difficulty, the maximum running time is fixed to 30 minutes.

Results on PVRP instances are presented in Table 4.3. The first four columns of this table respectively display instance identifier, number of customers, number of depots and number of periods. Moreover, in Column 5, different values of the upper bound, set on the budget constraint, are shown. In this column, ϵ^* refers to the total traveled distance of the BKS. The results of the proposed heuristic method are shown in Columns 6 and 7 as the average fleet size and computational time on 10 independent runs. We compare the performance of our heuristic algorithm to the results obtained with the modified version of the UTS implementation of Lahrichi et al. (2011) (UTS in Columns 8 and 9). Finally, the average percentage gap of the modular heuristic with respect to UTS, on each problem instance and for each value considered for ϵ, is reported in Column 10 (a negative value means a better performance of the modular heuristic).

As shown in Table 4.3, the proposed modular heuristic algorithm always produces either equivalent or better results compared to UTS. However, the average percentage gap between two algorithms clearly varies depending on values set as the upper bound of the budget constraint. In the case where the ϵ value is set to ϵ^*, the average percentage gap is -3.7% indicating that the modular heuristic performs significantly better than UTS. On the other hand, in the cases where the budget constraint is tightened by fixing the ϵ value to 0.8ϵ^* and 0.9ϵ^*, both algorithms face a more challenging task to produce good results and, consequently, their performance slightly worsens. However, the results show that the tighten the budget constraint is, the more the modular heuristic shows its superiority to produce better results. These results reveal this fact that the proposed heuristic algorithm has better capability, compared to UTS, to solve PVRP instances, especially for those problems having more restricted search space. The average percentage gaps between the two algorithms respectively increase to -6.3% and -5.6%, for 0.8ϵ^* and 0.9ϵ^* cases. Contrary, increasing the upper bounds to 1.1ϵ^* and 1.2ϵ^*

Table 4.3: Results on PVRP instances with Rule R_1

Instance	N	M	T	ϵ	Heuristic (Ave.)		UTS (Ave.)		Gap (%)
					K	Time (min)	K	Time (min)	
pr01	48	4	4	$0.8\epsilon^*$	3	0.19	3	0.37	0
				$0.9\epsilon^*$	3	0.18	3	0.35	0
				ϵ^*	2	0.18	2	0.35	0
				$1.1\epsilon^*$	2	0.16	2	0.35	0
				$1.2\epsilon^*$	2	0.16	2	0.33	0
pr02	96	4	4	$0.8\epsilon^*$	5	0.52	5	0.94	0
				$0.9\epsilon^*$	5	0.48	5	0.85	0
				ϵ^*	4	0.46	4	0.80	0
				$1.1\epsilon^*$	4	0.42	4	0.75	0
				$1.2\epsilon^*$	4	0.39	4	0.71	0
pr03	144	4	4	$0.8\epsilon^*$	7	4.73	7	6.69	0
				$0.9\epsilon^*$	7	4.27	7	6.13	0
				ϵ^*	6	3.81	6	5.56	0
				$1.1\epsilon^*$	6	3.27	6	5.38	0
				$1.2\epsilon^*$	5	3.12	6	5.09	-17
pr04	192	4	4	$0.8\epsilon^*$	9	8.33	10	9.57	-10
				$0.9\epsilon^*$	9	8.13	9	9.12	0
				ϵ^*	7	7.42	8	8.19	-13
				$1.1\epsilon^*$	7	7.13	8	8.10	-13
				$1.2\epsilon^*$	7	6.75	8	7.92	-13
pr05	240	4	4	$0.8\epsilon^*$	12	15.80	14	19.91	-14
				$0.9\epsilon^*$	12	15.51	14	19.72	-14
				ϵ^*	10	14.73	10	18.22	0
				$1.1\epsilon^*$	10	14.23	10	17.71	0
				$1.2\epsilon^*$	9	13.79	10	17.24	-10
pr06	288	4	4	$0.8\epsilon^*$	14	16.45	16	19.36	-13
				$0.9\epsilon^*$	13	16.02	16	19.29	-19
				ϵ^*	12	15.51	14	19.11	-14
				$1.1\epsilon^*$	12	15.32	13	18.72	-8
				$1.2\epsilon^*$	12	15.27	13	18.33	-8
pr07	72	6	6	$0.8\epsilon^*$	4	1.29	4	1.89	0
				$0.9\epsilon^*$	4	1.21	4	1.84	0
				ϵ^*	3	1.15	3	1.72	0
				$1.1\epsilon^*$	3	1.11	3	1.65	0
				$1.2\epsilon^*$	3	1.06	3	1.57	0
pr08	144	6	6	$0.8\epsilon^*$	7	5.64	8	7.50	-13
				$0.9\epsilon^*$	7	5.12	7	7.39	0
				ϵ^*	6	4.76	6	7.23	0
				$1.1\epsilon^*$	6	4.55	6	7.14	0
				$1.2\epsilon^*$	6	4.31	6	7.11	0
pr09	216	6	6	$0.8\epsilon^*$	11	17.12	11	18.99	0
				$0.9\epsilon^*$	10	16.70	11	18.96	-10
				ϵ^*	10	16.22	11	18.89	-10
				$1.1\epsilon^*$	10	16.17	11	18.69	-10
				$1.2\epsilon^*$	10	16.09	11	18.44	-10
pr10	288	6	6	$0.8\epsilon^*$	14	19.02	16	19.54	-13
				$0.9\epsilon^*$	14	18.95	16	19.23	-13
				ϵ^*	12	18.88	12	19.01	0
				$1.1\epsilon^*$	12	18.75	12	18.86	0
				$1.2\epsilon^*$	12	18.43	12	18.77	0

results in producing better solutions by both algorithms. In these cases, the average percentage gaps respectively change to -3.1% and -5.8%. Figure 4.1 schematically shows how the average percentage gap varies when changing the ϵ value.

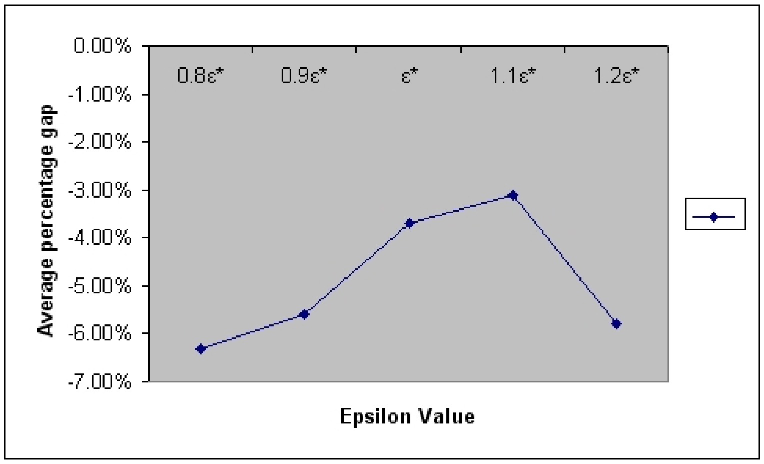


Figure 4.1: Average percentage gap in the PVRP instances

Results on MDVRP instances are displayed in Table 4.4, where, as shown in Table 4.3, Columns 2-4 represents respectively instance identifier, number of customers, number of depots and number of periods. The results obtained by the heuristic algorithm are compared, once again, to the modified version of the unified tabu search of Lahrichi et al. (2011), in terms of solution quality and computational time.

The main conclusions derived from Table 4.4 are similar to those stated above for the PVRP. The modular heuristic clearly outperforms the unified tabu search on the majority of problem instances. Figure 4.2 represents how the average percentage gap of two algorithms changes when varying the ϵ value.

Results on MDPVRP instances are finally summarized in Table 4.5 whose structure is similar to that of Tables 4.3 and 4.4. Once again, the results produced by the heuristic algorithm are compared to UTS in order to assess the efficiency of the algorithm, in terms of solution quality and computational time.

Table 4.5 show that the proposed modular heuristic algorithm is considered as a competitive solution methodology, able to produce high quality solutions in a reasonable time. As for the two previous problems, the relative performance of two algorithms, measured by the average

Table 4.4: Results on MDVRP instances with Rule R_1

					Heuristic (Ave.)		UTS (Ave.)		
Instance	N	M	T	ϵ	K	Time (min)	K	Time (min)	Gap (%)
pr01	48	4	4	$0.8\epsilon^*$	5	0.13	5	0.15	0
				$0.9\epsilon^*$	5	0.11	5	0.15	0
				ϵ^*	4	0.09	4	0.14	0
				$1.1\epsilon^*$	4	0.08	4	0.14	0
				$1.2\epsilon^*$	4	0.16	4	0.13	0
pr02	96	4	4	$0.8\epsilon^*$	8	0.41	9	0.81	-11
				$0.9\epsilon^*$	8	0.35	9	0.74	-11
				ϵ^*	7	0.31	8	0.69	-13
				$1.1\epsilon^*$	7	0.30	8	0.62	-13
				$1.2\epsilon^*$	7	0.29	8	0.60	-13
pr03	144	4	4	$0.8\epsilon^*$	13	1.06	14	1.52	-7
				$0.9\epsilon^*$	13	1.02	13	1.44	0
				ϵ^*	12	0.55	12	1.39	0
				$1.1\epsilon^*$	12	0.51	12	1.30	0
				$1.2\epsilon^*$	12	0.48	12	1.26	0
pr04	192	4	4	$0.8\epsilon^*$	16	3.88	17	4.68	-6
				$0.9\epsilon^*$	16	3.82	17	4.60	-6
				ϵ^*	15	3.77	16	4.51	-6
				$1.1\epsilon^*$	15	3.70	16	4.46	-6
				$1.2\epsilon^*$	15	3.64	16	4.39	-6
pr05	240	4	4	$0.8\epsilon^*$	21	7.50	22	9.51	-5
				$0.9\epsilon^*$	21	7.44	22	9.32	-5
				ϵ^*	20	7.29	20	9.16	0
				$1.1\epsilon^*$	20	7.22	20	9.09	0
				$1.2\epsilon^*$	20	7.16	20	9.03	0
pr06	288	4	4	$0.8\epsilon^*$	25	8.94	27	9.62	-7
				$0.9\epsilon^*$	25	8.90	27	9.56	-7
				ϵ^*	24	8.77	24	9.44	0
				$1.1\epsilon^*$	24	8.71	24	9.37	0
				$1.2\epsilon^*$	24	8.63	24	9.28	0
pr07	72	6	6	$0.8\epsilon^*$	7	0.30	7	0.55	0
				$0.9\epsilon^*$	7	0.28	7	0.54	0
				ϵ^*	6	0.26	6	0.51	0
				$1.1\epsilon^*$	6	0.24	6	0.47	0
				$1.2\epsilon^*$	6	0.22	6	0.41	0
pr08	144	6	6	$0.8\epsilon^*$	13	1.68	13	1.97	0
				$0.9\epsilon^*$	13	1.66	13	1.93	0
				ϵ^*	12	1.57	12	1.89	0
				$1.1\epsilon^*$	12	1.51	12	1.83	0
				$1.2\epsilon^*$	12	1.47	12	1.79	0
pr09	216	6	6	$0.8\epsilon^*$	20	7.77	22	8.84	-9
				$0.9\epsilon^*$	20	7.73	22	8.78	-9
				ϵ^*	19	7.68	20	8.71	-5
				$1.1\epsilon^*$	19	7.60	20	8.64	-5
				$1.2\epsilon^*$	19	7.55	20	8.59	-5
pr10	288	6	6	$0.8\epsilon^*$	25	8.80	26	9.73	-4
				$0.9\epsilon^*$	25	8.72	26	9.64	-4
				ϵ^*	24	8.65	24	9.57	0
				$1.1\epsilon^*$	24	8.60	24	9.51	0
				$1.2\epsilon^*$	24	8.53	24	9.46	0

Table 4.5: Results on MDPVRP instances with Rule R_1

Instance	N	M	T	ϵ	Heuristic (Ave.) K	Heuristic (Ave.) Time (min)	UTS (Ave.) K	UTS (Ave.) Time (min)	Gap (%)
pr01	48	4	4	$0.8\epsilon^*$	5	0.32	5	0.43	0
				$0.9\epsilon^*$	5	0.29	5	0.42	0
				ϵ^*	4	0.25	4	0.40	0
				$1.1\epsilon^*$	4	0.22	4	0.38	0
				$1.2\epsilon^*$	4	0.20	4	0.36	0
pr02	96	4	4	$0.8\epsilon^*$	5	0.86	5	1.39	0
				$0.9\epsilon^*$	5	0.81	5	1.34	0
				ϵ^*	4	0.73	4	1.28	0
				$1.1\epsilon^*$	4	0.69	4	1.26	0
				$1.2\epsilon^*$	4	0.64	4	1.19	0
pr03	144	4	4	$0.8\epsilon^*$	9	4.88	9	6.41	0
				$0.9\epsilon^*$	9	4.79	9	6.33	0
				ϵ^*	8	4.72	8	6.23	0
				$1.1\epsilon^*$	8	4.55	8	6.13	0
				$1.2\epsilon^*$	7	4.48	8	6.09	-13
pr04	192	4	4	$0.8\epsilon^*$	9	13.73	10	15.66	-10
				$0.9\epsilon^*$	8	13.62	9	15.58	-10
				ϵ^*	7	13.56	8	15.52	-13
				$1.1\epsilon^*$	7	13.50	8	15.44	-13
				$1.2\epsilon^*$	7	13.41	8	15.37	-13
pr05	240	4	4	$0.8\epsilon^*$	13	20.58	14	24.33	-7
				$0.9\epsilon^*$	13	20.53	14	24.17	-7
				ϵ^*	12	20.44	12	24.09	0
				$1.1\epsilon^*$	12	20.37	12	23.88	0
				$1.2\epsilon^*$	12	20.29	12	23.82	0
pr06	288	4	4	$0.8\epsilon^*$	14	17.41	16	21.45	-13
				$0.9\epsilon^*$	13	17.32	15	21.34	-13
				ϵ^*	12	17.22	14	21.25	-14
				$1.1\epsilon^*$	12	17.16	13	21.20	-8
				$1.2\epsilon^*$	12	17.08	13	21.11	-8
pr07	72	6	6	$0.8\epsilon^*$	7	1.46	7	2.06	0
				$0.9\epsilon^*$	7	1.39	7	1.98	0
				ϵ^*	6	1.35	6	1.96	0
				$1.1\epsilon^*$	6	1.30	6	1.91	0
				$1.2\epsilon^*$	6	1.24	6	1.84	0
pr08	144	6	6	$0.8\epsilon^*$	7	5.25	8	8.04	-13
				$0.9\epsilon^*$	7	5.14	7	7.97	0
				ϵ^*	6	5.06	6	7.91	0
				$1.1\epsilon^*$	6	5.03	6	7.88	0
				$1.2\epsilon^*$	6	4.98	6	7.82	0
pr09	216	6	6	$0.8\epsilon^*$	13	20.42	13	24.19	0
				$0.9\epsilon^*$	13	20.31	14	24.12	-7
				ϵ^*	12	20.19	13	23.96	-8
				$1.1\epsilon^*$	12	20.10	13	23.89	-8
				$1.2\epsilon^*$	12	19.89	13	23.81	-8
pr10	288	6	6	$0.8\epsilon^*$	19	23.14	20	26.79	-5
				$0.9\epsilon^*$	19	23.04	20	26.71	-5
				ϵ^*	18	22.88	18	26.53	0
				$1.1\epsilon^*$	18	22.76	18	26.49	0
				$1.2\epsilon^*$	18	22.70	18	26.41	0

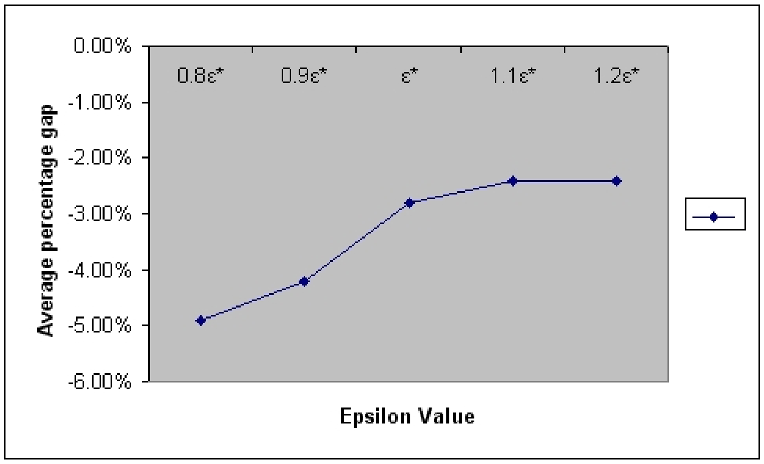


Figure 4.2: Average percentage gap in the MDVRP instances

percentage gap, depends upon the on values assigned to ϵ. Figure 4.3 depicts how different ϵ values affect the average percentage gap.

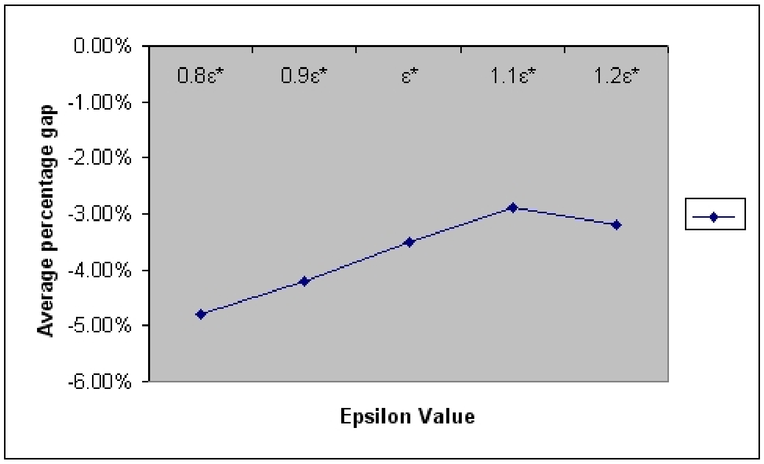


Figure 4.3: Average percentage gap in the MDPVRP instances

Rule R_2

In Rule R_2, we investigate how the algorithm reacts when an upper bound is imposed on the total distance that all vehicles assigned to each depot are permitted to travel in each period. The specifications of how the algorithms are applied in the case of Rule R_2 is exactly the same as in the case of Rule R_1 (i.e., the same number of runs is applied as well as identical maximum allotted run-times for the algorithms are considered for each problem).

Tables 4.6-4.8 display the results obtained by the heuristic algorithm on the PVRP, MDVRP and MDPVRP problems, respectively. Each of these tables use the same structure considered for the tables of the previous subsection. It should be noted that, in these tables, ϵ^* is a $T \times D$ matrix, in which ϵ_{td} corresponds to the total distance traveled by vehicles assigned to depot d in period t of the BKS. The performance of the heuristic algorithm is compared, on each set of test problems, to the modified version of the unified tabu search implementation of Lahrichi et al. (2011).

As shown in Tables 4.6-4.8, the proposed modular heuristic algorithm performs impressively well relative to the UTS, in terms of solution quality and computational time. For each of the problems considered, we investigated how the existing average percentage gap of two algorithms is affected by different ϵ values. The results obtained are shown by Figure 4.4.

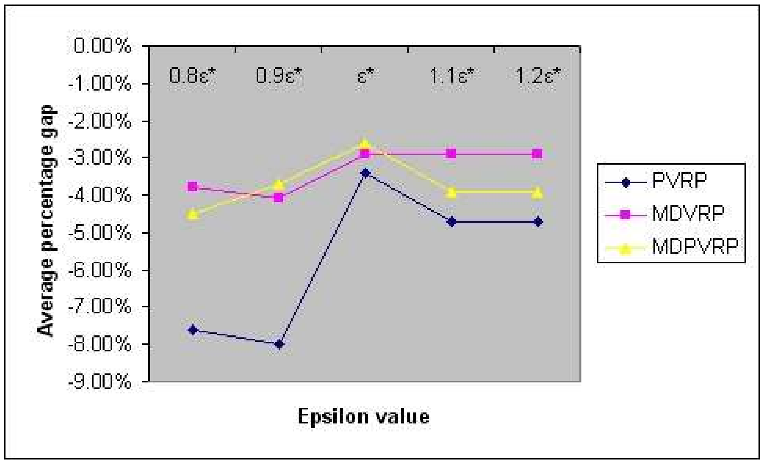


Figure 4.4: Average percentage gap

Moreover, the results displayed by Tables 4.6-4.8 reveal this fact that restricting the search space through bounding the total distance allowed to be traveled by vehicles assigned to a depot in each period may result in decreasing the quality of the heuristic algorithm.

The results of Tables show that restricting the search space through bounding the total distance allowed to be traveled by vehicles assigned to a depot in each period may result in a decrease of the quality of the heuristic algorithm.

Table 4.6: Results on PVRP instances with Rule R_2

					Heuristic (Ave.)		UTS (Ave.)		
Instance	N	M	T	ϵ	K	Time (min)	K	Time (min)	Gap (%)
pr01	48	4	4	$0.8\epsilon^*$	3	0.20	3	0.39	0
				$0.9\epsilon^*$	3	0.19	3	0.36	0
				ϵ^*	2	0.19	2	0.35	0
				$1.1\epsilon^*$	2	0.17	2	0.34	0
				$1.2\epsilon^*$	2	0.17	2	0.33	0
pr02	96	4	4	$0.8\epsilon^*$	5	0.54	5	0.94	0
				$0.9\epsilon^*$	5	0.49	5	0.85	0
				ϵ^*	4	0.47	4	0.82	0
				$1.1\epsilon^*$	4	0.44	4	0.77	0
				$1.2\epsilon^*$	4	0.40	4	0.73	0
pr03	144	4	4	$0.8\epsilon^*$	7	4.12	8	6.38	-13
				$0.9\epsilon^*$	7	3.92	8	6.16	-13
				ϵ^*	6	3.83	6	5.59	0
				$1.1\epsilon^*$	6	3.70	6	5.48	0
				$1.2\epsilon^*$	6	3.43	6	5.32	0
pr04	192	4	4	$0.8\epsilon^*$	9	8.39	10	9.61	-10
				$0.9\epsilon^*$	9	8.19	10	9.24	-10
				ϵ^*	8	7.44	8	8.25	0
				$1.1\epsilon^*$	7	7.31	8	8.19	-13
				$1.2\epsilon^*$	7	7.08	8	7.95	-13
pr05	240	4	4	$0.8\epsilon^*$	12	15.93	14	20.19	-14
				$0.9\epsilon^*$	12	15.24	14	18.95	-14
				ϵ^*	10	14.82	11	18.39	-9
				$1.1\epsilon^*$	10	14.73	11	18.11	-9
				$1.2\epsilon^*$	10	14.22	11	17.78	-9
pr06	288	4	4	$0.8\epsilon^*$	14	16.54	16	19.50	-13
				$0.9\epsilon^*$	14	16.09	16	19.42	-13
				ϵ^*	13	15.61	14	19.32	-7
				$1.1\epsilon^*$	13	15.45	14	18.80	-7
				$1.2\epsilon^*$	12	15.39	13	18.53	-7
pr07	72	6	6	$0.8\epsilon^*$	4	1.29	4	1.93	0
				$0.9\epsilon^*$	4	1.24	4	1.86	0
				ϵ^*	3	1.18	3	1.77	0
				$1.1\epsilon^*$	3	1.14	3	1.69	0
				$1.2\epsilon^*$	3	1.11	3	1.60	0
pr08	144	6	6	$0.8\epsilon^*$	7	5.79	8	7.53	-13
				$0.9\epsilon^*$	7	5.19	7	7.44	0
				ϵ^*	6	4.82	6	7.39	0
				$1.1\epsilon^*$	6	4.60	6	7.22	0
				$1.2\epsilon^*$	6	4.49	6	7.16	0
pr09	216	6	6	$0.8\epsilon^*$	11	17.19	12	19.12	0
				$0.9\epsilon^*$	10	16.76	12	19.04	-17
				ϵ^*	10	16.29	11	18.96	-10
				$1.1\epsilon^*$	10	16.20	11	18.78	-10
				$1.2\epsilon^*$	10	16.11	11	18.53	-10
pr10	288	6	6	$0.8\epsilon^*$	14	19.07	16	19.61	-13
				$0.9\epsilon^*$	14	18.99	16	19.44	-13
				ϵ^*	12	18.94	13	19.10	-8
				$1.1\epsilon^*$	12	18.83	13	18.97	-8
				$1.2\epsilon^*$	12	18.53	13	18.86	-8

Table 4.7: Results on MDVRP instances with Rule R_2

Instance	N	M	T	ϵ	Heuristic (Ave.) K	Heuristic (Ave.) Time (min)	UTS (Ave.) K	UTS (Ave.) Time (min)	Gap (%)
pr01	48	4	4	$0.8\epsilon^*$	5	0.14	5	0.17	0
				$0.9\epsilon^*$	5	0.12	5	0.17	0
				ϵ^*	4	0.10	4	0.16	0
				$1.1\epsilon^*$	4	0.09	4	0.14	0
				$1.2\epsilon^*$	4	0.07	4	0.14	0
pr02	96	4	4	$0.8\epsilon^*$	9	0.43	10	0.84	0
				$0.9\epsilon^*$	9	0.36	9	0.77	-10
				ϵ^*	8	0.33	8	0.72	0
				$1.1\epsilon^*$	8	0.31	8	0.66	0
				$1.2\epsilon^*$	8	0.30	8	0.62	0
pr03	144	4	4	$0.8\epsilon^*$	13	1.08	14	1.55	-7
				$0.9\epsilon^*$	13	1.04	13	1.45	0
				ϵ^*	12	0.59	12	1.44	0
				$1.1\epsilon^*$	12	0.59	12	1.35	0
				$1.2\epsilon^*$	12	0.48	12	1.29	0
pr04	192	4	4	$0.8\epsilon^*$	17	3.90	18	4.72	-6
				$0.9\epsilon^*$	17	3.85	18	4.66	-6
				ϵ^*	16	3.80	17	4.60	-6
				$1.1\epsilon^*$	16	3.74	17	4.52	-6
				$1.2\epsilon^*$	16	3.70	17	4.47	-6
pr05	240	4	4	$0.8\epsilon^*$	21	7.53	22	9.59	-5
				$0.9\epsilon^*$	21	7.49	22	9.35	-5
				ϵ^*	20	7.34	21	9.33	-5
				$1.1\epsilon^*$	20	7.29	21	9.19	-5
				$1.2\epsilon^*$	20	7.22	21	9.14	-5
pr06	288	4	4	$0.8\epsilon^*$	25	9.01	27	9.84	-7
				$0.9\epsilon^*$	25	8.97	27	9.60	-7
				ϵ^*	24	8.80	25	9.49	-4
				$1.1\epsilon^*$	24	8.75	25	9.40	-4
				$1.2\epsilon^*$	24	8.69	25	9.33	-4
pr07	72	6	6	$0.8\epsilon^*$	7	0.33	7	0.64	0
				$0.9\epsilon^*$	7	0.29	7	0.59	0
				ϵ^*	6	0.27	6	0.54	0
				$1.1\epsilon^*$	6	0.25	6	0.49	0
				$1.2\epsilon^*$	6	0.24	6	0.46	0
pr08	144	6	6	$0.8\epsilon^*$	13	1.73	13	2.04	0
				$0.9\epsilon^*$	13	1.69	13	1.99	0
				ϵ^*	12	1.62	12	1.95	0
				$1.1\epsilon^*$	12	1.56	12	1.88	0
				$1.2\epsilon^*$	12	1.50	12	1.84	0
pr09	216	6	6	$0.8\epsilon^*$	20	7.88	22	8.91	-9
				$0.9\epsilon^*$	20	7.79	22	8.84	-9
				ϵ^*	19	7.77	21	8.79	-10
				$1.1\epsilon^*$	19	7.66	21	8.74	-10
				$1.2\epsilon^*$	19	7.59	21	8.63	-10
pr10	288	6	6	$0.8\epsilon^*$	25	8.85	26	9.82	-4
				$0.9\epsilon^*$	25	8.80	26	9.72	-4
				ϵ^*	24	8.70	25	9.65	-4
				$1.1\epsilon^*$	24	8.64	25	9.55	-4
				$1.2\epsilon^*$	24	8.59	25	9.48	-4

Table 4.8: Results on MDPVRP instances with Rule R_2

Instance	N	M	T	ϵ	Heuristic (Ave.)		UTS (Ave.)		Gap (%)
					K	Time (min)	K	Time (min)	
pr01	48	4	4	$0.8\epsilon^*$	5	0.35	5	0.50	0
				$0.9\epsilon^*$	5	0.31	5	0.47	0
				ϵ^*	4	0.28	4	0.44	0
				$1.1\epsilon^*$	4	0.25	4	0.41	0
				$1.2\epsilon^*$	4	0.22	4	0.37	0
pr02	96	4	4	$0.8\epsilon^*$	5	0.89	5	1.43	0
				$0.9\epsilon^*$	5	0.84	5	1.38	0
				ϵ^*	4	0.77	4	1.31	0
				$1.1\epsilon^*$	4	0.71	4	1.28	0
				$1.2\epsilon^*$	4	0.67	4	1.22	0
pr03	144	4	4	$0.8\epsilon^*$	9	4.89	10	6.46	0
				$0.9\epsilon^*$	9	4.83	10	6.41	-10
				ϵ^*	8	4.75	9	6.29	-11
				$1.1\epsilon^*$	8	4.62	9	6.18	-11
				$1.2\epsilon^*$	8	4.53	9	6.11	-11
pr04	192	4	4	$0.8\epsilon^*$	9	13.76	9	15.70	0
				$0.9\epsilon^*$	9	13.69	9	15.66	0
				ϵ^*	8	13.62	8	15.64	0
				$1.1\epsilon^*$	7	13.56	8	15.49	-13
				$1.2\epsilon^*$	7	13.45	8	15.53	-13
pr05	240	4	4	$0.8\epsilon^*$	13	20.65	14	24.55	-7
				$0.9\epsilon^*$	13	20.59	14	24.48	-7
				ϵ^*	12	20.53	13	24.33	-8
				$1.1\epsilon^*$	12	20.42	13	23.94	-8
				$1.2\epsilon^*$	12	20.35	13	23.85	-8
pr06	288	4	4	$0.8\epsilon^*$	14	17.47	16	21.51	-13
				$0.9\epsilon^*$	14	17.35	16	21.40	-13
				ϵ^*	13	17.30	14	21.34	-7
				$1.1\epsilon^*$	13	17.41	14	21.29	-7
				$1.2\epsilon^*$	13	17.23	14	21.20	-7
pr07	72	6	6	$0.8\epsilon^*$	7	1.49	7	2.09	0
				$0.9\epsilon^*$	7	1.44	7	2.06	0
				ϵ^*	6	1.40	6	2.01	0
				$1.1\epsilon^*$	6	1.34	6	1.97	0
				$1.2\epsilon^*$	6	1.29	6	1.89	0
pr08	144	6	6	$0.8\epsilon^*$	7	5.28	8	8.09	-13
				$0.9\epsilon^*$	7	5.19	7	8.04	0
				ϵ^*	6	5.18	6	7.95	0
				$1.1\epsilon^*$	6	5.11	6	7.94	0
				$1.2\epsilon^*$	6	5.04	6	7.88	0
pr09	216	6	6	$0.8\epsilon^*$	14	20.45	15	24.22	-7
				$0.9\epsilon^*$	14	20.35	15	24.17	-7
				ϵ^*	13	20.31	13	24.12	0
				$1.1\epsilon^*$	13	20.23	13	23.95	0
				$1.2\epsilon^*$	13	20.07	13	23.87	0
pr10	288	6	6	$0.8\epsilon^*$	19	23.19	20	26.86	-5
				$0.9\epsilon^*$	19	23.12	19	26.76	0
				ϵ^*	18	22.94	18	26.69	0
				$1.1\epsilon^*$	18	22.83	18	26.57	0
				$1.2\epsilon^*$	18	22.77	18	26.46	0

4.6 Conclusions

This paper presented a new modular heuristic algorithm for addressing several classes of multi-depot and periodic vehicle routing problems. In each of the considered problem classes, the goal is to determine the optimal fleet size when three constraints, i.e., vehicle capacity, route duration and budget constraints, are to be satisfied.

This paper introduced several methodological contributions, particularly, a self-learning mechanism that leads the algorithm to assign better visit patterns to customers, and also to assign customers to better depots as the solution process evolves. This learning mechanism, in addition to other components of the algorithm, provided the capability of the heuristic algorithm to reach high quality solutions.

To validate the efficiency of the proposed heuristic algorithm, different test problems, existing in the literature, were solved. The computational results showed that the proposed algorithm performs very well, for all problem instances.

Acknowledgements

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair, Collaborative Research and Development, and Discovery Grant programs, by the Fonds de Recherche du Québec - Nature et Technologies (FRQ-NT) through its Team Research Project program, and by the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants.

Chapter 5

An integrative cooperative search approach for a bi-objective vehicle routing problem

Abstract

In this paper, we address a bi-criteria multi-depot periodic vehicle routing problem where two conflicting objectives, i.e., total number of used vehicles and total traveled distance, are to be simultaneously optimized. To solve the problem, we design an algorithmic framework that is based on a parallel cooperative search approach called Integrative Cooperative Search (ICS). The proposed algorithm combines several exploration and exploitation strategies which result in producing good and diverse solutions. Extensive computational experiments show that the algorithm performs impressively well, in terms of solution quality, diversification level and computational efficiency.

keywords: Bi-criteria multi-depot periodic vehicle routing problem, Cooperative parallel methods, Integrative cooperative search, Multi-objective algorithms.

5.1 Introduction

The Vehicle Routing Problem (VRP) is one of the most important and widely studied combinatorial optimization problems, with many real-life applications in distribution and transportation logistics. Its basic version can be described as follows: a set of customers having deterministic demands have to be satisfied from a central depot with a fleet of homogeneous delivery vehicles of known capacity. Usually, the objective of VRPs is to minimize the total distance traveled by the vehicle fleet, but it is also common to minimize other objectives like the total transportation costs and the number of used vehicles (Toth and Vigo (2002)).

The VRP has been a challenging subject for many researchers since it was introduced by Dantzig and Ramser (1959) for solving the truck dispatching problem. A large variety of different optimization methods have been proposed and studied. However, VRP research has often been criticized for being too focused on idealized models with non-realistic assumptions for practical applications. As a result, researchers have turned to variants of the VRP which before were considered too difficult to solve. The variants include aspects of the VRP that are essential to the routing of vehicles in real problems. These extended problems are called Multi-Attribute VRPs (MAVRPs) (Rieck and Zimmermann (2006)).

In the last decade, different variations and specializations of MAVRPs, each reflecting various real-life applications, have been studied. However, surveying the literature, one can notice that: 1) the most MAVRPs have been often set up with a single objective function, which does not necessarily reflect the problems encountered in practical settings, and 2) the most common way to deal with MAVRPs, either single- or multi-objective, is the sequential approach. According to this method, one solves the problem one dimension at a time instead of addressing it comprehensively. It is well-known that this leads to suboptimal solutions.

Our objective is to contribute toward addressing the above two challenges by addressing a variant of the MAVRP in which a daily plan is computed for a homogeneous fleet of vehicles that depart from different depots and must visit a set of customers for delivery operations in a planning horizon. In this MAVRP, we consider maximum route duration constraint and an upper limit of the quantity of goods that each vehicle can transport. Moreover, two practical contradicting objectives often found in reality, i.e., total number of used vehicles and total traveled distance, are to be simultaneously minimized. To solve the problem, we propose an algorithmic framework that works based on the principles of a cooperative parallel algorithm namely Integrative Cooperative Search (ICS).

The remainder of this paper is organized as follows: Section 5.2 gives the problem statement. In Section 5.3, the literature survey relevant to the topic of this study is presented. The different components of the proposed ICS algorithm are described in Section 5.4. The experimental results are given in Section 5.5. Finally, Section 5.6 provides conclusions and the evaluation of the work.

5.2 Problem statement

In this section, we formally state the bi-objective MDPVRP, introducing the notations used throughout this paper. The MDPVRP can be defined as follows (Cordeau et al. (1997), Vidal et al. (2012a)): Consider an undirected graph $G(V, E)$. The node set V is the union of two subsets $V = V_C \cup V_D$, where $V_C = \{v_1, ..., v_n\}$ represents the customers and $V_D = \{v_{n+1}, ..., v_{n+m}\}$ includes the depots. With each node $i \in V_C$ are associated a deterministic demand q_i and a service time s_i. The edge set E contains an edge for each pair of customers and for each depot-customer combination. There are no edges between depots. With each edge $(v_i, v_j) \in E$ is associated a travel cost c_{ij}. The travel distance for arriving to node j from node i (t_{ij}) is considered equal to c_{ij}. A homogeneous fleet of vehicles of known capacity Q is available at each depot. Moreover, the MDPVRP has a planning horizon, say T periods. Each customer i is characterized by a service frequency f_i, stating how often within this T periods this customer must be visited and a list L_i of possible visit-period combinations, called patterns. Each vehicle performs only one route per period and each vehicle route must start and finish at the same depot given that the travel duration of the route should not exceed D. The MDPVRP aims to design a set of vehicle routes servicing all customers, such that vehicle-capacity and route-duration are respected, and two conflicting objectives, i.e., total number of used vehicles and total traveled distance, are simultaneously minimized.

To formulate the bi-objective MDPVRP as a mixed-integer linear program, we use the notations introduced in Vidal et al. (2012a) as follows:

- **Parameters**
  - a_{rl}: A binary constant which equals to 1 if period *l* belongs to visit pattern $r \in L_i$, otherwise 0.
- **Variables**
  - x_{ijklo}: A binary variable which equals to 1 if customer *j* is immediately visited after customer *i* by vehicle *k* departing in period *l* from depot o, otherwise 0.
  - y_{iro}: A binary variable which equals to 1 if visit pattern $r \in L_i$ and depot o are assigned to customer *i*, otherwise 0.
  - K_{lo}: An integer variable which shows the number of vehicles assigned in period *l* to depot o.

The mathematical formulation of the bi-objective MDPVRP explained above can be summarized as follows:

$$\mathit{Min} \quad f_1 = \max_{l=1,\ldots,T} \{ \sum_{v_o \in V_D} K_{lo} \} \tag{5.1}$$

$$\mathit{Min} \quad f_2 = \sum_{v_i \in V} \sum_{v_j \in V} \sum_{k=1}^{\infty} \sum_{l=1}^{T} \sum_{v_o \in V_D} t_{ij} x_{ijklo} \tag{5.2}$$

S.T

$$\sum_{r \in L_i} \sum_{v_o \in V_D} y_{iro} = 1 \qquad (v_i \in V_C); \quad (5.3)$$

$$\sum_{v_j \in V} \sum_{k=1}^{\infty} x_{ijklo} - \sum_{r \in L_i} a_{rl} y_{iro} = 0 \qquad (v_i \in V_C; v_o \in V_D; l = 1, ..., T); \quad (5.4)$$

$$\sum_{v_j \in V} x_{jiklo} - \sum_{v_j \in V} x_{ijklo} = 0 \qquad (v_i \in V; v_o \in V_D; k = 1, 2, ...; l = 1, ..., T); \quad (5.5)$$

$$\sum_{v_j \in V} x_{ojklo} \leq 1 \qquad (v_o \in V_D; k = 1, 2, ...; l = 1, ..., T); \quad (5.6)$$

$$\sum_{v_j \in V} x_{ijklo} = 0 \qquad (v_i \in V_D; v_o \in V_D; v_i \neq v_o; k = 1, 2, ...; l = 1, ..., T); \quad (5.7)$$

$$\sum_{k=1}^{\infty} \sum_{v_j \in V} x_{ojklo} = K_l o \qquad (l = 1, ..., T; v_o \in V_D); \quad (5.8)$$

$$\sum_{v_i \in V} \sum_{v_j \in V} q_i x_{ijklo} \leq Q \qquad (v_o \in V_D; k = 1, 2, ...; l = 1, ..., T); \quad (5.9)$$

$$\sum_{v_i \in V} \sum_{v_j \in V} (t_{ij} + s_i) x_{ijklo} \leq D \qquad (v_o \in V_D; k = 1, 2, ...; l = 1, ..., T); \quad (5.10)$$

$$\sum_{v_i \in S} \sum_{v_j \in S} x_{ijklo} \leq |S| - 1 \, (S \in V_C; |S| \geq 2; v_o \in V_D; k = 1, 2, ...; l = 1, ..., t); \quad (5.11)$$

$$x_{ijklo} \in \{0, 1\} \qquad (v_i \in V; v_j \in V; v_o \in V_D; k = 1, 2, ...; l = 1, ..., T); \quad (5.12)$$

$$y_{iro} \in \{0, 1\} \qquad (v_i \in V; r \in L_i; v_o \in V_D); \quad (5.13)$$

In the above model, the first objective function (f_1) attempts to minimize the total number of vehicles used over the planning horizon, while the second objective function (f_2) minimizes the total traveled distance. Constraints (5.3) show that only one visit pattern and one depot should be assigned to each customer. Constraints (5.4) ensure that each customer is visited in the periods corresponding to the assigned visit pattern by a vehicle departing from the chosen depot. Constraints (5.5) guarantee that when a vehicle arrives at a customer on a given period, it also leaves that customer on the same period. Constraints (5.6) impose that each vehicle is used at most once every period. Constraints (5.7) enforce the compatibility issues between depot assignments and and route starting and ending points. Constraints (5.8) show the number of vehicles assigned to each depot in each period. Constraints (5.9) and (5.10) define respectively the capacity and maximum route duration restrictions. Finally, Constraints (5.11) are the standard subtour elimination constraints.

5.3 Literature review

In this section, we focus the literature review on the MDPVRP. The aim is to present the different solution approaches that have been developed to solve both the single- and multi-objective versions of the MDPVRP and, in doing so, identify which of these methods are the most efficient.

By studying the literature, one can perceive that the solution methods, targeting the MD-PVRP, are divided into three main groups: 1) Exact methods, which are limited in the size of instances they may handle, 2) classical heuristics, which often solve the problem in a sequential manner, and 3) Sophisticated meta-heuristics and parallel algorithms, which tackle the problem by simultaneously optimizing all the involved attributes.

To the best of our knowledge, the only exact method, belonging to the first group, was the one designed by Mingozzi (2005). In the proposed method, first, an integer programming model which is an extension of the set partitioning formulation of the CVRP is described. Then, an exact method for solving the problem, which uses variable pricing in order to reduce the set of variables to more practical proportions, is proposed. The pricing model is based on the bounding procedure for finding near optimal solutions of the dual problem of the LP relaxation of the proposed integer programming model. The bounding procedure is an additive procedure that determines a lower bound on the MDPVRP as the sum of the dual solution costs obtained by a sequence of five different heuristics for solving the dual problem, where each heuristic explores a different structure of the MDPVRP. Three of these heuristics are based on relaxations, whereas the two others combine subgradient optimization with column generation. We also aware of three heuristic algorithms in this category.

We are aware of three heuristics in the second group. Hadjiconstantinou and Baldacci (1998) formulated the problem of providing maintenance services to a set of customers as the MDPVRP with Time Windows (MDPVRPTW). To solve the problem, the authors proposed a four-phase algorithm. In the developed algorithm, all customers are first assigned to particular depot. Then, customer visits are successively inserted among available periods to obtain feasible visit combinations. In the third phase, each of the depot-period VRP sub-problems is separately solved using a tabu search algorithm. Finally, in the last phase, solutions obtained during the optimization process are improved by modifying the period or depot assignments through a 3-opt procedure. Kang et al. (2005) designed a two-phase heuristic method to address the same problem. In the proposed method, all feasible schedules are first generated from each depot for each period and, then, the set of routes are determined through using the shortest path problem. Parthanadee and Logendran (2006) proposed a tabu search heuristic to tackle the problem considered by Hadjiconstantinou and Baldacci (1998). In this algorithm, all the initial assignments are built by cheapest insertion, where each customer is assigned to its nearest depot and is given its most preferred visit pattern. In the improvement phase, a neighbourhood search is defined by depot and visit pattern interchanges.

We are also aware of two contributions belonging to the third group. The first contribution was the evolutionary meta-heuristic proposed by Vidal et al. (2012a). The authors developed a hybrid Genetic Algorithm (GA) to tackle the MDPVRP and two of its special cases, i.e., the Multi-depot VRP (MDVRP) and the Periodic VRP (PVRP). The main of the proposed GA is a new population-diversity management mechanism which allows a broader access to reproduction, while preserving the memory of what characterizes good solutions represented by the elite individuals of the population. The second contribution was the cooperative parallel algorithm designed by Lahrichi et al. (2012). The authors proposed a structured cooperative parallel search method, called Integrative Co-operative Search (ICS), to solve highly complex combinatorial optimization problems. The proposed ICS framework involves problem decomposition by decision sets, integration of elite partial solutions yielded by the sub-problems, and adaptive guiding mechanism. The authors used the MDPVRP to present the applicability of the developed methodology.

Vehicle routing problems have also been studied as a multi-objective problem. Jozefowiez et al. (2008) provided a comprehensive overview of the research into vehicle routing problems with several objectives. The authors presented that the most methodological developments in multi-objective VRPs target a special problem type, the CVRP or the VRPTW and other variants including more complicated attributes have not been addressed with the same degree of attention. Table 5.1 provides a general overview of the multi-objective VRP contributions reported by Jozefowiez et al. (2008).

This brief literature survey supports the general statements mentioned in the first section that: 1) the MDPVRP is among the MAVRP variants which did not receive adequate degree of attention, even as a single-objective problem, 2) solution algorithms capable of efficiently solving the single-objective MDPVRP as a whole by simultaneously considering all attributes are scarce, and 3) to the best of our knowledge, there is no significant contribution in the literature tackling the multi-objective MDPVRP. These three remarks motivated us to develop an ICS approach to solve the considered bi-objective MDPVRP. The details of the proposed ICS algorithm are given in the following section.

5.4 The integrative cooperative search for the bi-objective MD-PVRP

The solution approach that we propose performs based on the integrative cooperative search paradigm designed by Lahrichi et al. (2012) but includes some advanced and well-structured features which enable the algorithm to generate good and diverse non-dominated solutions as near as possible to the true Pareto frontier. We initiate this section by introducing the fundamental concepts underlying the general ICS methodology, particularly the decision-set attribute-set decomposition and the ICS algorithmic structure. Then, concepts, challenges, and different components of the ICS algorithm, designed in this paper to tackle the bi-criteria MDPVRP, are thoroughly discussed.

5.4.1 The general ICS methodology

The ICS methodology, as a multi-thread cooperative search method, was designed by Lahrichi et al. (2012) to efficiently address multi-attribute combinatorial optimization problems. The general structure of ICS consists of three main building blocks, i.e., decomposition, integration, and global search coordination, which cooperate using the central-memory cooperative search

Table 5.1: Multi-objective VRP contributions

Problem type	Author(s)	Objective functions	Solution method
CVRP	Park and Koelling (1986) Park and Koelling (1989)	Max. the travel distance of vehicles, Min. the total fulfilment of emergent services, Min. the conditional dependencies of stations	Goal programming heuristic
	Murata and Itai (2006)	Min. the total length of the routes, Max. the balance of the routes	Pareto approach
	Murata and Itai (2007)	Min. the total delivery cost, Min. the maximum cost, Min. the number of vehicles, Min. the total delay	Memetic algorithm
VRPTW	Hong and Park (1999)	Min. the total vehicle travel time, Min. the total customer waiting time	Goal programming
		Min. the number of vehicles,	
	Tan et al. (2006)	Min. the total length, Min. the number of vehicles	Genetic algorithm
	Ombuki et al. (2006)	Min. the total length, Min. the number of vehicles	Genetic algorithm
	Geiger (2008)	Min. the total distance, Min. the total deviation from time window bounds,	Genetic algorithm algorithm
PVRP	Ribeiro and Lourenço (2001)	Min. the traveled distance, Max. the work levels balance, Max. the customer satisfaction	Iterated local search

paradigm.

ICS initiates solving a problem by decomposing it into some more tractable and easier sub-problems, each defined by suitably selected subsets of decision-set attributes of the problem. By decision-set attributes, we mean the sets of decisions defining the particular problem setting. Lahrichi et al. (2012) enumerated various reasons for using such a decomposition which primarily include several computational benefits in terms of both reliability and algorithm speed, the enhancement of parallel and distributed computing, reduced programming and debugging effort, and the possibility of employing different solution techniques for the different decomposed sub-problems. Following the initial decomposition of the original problem, each resulting sub-problem is addressed by one or several solution methods, called *partial solvers*. Each partial solver may be a simple constructive method, exact or meta-heuristic algorithm and aims to generate and send a set of elite partial solutions to a partition of the central-memory, namely *partial set*. In the general ICS, a guidance mechanism, called Local Search Coordinator (LSC), is usually embedded to each partial solver to locally monitor the performance of the partial solver and to act when this performance becomes unsatisfactory through sending appropriate instructions. The LSC can force a partial solver to restart from a suitable solution in memory, modify the search parameters, or even change the solution method.

Concurrently with partial solver activities, some integrators combine the elite partial solutions of the partial set into complete ones and, eventually, improve them. (Note that, all solutions in the ICS are complete; the terms ”partial” and ”complete” solutions are used, however, to distinguish between solutions generated in the decomposition and integration phases). Integrators play an essential role in the ICS methodology. While partial solvers address a single aspect of the original problem, integrators build complete solutions by mixing elite partial solutions with promising features. Similarly to the partial solvers, integrators can be very simple procedures or sophisticated exact and meta-heuristic algorithms. All the elite complete solutions found by integrators are sent to another partition of the central-memory, called the complete set.

Another component of the ICS methodology is a controlling mechanism, namely Global Search Coordinator (GSC), which guides the global search by sending appropriate instructions to partial solvers and, eventually, integrators. The GSC guides the search by monitoring the central-memory and through statistical information, memories, on the evolution of solutions in the complete and partial sets, the contribution of solutions and their components to the evolution of the search, and relative performances of partial solvers. One possible implementation of the ICS methodology described above is depicted by Figure 5.1.

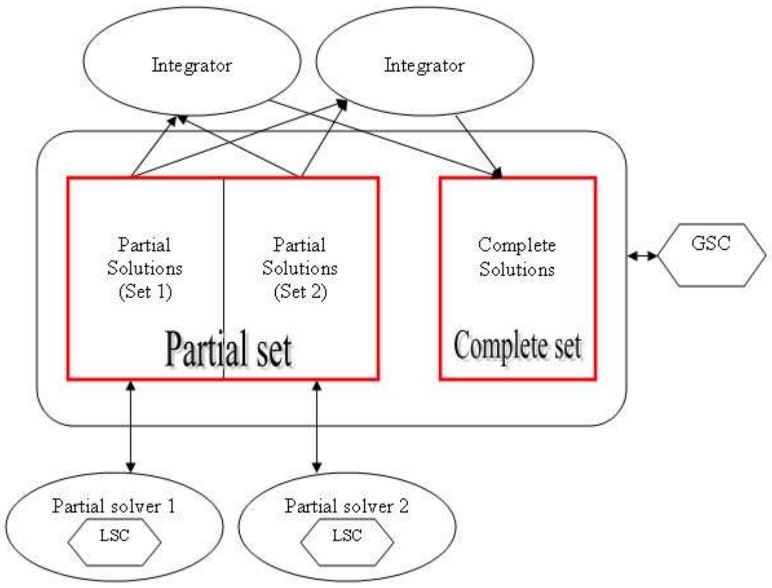


Figure 5.1: The integrative cooperative search scheme

5.4.2 The proposed ICS algorithm

The goal of this section is to present and discuss the proposed bi-objective ICS concepts, its structure, main building blocks, and operating principles. We first thoroughly present how the ICS works in the decomposition phase. Then, we proceed with detailed discussions of the integration phase. Thereafter, the proposed guiding mechanism is described in details. Finally, the termination criteria are given.

Decomposition phase

As previously mentioned, the underlying idea of following the decomposition phase in the ICS is to simplify an original and relatively difficult problem as a collection of auxiliary and comparably easier sub-problems that can be solved separately and properly by dedicated partial solvers. Therefore, an important question to answer is how to perform this decomposition. A straightforward approach often used in the general ICS is to use the decision-set attribute-based decomposition. Recall that, in this decomposition, the objective-set remains unchanged for each sub-problem and only the decision-set attributes not part of the sub-problem definition are simply fixed (Note that by objective-set, we mean the set of the objective functions defined for a problem). In our opinion, this approach has the drawback, while dealing with an multi-objective problem, of creating multi-objective sub-problems that may be as challenging as the original problem to solve. We therefore propose a decomposition scheme which takes into account the decision-set as well as the objective-set to split the main problem. Towards this end, a new Hierarchical Decomposition Procedure (HDP) is proposed to decompose the bi-objective MDPVRP.

The HDP, used in this paper, is the process of breaking down the original problem into successive layers of more manageable and comprehensive pieces. More precisely, the HDP creates two different layers. In the first layer, the MDPVRP is decomposed into two bi-objective vehicle routing problems with exactly one less attribute, i.e., PVRP and MDVRP, by respectively fixing the depot and period decision sets. In other words, in the PVRP, the pair (customer, depot) is set to be fixed, whereas in the MDVRP, the pair (customer, visit combination) is considered as a frozen pair. Since the simultaneous minimization of two contradicting objective functions, in each of the sub-problems generated in the first layer, is a challenging task, each sub-problem is decomposed once again into two simpler problems, each constructed using the well-known ϵ-constraint method. The ϵ-constraint method generates single objective problems, called ϵ-constraint problems, by transforming all but one objectives into constraints. The upper bounds of these constraints are given by the ϵ-vector and, by varying it, the exact Pareto front can theoretically be generated (Bérubé et al. (2009)). The main advantage of applying such a decomposition procedure is that working on sub-problems having smaller number of objectives and attributes, instead of considering a large-size multi-objective problem, provide relatively high quality and diverse solutions rapidly. On the other hand, well-known single-objective optimization methods, existing in the literature, may be used to solve each of the sub-problems. Figure 5.2 shows the HDP scheme.

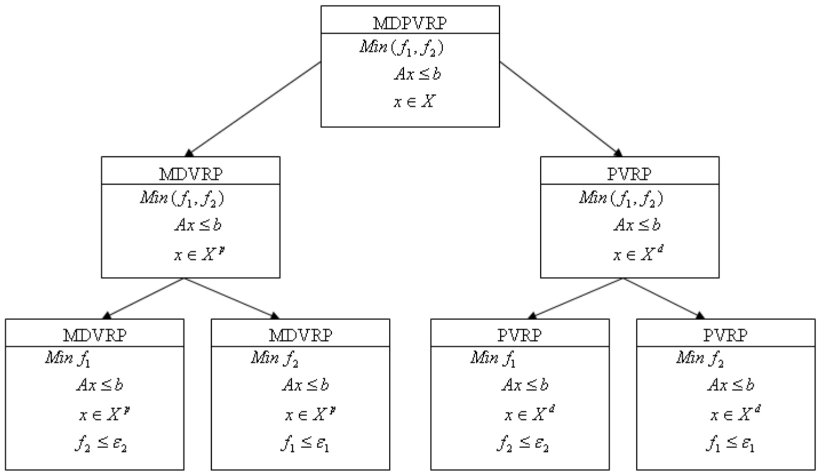


Figure 5.2: The decomposition scheme

In Figure 5.2, f_1 and f_2 respectively represent the first and second objective functions defined in the mathematical programming model of Section 2. $Ax \leq b$ is the union of Constraints 3-11 and X shows the set of decision variables 12-13. Moreover, X^p and X^d are defined as the set of decision variables, in which the period and depot decision sets are respectively fixed.

Once the decomposition and the sub-problem hierarchy have been defined, the sub-problems of the second layer are simultaneously solved by some partial solvers. For the sub-problems where the total traveled distance is considered as the objective function, there exist various efficient optimization algorithms, i.e., exact, heuristic and meta-heuristic methods. Each of these algorithms can be directly applied as a solver to generate partial solutions. In this paper, we used the Unified Tabu Search (UTS) implemented in Lahrichi et al. (2011). On the other hand, in the other sub-problems where the total number of used vehicles is to be minimized, there are very limited number of papers previously published in the literature. In this study, we use the three-phase heuristic algorithm developed by Rahimi-Vahed et al. (2012b). The details of the above partial solvers are described in Section 5.5.

Each of the above partial solvers solves a dedicated sub-problem in α different runs, each considering a different value for ϵ. To modify the value of ϵ, from one run to another, a Local Search Coordinator (LSC) is assigned to each partial solver. The LSC, implemented in this paper, is an adaptive updating procedure similar to the one proposed by Laumanns et al. (2005). In this updating procedure, ϵ is first set to a large value, denoted by Δ, in order to obtain the lower bound on the value of Pareto solutions. Then, Δ iteratively decreases to generate other solutions of the Pareto front. For more detailed explanations, readers refer to Laumanns et al. (2005). Note that, the whole procedure, described in this section, is repeated for β successive iterations. Algorithm 4 schematically shows how the ICS works in the decomposition phase.

Algorithm 4 The decomposition phase

```
$\epsilon \leftarrow \Delta$.
Set $\nu$ as an empty list.
**for** $i$=1...$\beta$ **do**
    Fix the value of the corresponding decision-set attribute.
    **for** $j$=1...$\alpha$ **do**
        Solve the constructed sub-problem by the dedicated partial solver.
        Add the solution obtained by the partial solver to $\nu$.
        Update the $\epsilon$ value using LSC.
    **end for**
**end for**
Report the solutions of $\nu$ as the corresponding partial solutions.
```

After each of the partial solvers generates its corresponding partial solution set, obtained elite partial solutions are sent to the partial set of the central memory. The partial set plays a key role on the performance of the developed ICS. What solutions are included in the partial set, how good and how diversified they are, have a major impact on the quality of new solutions generated by the ICS. More precisely, the partial set is the union of four different subsets, each preserving elite partial solutions sent by a partial solver. In this phase, it should be carefully determined how each of these subsets are initially constructed and how they are iteratively updated in the course of the optimization procedure.

In this paper, the maximum size of each subset is fixed to a predetermined positive value denoted by b. Moreover, each subset consists of two different parts. The first part includes high quality solutions, while diverse solutions are kept in the second part. To construct the ith subset of the partial set, the partial solutions generated by the ith partial solver are first partitioned into different Pareto fronts using the non-domination sorting algorithm. Note that, the first front is formed by non-dominated partial solutions, the second front covers all those partial solutions which are dominated by each solution of the first front and dominate all solutions of the other fronts, and so on. Then, $\lceil b/2 \rceil$ solutions that have lower front numbers are successively added to the first part.The construction procedure continues by computing the minimum Hamming distance of each remaining partial solution to the solutions added to the first part. Finally, the second part is filled up by the $\lfloor b/2 \rfloor$ solutions that have larger minimum Hamming distance values. Note that, the Hamming distance is used as a measure to diversify the solution space. This measure is the number of positions in two strings of equal length for which the corresponding elements are different. Put another way, it measures the number of substitutions required to change one into the other. Figure 5.3 schematically depicts how the partial set is constructed.

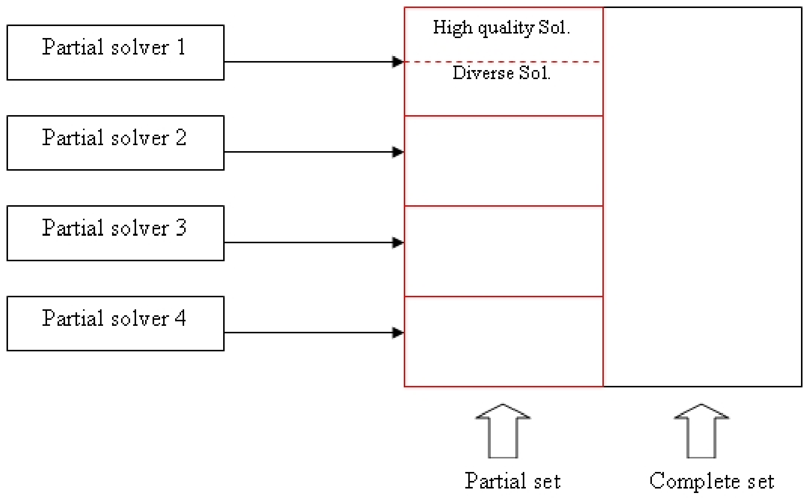


Figure 5.3: The partial set

Integration phase

As previously described at the beginning of this section, the decomposition of a complex problem very often needs to be coupled with an integration task, i.e., an algorithmic manner to use the solutions obtained from the sub-problems to build a solution of the original problem (Lahrichi et al. (2012)). This necessitates the development of an integrator which should address three, possibly contradictory, challenges: 1) using an efficient way to select partial solutions from the partial set, 2) generating high quality complete solutions to the original problem by combining selected partial solutions, and 3) being computationally efficient.

The integrator that we proposed is inspired by a swarm intelligence-based method, called Shuffled Frog-Leaping Algorithm (SFLA). The SFLA is a population-based meta-heuristic proposed to perform an informed and non-blind heuristic search to seek prominent solutions of a combinatorial optimization problem. This meta-heuristic works based on evolution of memes carried by the interactive individuals, and a global exchange of information among themselves (Eusuff et al. (2006)). In the SFLA, the population consists of a set of frogs (solutions) that is partitioned into sub-populations referred to as memeplexes. The different memeplexes are considered as different cultures of frogs, each performing a local search. Within each memeplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of memetic evolution. After a defined number of memetic evolution steps, ideas are passed among memeplexes in a shuffling process. The local search and the shuffling processes continue until predefined stopping criteria are met.

Compared to well-known evolutionary meta-heuristics, genetic algorithm for example, the advantages of the SFLA are that it is easy to implement and there are few parameters to adjust. Moreover, The prominent features of the SFLA, such as escaping from local optima traps, global optimization, good robustness, simple mechanism and fast convergence, make the SFLA as a promising optimization approach. These advantages and benefits motivated us to develop a SFLA-based integrator to re-create solutions of the bi-objective MDPVRP using the partial solutions of the partial set as the initial solutions.

The proposed integrator initiates by dividing partial solutions of the partial set into different sub-populations, each independently evolving to search the solution space in different directions. Towards this end, all the partial solutions preserved in the partial set are first copied into a virtual list and, then, are ordered, within the list, in non-domination sense. Finally, partial solutions of the ith front are sent to the ith sub-population. The idea involved in such a sub-populations construction is to search different regions (not only promising ones which include non-dominated solutions) to avoid quickly trapping on local Pareto optimal solutions.

The integrator continues by implementing a neighbourhood search within each constructed sub-population. The neighbourhood search, often used in a general SFLA, selects the worst and best solutions of a sub-population and generates a single new solution by modifying the structure of the worst solution based on the characteristics of the best solution. This neighbourhood search is very computationally efficient but, based on our experiments, it is very hard to handle and often fails to produce good results when the selected solutions are structurally very different. The neighbourhood search that we implemented generates a path of δ solutions ($\delta > 1$) connecting the selected worst and best solutions and tries to eventually introduce good characteristics of the best solution into the new generated solutions. The proposed neighbourhood search performs similar to the Complete Relinking Strategy (CRS) proposed by Rahimi-Vahed et al. (2012a). The CRS is a moving mechanism that the authors proposed in their Path Relinking Algorithm (PRA) to generate high quality solutions of a single-objective MDPVRP. All the components of our neighbourhood search are the same as designed in Rahimi-Vahed et al. (2012a), except the two following modifications (For more details, readers should refer to Rahimi-Vahed et al. (2012a).):

1. The neighbourhood search is not permitted to generate infeasible solutions.
2. The criteria based on which a new generated solution is accepted or rejected are modified as follows:
    - If the new generated solution is infeasible, the old solution remains unchanged.
    - If the new solution is feasible and dominates the old solution, the old solution is replaced by the new one.
    - If the new solution is feasible and dominated by the old solution, the old solution remains unchanged.
    - If the new solution is feasible and the two solutions are non-dominated, the old solution is replaced by the new one if the crowding distance of the old solution is less than or equal to the crowding distance of the new solution

If the above neighbourhood search fails to locally improve a selected worst solution within a sub-population, the integrator improves it globally. Towards this end, the integrator applies the same neighbourhood search on the selected worst solution and improves it, this time, based on the characteristics of the best solution of the entire search space.

After the neighbourhood search is repeated for λ successive iterations within each sub-population, the search space is forced to be shuffled and sub-populations are re-created. This shuffling procedure helps to construct better sub-populations reflecting more promising regions of the search space. The whole procedure, described in this section, is repeated γ iterations. Algorithm 5 shows how the proposed integrator works.

Algorithm 5 The integrator

```
Divide partial solutions of the partial set into different sub-populations.
for $i$=1...$\gamma$ do
    For each sub-population:
    for $j$=1...$\lambda$ do
        Determine the position of the best and worst solutions of the sub-population.
        Set the worst solution as the current solution.
        for $k$=1...$\delta$ do
            Apply the neighbourhood search on the current solution and generate a new
            solution.
            if the new solution is feasible and better than the current solution. then
                Set the new solution as the current solution.
                Update the complete set.
            end if
        end for
        if the neighbourhood search does not generate any improving feasible solution. then
            Go to the globally improving phase.
        end if
        Upgrade the sub-population.
    end for
    Shuffle the search space and re-construct sub-populations.
end for
```

Guiding phase

As previously mentioned in this section, the optimization procedure of the designed ICS is controlled by a guiding mechanism called Global Search Coordinator (GSC). The GSC, implemented in this paper, repeatedly guides the global search through learning the partial solvers and, eventually, the integrator how to assign better depot and visit pattern to each customer as the ICS reaches its termination criteria. Towards this end, the GSC uses a two-phase algorithm whose details are summarized as follows:

1. **Information extracting phase**: In this phase, the GSC extracts the required information from the non-dominated solutions of the complete set using a Pattern Identification Mechanism (PIM). The PIM is a simple procedure performing based on the inclusion of pairs (customer, depot) and (customer, visit pattern) in the solutions of the complete set. More precisely, the goal of the PIM is to recognize the most frequent depot and visit pattern based on which each of the customers has been visited in the complete set. A pair (customer, depot) ((customer, visit pattern)) with the highest inclusion frequency reflect this fact that the corresponding customer may be serviced by that depot (visit pattern) in a Pareto-optimal solution. We call a depot (visit pattern) with the highest frequency as potentially good depot (visit pattern).
2. **Information evaluating phase**: In this phase, the GSC evaluates the extracted information in order to send appropriate instructions to the partial solvers. Towards this end, the GCS successively selects the customers and investigates the following criteria: If the selected customer is visited by its potentially good depot (visit pattern) in more than $\%\theta$ of the solutions existing in the complete set, the GSC enforces all the partial solvers to assign that customer to its potentially good depot (visit pattern). Otherwise, the GCS reports the lack of sufficient information and, in such a case, each partial solver assigns the corresponding customer to a depot (visit pattern) using its own assignment mechanism.

Algorithm 6 summarizes the global search coordinator used in the ICS.

Termination criterion

It is a condition that terminates the optimization procedure. In this paper, the two following stopping criteria are considered:

- The algorithm is stopped if no new non-dominated solution is added to the complete set for μ successive iterations. μ is a positive value which is determined at the beginning of the ICS. Or,

Algorithm 6 The GSC

```
for $i$=1...$n$ do
    -Information extracting phase:
        Select customer $i$.
        Determine depot (visit pattern) by which customer $i$ is visited the most in
        non-dominated solutions
        of the complete set.
        $R \leftarrow$ The number of solutions in which customer $i$ is visited by the chosen depot (visit
        pattern).
    -Information evaluating phase:
    if $R \geq \%\theta\times$ the complete set size then
        Fix the selected depot (visit pattern) for customer $i$ in all partial solvers.
    end if
end for
```

- The algorithm is terminated if it passes a maximum allowable running time.

5.5 Computational results

In this section, we evaluate the performance of the proposed ICS algorithm based on different test problems. The only MDPVRP instances existing in the literature are those proposed by Vidal et al. (2012a). The authors created 10 different problems whose main characteristics are presented in Table 5.2.

Table 5.2: Problem instances

Instance	*n*	*m*	*T*
pr01	48	4	4
pr02	96	4	4
pr03	144	4	4
pr04	192	4	4
pr05	240	4	4
pr06	288	4	4
pr07	72	6	6
pr08	144	6	6
pr09	216	6	6
pr10	288	6	6

The ICS is executed on each of the above problems and its performance is compared, based on some comparison metrics, with two algorithms existing in the literature:

1. A modified version of the Unified Tabu Search (UTS) implemented in Lahrichi et al. (2011): The original UTS, developed by Lahrichi et al. (2011), was proposed to solve a single-objective MDPVRP, where the objective is the minimization of the total distance traveled over the planning horizon. We modified this algorithm to make it applicable to the bi-objective problem addressed in this paper. In our implementation, the UTS tackles the problem using an ϵ-constraint method, where the total traveled distance remains as the objective and the total number of used vehicles is added into the constraints set and bounded by a positive ϵ value. All the components of our algorithm are the same as developed by Lahrichi et al. (2011), except the penalty function that was modified to handle ϵ-constraint violations. Moreover, the procedure used to update the ϵ value is similar to the one proposed by Laumanns et al. (2005).

2. The Modular Heuristic Algorithm (MHA) of Rahimi-Vahed et al. (2012b): The MHA is a three-phase optimization algorithm that the authors proposed to address three single-objective vehicle routing problems, i.e., multi-depot VRP, periodic VRP and multi-depot periodic VRP. In each of these problems, the total number of used vehicles was set as the objective function and three kinds of constraints, i.e., vehicle capacity, route duration and budget constraints, were considered. The authors defined the budget constraint as a Travel-Distance Budget (TDB) constraint which imposes a threshold ϵ on the total distance traveled by vehicles for delivery operations. The only difference between our implementation and the original MHA is that the ϵ value is not considered as a fixed parameter and it is iteratively updated using the same updating procedure proposed by Laumanns et al. (2005) in order to generate different solutions of the Pareto front.

The experimentation of the proposed ICS algorithm has been conducted on quad-core AMD Opteron 248 processors, with 2,2 GHz clock speed. The procedure has been coded in C++, with the communication layer written in OpenMP 2.0.

Different aspects of the experimental results are discussed as follows: In Section 5.5.1, we first use a well structured algorithm to calibrate all the parameters involved in the ICS. Then, in Section 5.5.2, all the comparison metrics, used to investigate the performance of the ICS, and computational results are given in details.

5.5.1 Parameter setting

Like the most optimization algorithms, the developed ICS relies on a set of related parameters. Table 5.3 presents a summary of all the parameters used in the ICS.

Table 5.3: Parameters of the heuristic algorithm

Symbol	Description
α	The number of times that ϵ is modified
β	The number of iterations that attributes are fixed in the first layer of HDP
b	The maximum size of each subset in the partial set
λ	The number of times that the neighbourhood of the integration phase iterates within each sub-population
γ	The number of iterations that the integrator is repeated
μ	Maximum allowable number of iterations that the complete set can remain unchanged

There are various robust methods in the literature to tune parameters used in a optimization algorithm. Recently, Smith and Eiben (2010) developed a sophisticated calibration method called Relevance Estimation and VAlue Calibration (REVAC) which is capable of finding good parameter values for a set of problems. The REVAC method provides an information theoretic measure on how sensitive a parameter is to the choice of its value. This can be used to estimate the relevance of parameters, to choose between different possible sets of parameters, and to allocate resources to the calibration of relevant parameters. Technically, REVAC is a heuristic generate-and-test method that is iteratively searching for the set of parameter vectors of a given EA with a maximum performance. In each iteration a new parameter vector is generated and its performance is tested. Testing a parameter vector is done by executing the EA with the given parameter values and measuring the EA performance. A detailed explanation of REVAC can be found in Smith and Eiben (2010).

In this paper, we adopted this method to calibrate the parameters used in the ICS. Table 5.4 shows the calibration results together the range of values we estimated to be appropriate for each of the involved parameters.

Table 5.4: Calibration results

Symbol	Range	Value
α	[5, 50]	10
β	[5, 100]	10
b	[10, 100]	20
λ	[10, 40]	16
γ	[50, 300]	100
μ	[500, 5000]	1000

5.5.2 Comparative results

In this section, we first introduce the five comparison metrics that we used to validate the good performance of the proposed ICS algorithm:

- **The Number of Non-Dominated Solutions (N.N.D.S)**- This metric shows the number of non-dominated solutions that each algorithm is able to find.

- **Spacing Metric (SM)**- The spacing metric allows us to measure the uniformity of the spread of non-dominated solutions found by the ICS. The definition of this metric is the following:

$$SM = [\frac{1}{N-1} \times \sum_{i=1}^{N} (\bar{d} - d_i)^2] \tag{5.14}$$

  where *N* is the total number of found non-dominated solutions, d_i is the Euclidean distance between non-dominated solution *i* and the closest belonging to the complete set, and $\bar{d}$ is the mean value of all d_i.

- **Diversification Metric (DM)**- This metric measures the spread of found non-dominated solutions. Its definition is the following:

$$DM = \sqrt{\sum_{i=1}^{N} d_i'} \tag{5.15}$$

  where d_i' is the Euclidean distance between non-dominated solution *i* and the farthest solution existing in the complete set.

- **Quality Metric (QM)**- This metric is simply measured by putting together the non-dominated solutions found by two algorithms, i.e. *A* and *B*, and reporting the ratio of the number of non-dominated solutions which are generated by algorithm *A* to the number of non-dominated solutions which are found by algorithm *B*.

- **Time Metric (TM)**- This metric simply shows the execution time of each algorithm.

We tested the developed ICS algorithm on the problem instances described at the beginning of this section. To solve these problems, the maximum running time is set to 30 minutes. Table 5.5 shows the average number of non-dominated solutions that each algorithm was able to find in 10 independent runs.

Table 5.5: Number of non-dominated solutions found by the algorithms

Instance	ICS	MHA	UTS
pr01	6	6	6
pr02	8	8	8
pr03	11.25	10.66	10.21
pr04	14.77	12.05	11.36
pr05	13.31	11.80	9.74
pr06	16.29	13.45	11.73
pr07	8	7.12	6.62
pr08	10.22	7.29	6.14
pr09	19.91	14.72	12.27
pr10	22.45	17.12	15.49

As shown in Table 5.5, the proposed ICS algorithm produces more non-dominated solutions compared to the other algorithms. The obtained results show that if the ICS is used as a decision support tool, it may generate more solution alternatives for the decision maker. The average gap between the ICS and each of the two benchmark algorithms vary clearly depending on the problem difficulty. On two problems (pr01 and pr02), all three algorithms seem to generate always the same results, while for problems pr08 to pr10, with larger number of depots and periods, the ICS seems considerably to be more efficient to produce better results.

Table 5.6 represents the average results on the spacing metric obtained for each algorithm in 10 different runs.

Table 5.6: Comparison of spacing metric obtained for different algorithms

Instance	ICS	MHA	UTS
pr01	3.42	3.42	3.42
pr02	5.12	5.12	5.12
pr03	2.45	2.92	3.21
pr04	2.84	3.33	4.01
pr05	4.29	5.71	5.18
pr06	1.12	2.63	3.38
pr07	0.79	1.12	1.09
pr08	2.19	4.41	3.77
pr09	4.19	6.62	9.15
pr10	8.73	11.04	10.18

As shown in Table 5.6, the average values of the spacing metric generated by the ICS are lower than those obtained by the HMA and the UTS in the most problem instances. These results reveal that non-dominated solutions generated by the ICS are scattered more uniformly on the non-dominated solutions frontier, compared to the other considered algorithms. This fact shows the promising capability of the ICS to produce different non-dominated solutions from different zones of the frontier and to avoid being trapped on a limited number of local non-dominated solutions.

Table 5.7 reports the average values of the diversification metric obtained by each of the algorithms.

By studying Table 5.7, one can notice that the proposed ICS algorithm is considerably able to produce more diverse non-dominated solutions, specially for the large-sized problem instances. These results show that non-dominated solutions found by the ICS cover the optimal frontier with a higher diversification level.

Table 5.8 shows the average values of the quality metric produced by each of the algorithms.

The results of Table 5.8 clearly presents the superiority of the developed ICS algorithm, compared to the benchmark algorithms, to generate better non-dominated solutions. In other words, in the majority of problem instances, non-dominated solutions obtained by the ICS are

Table 5.7: Comparison of diversification metric obtained for different algorithms

Instance	ICS	MHA	UTS
pr01	6.21	6.21	6.21
pr02	7.75	7.75	7.75
pr03	5.91	5.45	5.08
pr04	5.44	5.29	5.19
pr05	9.31	8.23	7.92
pr06	6.16	5.52	5.29
pr07	3.18	3.04	3.15
pr08	7.49	4.14	4.92
pr09	5.51	3.02	2.88
pr10	5.01	2.29	2.87

Table 5.8: Comparison of quality of solutions obtained for different algorithms

Instance	ICS:MHA	ICS:UTS
pr01	1:1	1:1
pr02	1:1	1:1
pr03	5.60:4.40	5:75:4.25
pr04	6.62:3.38	6.25:3.75
pr05	6.04:3.96	6.24:4.76
pr06	6.35:3.65	6.88:3.12
pr07	5.89:4.11	5.62:4.38
pr08	6.37:3.63	6.88:3.12
pr09	7.25:2.75	7.44:2.54
pr10	7.07:2.93	7.14:2.86

nearer to the true Pareto frontier than those solutions produced by the HMA and the UTS. This fact reveals that the ICS gives a better image of the Pareto frontier, specially for those problem instances having larger number of customers, depots and periods.

Finally, Table 5.9 represents the average computational times consumed by each algorithm. These results illustrate that the average computational time of the ICS is short, barely higher than the other benchmark algorithms, and suitable for many operational decisions.

To sum up, we can deduce that the proposed cooperative parallel algorithm is a competitive and powerful solution methodology which is able to find diverse non-dominated solutions, truly reflecting the Pareto-optimal frontier, in a reasonable execution time.

Table 5.9: The average values of computational times (min.)

Instance	ICS	MHA	UTS
pr01	0.62	0.19	0.37
pr02	1.28	0.53	0.95
pr03	7.92	4.73	6.69
pr04	11.41	8.35	9.55
pr05	21.34	15.80	18.81
pr06	23.19	16.44	19.38
pr07	2.37	1.30	1.89
pr08	9.15	5.64	7.50
pr09	23.91	17.14	18.99
pr10	25.16	19.02	19.54

5.6 conclusions

This paper presented a new algorithmic framework, performing based on a cooperative parallel algorithm, namely integrated cooperative search, to efficiently solve a bi-criteria multi-depot periodic vehicle routing problem, for which no efficient algorithm is currently available. The developed algorithm was designed based on a new hierarchical decomposition procedure, the integration of elite partial solutions yielded by the sub-problems, and an adaptive guiding mechanism. To validate the performance of the proposed algorithm, in terms of solution quality and diversity level, various test problems were examined. Further, the efficiency of the algorithm, based on various useful metrics, was compared against two prominent algorithms existing in the literature. Our experimental results indicated that the designed algorithm outperforms the benchmark algorithms and improved the quality of the obtained solutions, especially for large-sized problems.

Acknowledgements

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair, Collaborative Research and Development, and Discovery Grant programs, by the Fonds de Recherche du Québec - Nature et Technologies (FRQ-NT) through its Team Research Project program, and by the

EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants.

Chapter 6

Conclusions

The VRP, as a well-known combinatorial NP-hard problem, has drawn huge interest from many researchers during the past decades because of the vital role it plays in the planning of distribution systems and logistics in many real-life applications such as garbage collection, mail delivery, and task sequencing. Due to the significant economic benefits that can be achieved by optimizing routing problems in practice, more and more attention has been given to various extensions of the VRP that arise in reality. These extensions are often called Multi-Attribute Vehicle Routing Problems (MAVRPs). Contrary to the research of the classical VRP that focuses on the idealized models with unrealistic assumptions, the research on MAVRPs considers those complicated attributes and constraints encountered in the real-life planning and provides solutions that are executable in practice.

In this thesis, we studied the models and solution methods of three practical MAVRPS. Each of them involves special practical issues that are only considered in very few papers. The models and methods proposed in the thesis are general and can be applied to practical routing problems arising in many other distribution companies as well. We first considered a Multi-Depot Vehicle Routing Problem (MDVRP) in which a fleet of homogeneous vehicles, departing from different depots, are responsible of visiting a set of geographically dispersed customers over a planning horizon. In this problem, two practical constraints, often found in reality, i.e., maximum route duration constraint and an upper limit of the quantity of goods that each vehicle can transport, are considered. The goal of the considered MDPVRP is to minimize the total distance traveled by all the vehicles over the planning horizon. We presented a mathematical model and proposed a Path Relinking (PR) heuristic to solve it. The proposed PR was developed based on purposeful exploration and exploitation strategies which permit the algorithm to solve the problem in two different ways: 1) As a pure stand alone algorithm, and 2) As an integrator in a parallel cooperative solution framework. To prove the ability of the PRA in generating high-quality and diverse solutions, various test problems, derived from the literature, were solved. The experimental results showed that the designed Path Relinking Algorithm performs impressively well, on all the problem instances. In the second part of this thesis, we proposed a new modular heuristic algorithm for addressing the MDPVRP studied in the first part and two of its special variants, i.e., Multi-Depot VRP (MDVRP) and Periodic VRP (PVRP). In each of the considered problems, the goal was to determine the optimal fleet size when three constraints, i.e., vehicle capacity, route duration and budget constraints, were to be satisfied. The proposed heuristic algorithm introduced several methodological contributions, particularly, a self-learning mechanism that leads the algorithm to assign better visit patterns to customers, and also to assign customers to better depots as the solution process evolves. This learning mechanism, in addition to other components of the algorithm, provided the capability of the heuristic algorithm to reach high quality solutions. To show that the MHA is a highly efficient solution method, in producing good solutions, several test problems were solved. The computational results revealed that the MHA performs considerably well, in terms of both solution quality and computational efficiency. Finally, in the last part, we addressed the bi-objective MDPVRP studied in the second part, including the same set of constraints and objective functions. To solve the problem, we designed a parallel cooperative solution methodology, performing based on the principles of the Integrative Cooperative Search (ICS) paradigm. The developed solution method was designed based on some well-structured searching mechanisms, particularly including a new hierarchical decomposition procedure, the integration of elite partial solutions yielded by the sub-problems, and an adaptive guiding mechanism. To prove the efficiency of the proposed ICS, in terms of solution quality and diversity level, different test problems were tested. Moreover, the performance of the solution method, based on several useful metrics, was compared to two state-of-the-art algorithms existing in the literature. Our experimental results showed the superiority of the designed ICS in generating good and diverse non-dominated solutions better reflecting the true Pareto-optimal frontier compared to the benchmark algorithms.

We conclude this chapter by highlighting some research perspectives. As this thesis illustrated, a number of heuristic methods are widely acknowledged for their performance on a variety of MAVRPs. Given how differently these methods define and explore the search space, they are very likely to lead to effective hybrid algorithms and parallel cooperative methods. It appears that most successful heuristics are not determined by a single factor but are the result of a good balance between several elements: the use of different search spaces, variable neighbourhoods, short-, medium- and long-term memories, trade-off between diversification and intensification, and cooperation. We believe that working on each of these elements in order to build an efficient heuristic is an extremely challenging research field, particularly given the trend toward problem settings including a continuously increasing number of attributes and solutions methods capable of simultaneously addressing these attributes.

Bibliography

J. Alegre, M. Laguna, and J. Pacheco. Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. *European Journal of Operational Research*, 179(3):736–746, 2007.

F. Alonso, M. J. Alvarez, and J. E. Beasley. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59:963–976, 2008.

E. Angelelli and M. G. Speranza. The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137(2):233–247, 2002.

C. Archetti, M. Grazia Speranza, and A. Hertz. A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem. *Transportation Science*, 40:64–73, 2006.

A. Bachem, W. Hochstättler, and M. Malich. The Simulated Trading Heuristic for Solving Vehicle Routing Problems. *Discrete Applied Mathematics*, 65:47–72, 1996.

B. M. Baker and M.A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, 2003.

M. L. Balinski and R. E. Quandt. On an Integer Program for a Delivery Problem. *Operations Research*, 12:300–304, 1964.

J. E. Beasley. Route first-Cluster second methods for vehicle routing. *Omega*, 11:403–408, 1983.

L. Bertazzi, G. Paletta, and M. G. Speranza. An improved heuristic for the period traveling salesman problem. *Computers & Operations Research*, 31:1215–1222, 2004.

D. J. Bertsimas and D. Simchi-Levi. A New Generation of Vehicle Routing Research: Robust Algorithms, Addressing Uncertainty. *Operations Research*, 44:286–304, 1996.

J. F. Bérubé, M. Gendreau, and J.Y. Potvin. An exact epsilon-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. *European Journal of Operational Research*, 194:39–50, 2009.

A. Le Bouthillier, T.G. Crainic, and P. Kropf. A Guided Cooperative Search for the Vehicle Routing Problem with Time Windows. *IEEE Expert / IEEE Intelligent Systems*, 20:36–42, 2005.

J. Bramel and D. Simchi-levi. A Location Based Heuristic for General Routing Problems. *Operations Research*, 43:649–1660, 1995.

P. J. Cassidy and H. S. Bennett. TRAMPâ A Multi-depot Vehicle Scheduling System. *Journal of the Operational Research Society*, 23:151–163, 1972.

I. M. Chao, B. L. Golden, and E. Wasil. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences*, 13(3-4):371–406, 1993.

E. Choi and D. Tcha. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34:2080–2095, 2007.

N. Christofides and J. E. Beasley. The period routing problem. *Networks*, 14(2):237–256, 1984.

N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In *Combinatorial Optimization*, pages 315–338, 1979.

P. C. Chu and J. E. Beasley. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24:17–23, 1997.

G. Clarke and J. W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12:568–581, 1964.

J.F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30:105–119, 1997.

S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil. Using Experimental Design to Find Effective Parameter Settings for Heuristics. *Journal of Heuristics*, 7:77–97, 2000.

T. G. Crainic. Parallel solution methods for vehicle routing problems. *CIRRELT*, pages 1–27, 2007.

T. G. Crainic, G. C.Crisan, M. Gendreau, N. Lahrichi, and W. Rei. A concurrent evolutionary approach for rich combinatorial optimization. In *Genetic and Evolutionary Computation Conference*, pages 2017–2022, 2009.

G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6: 80–91, 1959.

R. Dondo and J. Cerdà. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176:1478–1507, 2007.

M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5:137–172, 1999.

M. Dror and P. Trudeau. Savings by Split Delivery Routing. *Transportation Science*, 23: 141–145, 1989.

L. A. Drummond, L. S. Ochi, and D. S. Vianna. An asynchronous parallel metaheuristic for the period vehicle routing problem. *Future Generation Computer Systems*, 17:379–386, 2001.

G. Dueck. New Optimization Heuristics The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computational Physics*, 104:86–92, 1993.

M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for the vehicle routing problem. In *Conference on Computer Networks*, 1981.

B. A. Foster and D. M. Ryan. An Integer Programming Approach to the Vehicle Scheduling Problem. In *Operations Research*, 1976.

P. Frizzell and J. Giffin. The bounded split delivery vehicle routing problem with grid network distances. *Asia-Pacific Journal of Operational Research*, 9:101–116, 1992.

M. J. Geiger. Genetic Algorithms for multiple objective vehicle routing. *Computing Research Repository*, 2008.

M. Gendreau, A. Hertz, and G. Laporte. New Insertion and Postoptimization Procedures for the Traveling Salesman Problem. *Operations Research*, 40:1086–1094, 1992.

M. Gendreau, A. Hertz, and G. Laporte. A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, 40:1276–1290, 1994.

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

I. Ghamlouche, T. G. Crainic, and M. Gendreau. Path Relinking, Cycle-Based Neighbourhoods and Capacitated Multicommodity Network Design. *Annals of Operations Research*, 131: 109–133, 2004.

F. Gheysens, B. L. Golden, and A. A. Assad. A new heuristic for determining fleet size and composition. In *Mathematical Programming Study*, 1986.

B. E. Gillett and L. R. Miller. A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations Research*, 22:340–349, 1974.

B.E. Gillett and J.G. Johnson. Multi-terminal vehicle-dispatch algorithm. *Omega*, 4(6):711–718, 1976.

F. Glover. New ejection chain and alternating path methods for traveling salesman problems. *Computers & Operations Research*, pages 449–509, 1992.

F. Glover and M. Laguna. Tabu Search. *Ibm Journal of Research and Development*, 7, 2003.

F. Glover, M. Laguna, and R. Martí. Fundamentals of Scatter Search and Path Relinking. *Control and Cybernetics*, 39:653–684, 2000.

B. L. Golden, T. L. Magnanti, and H. Q. Nguyen. Implementing vehicle routing algorithms. *Networks*, 7:113–148, 1977.

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

B.L. Golden, E.A. Wasil, J.P. Kelly, and I.M. Chao. The impact of metaheuristics on solving the vehicle routing problems: algorithms, problem sets, and computational results. In *Fleet Management and Logistics, T. G. Crainic and G. Laporte, eds*, pages 33–56. Kluwer, Boston, 1998.

D. Gulczynski, B.L. Golden, and E.A. Wasil. The period vehicle routing problem: New heuristics and real-world variants. *Transportation Research Part E-logistics and Transportation Review*, 47:648–668, 2011.

E. Hadjiconstantinou and R. Baldacci. A multi-depot period vehicle routing problem arising in the utilities sector. *Journal of The Operational Research Society*, 49:1239–1248, 1998.

M. Haimovich and A. H. G. Rinnooy Kan. Bounds and Heuristics for Capacitated Routing Problems. *Mathematics of Operations Research*, 10:527–542, 1985.

V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195:791–802, 2009.

S. C. Ho and D. Haugland. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers & Operations Research*, 31:1947–1964, 2004.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

S.C. Hong and Y.B. Park. A heuristic for bi-objective vehicle routing with time window constraints. *International Journal of Production Economics*, 62:249–258, 1999.

A. Imran, S. Salhi, and N. A. Wassan. A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, 197: 509–518, 2009.

N. Jozefowiez, F. Semet, and E.G. Talbi. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189:293–309, 2008.

K.H. Kang, Y.H. Lee, and B.K. Lee. An Exact Algorithm for Multi Depot and Multi Period Vehicle Scheduling Problem. In *Computational Science and Its Applications*, pages 350–359, 2005.

J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34:2743–2757, 2007.

N. Lahrichi, T.G. Crainic, M. Gendreau, W. Rei, and L.M. Rousseau. Solving the dairy problem for the province of Quebec. *CIRRELT*, 34, 2011.

N. Lahrichi, T. G. Crainic, M. Gendreau, W. Rei, G. C. Crisan, and T. Vidal. An integrative cooperative search framework for multi-decision-attribute combinatorial optimization. *CIRRELT*, 42, 2012.

H.C.W Lau, T.M. Chan, W.T. Tsui, and W.K. Pang. Application of Genetic Algorithms to Solve the Multidepot Vehicle Routing Problem. *IEEE Transactions on Automation Science and Engineering*, 7(2):383–392, 2010.

M. Laumanns, L. Thiele, and E. Zitzler. An Adaptive Scheme to Generate the Pareto Front Based on the Epsilon-Constraint Method. In *Dagstuhl Seminars*, 2005.

T. S. Lin. Computer solutions of the travelling salesman problem. *Bell System Technology Journal*, 44:2245–2269, 1965.

S. Liu, W. Huang, and H. Ma. An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E-logistics and Transportation Review*, 45: 434–445, 2009.

A. Mingozzi. The Multi-depot Periodic Vehicle Routing Problem. In *Symposium on Abstraction, Reformulation and Approximation*, pages 347–350, 2005.

R.H. Mole and S.R. Jameson. A sequential route-building algorithm employing a generalized savings criterion. *Operational Research Quarterly*, 27:503–511, 1976.

M. Mourgaya and F. Vanderbeck. Problème de tournées de véhicules multipériodiques: Classification et heuristique pour la planification tactique. *RAIRO Operations Research*, 40: 169–194, 2006.

P. A. Mullaseril, M. Dror, and J. Leung. Split-delivery routing in livestock feed distribution. *Journal of The Operational Research Society*, 48:107–116, 1997.

T. Murata and R. Itai. Multi-objective vehicle routing problems using twofold emo algorithms to enhance solution similarity on non-dominated solutions. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, pages 885–896, 2006.

T. Murata and R. Itai. Local Search in Two-Fold EMO Algorithm to Enhance Solution Similarity for Multi-objective Vehicle Routing Problems. In *Evolutionary Multi-Criterion Optimization*, pages 201–215, 2007.

B. M. Ombuki, B. J. Ross, and F. Hanshar. Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows. *Applied Intelligence*, 24:17–30, 2006.

I. Or. Traveling salesman-type combinatorial problems and their relations to the logistics of blood banking. *PhD thesis, Department of Industrial Engineering and Management Science, North-western University*, 1976.

I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.

D. C. Paraskevopoulos, P. P. Repoussis, C. D. Tarantilis, G. Ioannou, and G. P. Prastacos. A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics*, 14:425–455, 2008.

Y. Park and C. Koelling. A solution of vehicle routing problems in a multiple objective environment. *Engineering Costs and Production Economics*, 10:121–132, 1986.

Y. Park and C. Koelling. An interactive computerized algorithm for multi-criteria vehicle routing problems. *Computers and Industrial Engineering*, 16:477–490, 1989.

P. Parthanadee and R. Logendran. Periodic product distribution from multi-depots under limited supplies. *IIE Transactions*, 38:1009–1026, 2006.

S. Pirkwieser and G. R. Raidl. Multilevel Variable Neighborhood Search for Periodic Routing Problems. In *EvoWorkshops*, pages 226–238, 2010.

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

O. M. Raft. A modular algorithm for an extended vehicle scheduling problem. *European Journal of Operational Research*, 11:67–76, 1982.

A. Rahimi-Vahed, T.G. Crainic, M. Gendreau, and W. Rei. A path relinking algorithm for a multi-depot periodic vehicle routing problem. *CIRRELT*, 2012a.

A. Rahimi-Vahed, T.G. Crainic, M. Gendreau, and W. Rei. Fleet-sizing for multi-depot and periodic vehicle routing problems using a modular heuristic algorithm. *CIRRELT*, 2012b.

C. Rego and C. Roucairol. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In *Meta-Heuristics: Theory and Applications*, 1996.

M. Reimann, M. Stummer, and K. Doerner. A Savings Based Ant System For The Vehicle Routing Problem. In *Genetic and Evolutionary Computation Conference*, pages 1317–1326, 2002.

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

J. Renaud and F. F. Boctor. A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, 140:618–628, 2002.

J. Renaud, F. F. Boctor, and G. Laporte. A Fast Composite Heuristic for the Symmetric Traveling Salesman Problem. *Informs Journal on Computing*, 8:134–143, 1996a.

J. Renaud, F. F. Boctor, and G. Laporte. An improved petal heuristic for the vehicle routing problem. *Journal of the Operational Research Society*, 47:329–336, 1996b.

J. Renaud, G. Laporte, and F. F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23:229–235, 1996c.

R. Ribeiro and H. R. Lourenço. A Multi-Objective Model For A Multi-Period Distribution Management Problem. *Ssrn Electronic Journal*, 2001.

J. Rieck and J. Zimmermann. A Sampling Procedure for Real-Life Rich Vehicle Routing Problems. In *Operations Research*, pages 355–360, 2006.

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

R. Russell and D. Gribbin. A multi-phase approach to the period routing problem. *Networks*, 21:747–765, 1991.

D. M. Ryan, C. Hjorring, and F. Glover. Extensions of the petal method for vehicle routing. *Journal of Operational Research Society*, 44:289–296, 1993.

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

S. Salhi and M. Sari. A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. *European Journal of Operational Research*, 103:95–112, 1997.

S. K. Smith and A. E. Eiben. Parameter Tuning of Evolutionary Algorithms: Generalist vs. Specialist. In *EvoWorkshops*, pages 542–551, 2010.

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

C. Tan and J. Beasley. A heuristic algorithm for the period vehicle routing problem. *Omega-international Journal of Management Science*, 12:497–504, 1984.

K. C. Tan, Y. H. Chew, and L. H. Lee. A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows. *Computational Optimization and Applications*, 34:115–151, 2006.

C. D. Tarantilis and C. T. Kiranoudis. BoneRoute: An Adaptive Memory-Based Method for Effective Fleet Management. *Annals of Operations Research*, 115:227–241, 2002.

P. M. Thompson and H. N. Psaraftis. Cyclic transfer algorithms for multi-vehicle routing and scheduling problems. In *Operations Research*, 1993.

F. A. Tillman. The Multiple Terminal Delivery Problem with Probabilistic Demands. *Transportation Science*, 3:192–204, 1969.

F. A. Tillman and T.M. Cain. An upper bound algorithm for the single and multiple terminal delivery problem. *Management Science*, 18:664–682, 1972.

F. A. Tillman and R.W. Hering. A study of a look-ahead procedure for solving the multiterminal delivery problem. *Transportation Research*, 5:225–229, 1971.

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

P. Toth and D. Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *Informs Journal on Computing*, 15:333–346, 2003.

T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A Hybrid Genetic Algorithm for Multi-Depots and Periodic Vehicle Routing Problems. *Operations Research*, 60:611–624, 2012a.

T. Vidal, T.G. Crainic, M. Gendreau, and C. Prins. Heuristics for Multi-Attribute Vehicle Routing Problems: A Survey and Synthesis. *CIRRELT*, 5, 2012b.

N. A. Wassan and I. H. Osman. Tabu search variants for the mix fleet vehicle routing problem. *Journal of The Operational Research Society*, 53:768–782, 2002.

J. A. G. Willard. Vehicle routing using r-optimal tabu search. Master's thesis, The Management School, Imperial College, London, 1989.

J. Xu and J. P. Kelly. A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. *Transportation Science*, 30:379–393, 1996.

B. Yu, Z.Z. Yang, and J.X. Xie. A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society*, 62:183–188, 2011.