UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

A cartographic approach to the dynamic vehicle routing problem with time windows and stochastic customers

Daniel Bustos Coral

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Daniel Bustos Coral

A cartographic approach to the dynamic vehicle routing problem with time windows and stochastic customers

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Maristela Oliveira dos Santos

USP – São Carlos
August 2018

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Bustos, Daniel
B982c A cartographic approach to the dynamic vehicle routing problem with time windows and stochastic customers / Daniel Bustos; orientadora Maristela Oliveira dos Santos. -- São Carlos, 2018.
112 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2018.

1. vehicle routing problem. 2. optimization. 3. heuristics. 4. logistics. I. Santos, Maristela Oliveira dos, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Daniel Bustos Coral

Uma abordagem cartográfica ao problema de roteamento dinâmico de veículos com janelas de tempo e clientes estocásticos

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Maristela Oliveira dos Santos

USP – São Carlos
Agosto de 2018

I dedicate this work to my parents.

ACKNOWLEDGEMENTS

The completion of this undertaking could not have been possible without the support of Dr. Maristela Santos. I gratefully acknowledge her kindness and constant help.

I would also like to thank Drs Claudio Toledo and Augusto Hauber Gameiro for their support along the way.

Finally, I would like to thank CAPES (PROEX-9739362/M) for the financial support received.

“While we wait for life, life passes.”

Seneca

RESUMO

BUSTOS, D. **Uma abordagem cartográfica ao problema de roteamento dinâmico de veículos com janelas de tempo e clientes estocásticos**. 2018. 112 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Esta dissertação apresenta uma abordagem cartográfica para o problema de roteamento de veículos dinâmico com janelas de tempo e clientes estocásticos (DVRPTWSC, por sua sigla em inglês). Os objetivos considerados são minimizar o tempo total de viagem e maximizar o número de pedidos novos atendidos. Para abordar o DVRPTWSC é necessário resolver o problema de roteamento de veículos com janelas de tempo (VRPTW, por sua sigla em inglês). Assim, para tratar o VRPTW propõe-se um algoritmo memético (MA, por sua sigla em inglês). O MA reduz o espaço de busca usando informação obtida por meio de um procedimento de clusterização, o qual é aplicado aos dados espaciais dos clientes. Para o DVRPTWSC, a abordagem cartográfica é incorporada em um sistema multiagente, no qual um agente roteirizador planeja as rotas para os agentes veículos. O processamento cartográfico é aplicado antes de criar o plano de rotas inicial para o DVRPTWSC. Este procedimento usa clusterização hierárquica para dividir a região onde estão os clientes em uma hierarquia de regiões encaixadas. O plano de rotas inicial considera pedidos conhecidos e pedidos potenciais amostrados de distribuições de probabilidade conhecidas. Para obter o plano de rotas inicial, usam-se os operadores de busca do MA, os quais utilizam a informação obtida da clusterização hierárquica para fazer a busca. Ao longo do horizonte de planejamento, o roteirizador atualiza o plano de rotas: Pedidos potenciais que foram considerados no plano de rotas inicial e que não foram consolidados são removidos e novos pedidos são incluídos usando o procedimento *assignation of requests based on nested regions* (ARNR). O procedimento ARNR visa reduzir o número de veículos considerados para atender novos pedidos. Para isso, tenta designar os novos pedidos aos veículos disponíveis para o atendimento que possuem os menores custos de desvio da rota pré-determinada. As regiões encaixadas criadas no processamento cartográfico são utilizadas para identificar esses veículos. Para o VRPTW, resultados experimentais mostram que o MA proposto é competitivo com métodos do estado da arte. A abordagem proposta para o DVRPTWSC supera abordagens que não incluem pedidos potenciais no plano de rotas inicial. O uso do procedimento ARNR reduz significativamente o número de veículos considerados para atender novos pedidos, e produz soluções similares às produzidas quando se consideram todos os veículos em operação. A abordagem desenvolvida para o DVRPTWSC tem um desempenho consistente para três níveis de dinamismo: baixo, médio e alto.

Palavras-chave: DVRPTWSC, VRPTW, algoritmo memético, sistema multiagente, abordagem cartográfica.

ABSTRACT

BUSTOS, D. **A cartographic approach to the dynamic vehicle routing problem with time windows and stochastic customers**. 2018. 112 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

This dissertation presents a cartographic approach to the dynamic vehicle routing problem with time windows and stochastic customers (DVRPTWSC). The objectives are to minimize the total travel time and maximize the number of new requests served. Addressing the DVRPTWSC requires solving the vehicle routing problem with time windows (VRPTW). A memetic algorithm (MA) for the VRPTW is proposed. The MA prunes the search space using the information gathered by a clustering procedure, which is applied to customers' spatial data. The cartographic approach to the DVRPTWSC is incorporated into a multiagent system where a dispatcher agent plans the routes for vehicle agents. Before creating the initial routing plan, a cartographic processing is applied. This procedure uses hierarchical clustering to divide the region where customers are located into a hierarchy of nested regions. The initial routing plan considers known requests and potential requests sampled from known probability distributions. It is created using the search operators of the MA, which in turn use the information obtained from the hierarchical clustering to perform the search. Over the planning horizon, the dispatcher updates the routing plan: Potential requests that were included in the initial routing plan and do not materialize are removed and new requests are processed using the assignation of requests based on nested regions (ARNR). The ARNR procedure is aimed at reducing the number of vehicles considered for serving new requests. It tries to assign the requests among the vehicles that can serve them at low detour costs. The nested regions created in the cartographic processing are used to identify such vehicles. Experimental results show that the proposed MA performs competitively with state-of-the-art heuristics for the VRPTW. The proposed approach to the DVRPTWSC outperforms approaches that do not include potential requests in the initial routing plan. The use of the ARNR procedure significantly reduces the number of vehicles considered for serving new requests, and it yields solutions similar to those obtained when considering all vehicles in operation. The proposed approach performs consistently under three levels of dynamism: low, medium, and high.

Keywords: DVRPTWSC, VRPTW, memetic algorithm, multiagent system, cartographic approach.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AHGA	adaptive hybrid genetic algorithm
AL	Alvarenga, Mateus and Tomi (2007)
ARNR	assignation of requests based on nested regions
BCRC	best cost route crossover
BPC	branch-price-and-cut
BRH	branch-and-regret heuristic
CB-BCRC	clustering-based best cost route crossover
CB-Inter-Route-NS	clustering-based inter-route neighborhood search
CGA	co-evolutionary genetic algorithm
CGH	column generation heuristic
CH	Chiang and Russell (1997)
CNV	cumulative number of vehicles
CO	Cordeau, Laporte and Mercier (2001)
CTC	cumulative total travel distance (or travel time)
DSHH	dynamic stochastic hedging heuristic
DVRPTWSC	dynamic vehicle routing problem with time windows and stochastic customers
EAX	edge assembly crossover
FDR	false discovery rate
GA	genetic algorithm
GIS	geographic information system
GN	Garcia-Najera and Bullinaria (2011)
GPS	global positioning system
HAC	hierarchical agglomerative clustering
HC	hill-climbing
JU	Jung and Moon (2002)
Kw-AllV	approach to the DVRPTWSC that does not include potential requests in the initial routing plan and considers all active vehicles as candidates to serve new requests
Kw-ARNR	approach to the DVRPTWSC that does not include potential requests in the initial routing plan and uses the ARNR procedure
LOF	localized optimization framework
MA	memetic algorithm
MDH	myopic dynamic heuristic

MIT	Massachusetts Institute of Technology
MS-ILS	multistart iterated local search
MS-LI	multistart local improvement
MSA	multi scenario approach
NN	nearest neighbor
NV	number of vehicles
OL	Oliveira and Vasconcelos (2010)
OX	ordered crossover
RO	Rochat and Taillard (1995)
SA	simulated annealing
Sc-AllV	approach to the DVRPTWSC that includes potential requests in the initial routing plan and considers all active vehicles as candidates to serve new requests
Sc-ARNR	approach to the DVRPTWSC that includes potential requests in the initial routing plan and uses the ARNR procedure
SPP	set partitioning problem
TA	Tan *et al.* (2001)
TC	travel cost in terms of distance or time
TT	travel time
UPGMA	unweighted pair group method with arithmetic mean
UR	Ursani *et al.* (2011)
VI	Vidal *et al.* (2015)
VNS	variable neighborhood search
VRP	vehicle routing problem
VRPTW	vehicle routing problem with time windows

CONTENTS

CHAPTER

1

INTRODUCTION

The vehicle routing problem (VRP) consists of determining an optimal routing plan to provide transportation services to a set of customers. Since its introduction by Dantzig and Ramser (1959), the VRP has come to be one of the most important problems in transportation logistics. Commonly, VRPs have been modeled and solved assuming that all the information about the problem is deterministic (i.e., it is known with certainty) and static (i.e., it does not change over time). Nevertheless, in real-life VRPs decisions must be taken in dynamic and uncertain environments (BEKTAş; REPOUSSIS; TARANTILIS, 2014). Travel times can vary due to traffic, customers' demands can be known only partially, new requests can arrive while vehicles are in operation, and vehicles can even experience breakdowns. Thus, practical approaches to VRPs must avoid introducing unrealistic assumptions on the uncertainty and dynamism of the environment.

Dynamic VRPs have been conceived in an effort to better represent real-life transportation problems. In dynamic VRPs, routing plans can be changed as they are executed, in response to changes in the environment. Instead of an optimal routing plan, the solution of a dynamic VRP defines a set of strategies to better anticipate and react to dynamic events. Given that routing plans are subject to changes in dynamic VRPs, the study of these problems requires considering communication and coordination mechanisms between the actors involved. Indeed, most of the literature on dynamic VRPs has been published since the late 1990s, simultaneously with the development and adoption of technologies that are critical to implement the solutions for these problems, such as mobile computing, Internet, cellphone networks, global positioning systems (GPS), and geographical information systems (GIS) (PSARAFTIS; WEN; KONTOVAS, 2016).

The dynamism of real-life VRPs can be tackled by using static routing approaches: An initial routing plan is generated considering the known information and whenever the information of the problem changes, the routing plan is updated by performing a replanning from scratch with the updated information. This approach works well on VRPs with low levels of dynamism (BRÄYSY; HASLE, 2014). However, many important VRP applications are highly dynamic

and/or uncertain and require more suitable approaches. That is the case of VRPs arising in modern last-mile logistics, where e-commerce poses new challenges for logistic carriers. In these problems, dynamic routing is crucial: Orders that arrive in the morning must be processed by noon, vehicles might need to divert their routes due to traffic jams and they may also need to perform unplanned stops to pick up returned packages. The routes are planned using historical data of traffic speeds and GPS-based solutions help to track the status of the vehicles and even provide dynamic estimated times of delivery to customers (BANKER, 2018). Food delivery has been experiencing a boost with the consolidation of on-demand meal-ordering platforms like Grubhub, Postmates, and UberEats (ISAAC, 2017). These platforms need to cope with complex routing problems, which are characterized by high levels of dynamism and urgency. In view of problems like those mentioned above, large-scale highly dynamic delivery systems will be the norm in the future (SAVELSBERGH; WOENSEL, 2016).

Recognizing the need for new approaches to handle dynamism in VRPs and considering that time-window constraints are common in real-life VRPs, this study presents a cartographic approach to the dynamic vehicle routing problem with time windows and stochastic customers (DVRPTWSC).

The problem considered consists of defining a decision-making process for a dispatcher to assign customer requests to a homogeneous fleet of vehicles with limited capacities. Vehicles depart and return to a single depot. Some requests are known at the beginning of the planning horizon and other requests arrive over the planning horizon, while vehicles are traversing their routes. Customers can make up to one request over the planning horizon. The probabilities of receiving new requests are known. All requests known at the beginning of the planning horizon must be served and the service of new requests is subject to operational capacity. The requests must be served within a given time window and all vehicles must arrive at the depot by the end of the planning horizon. Furthermore, vehicle capacities cannot be exceeded. The objectives are to minimize the total travel time and maximize the number of new requests served.

To address the DVRPTWSC, it is necessary to solve the vehicle routing problem with time windows (VRPTW). A simple yet effective memetic algorithm (MA) for the VRPTW is proposed. The main search operators of the MA are the clustering-based best cost route crossover (CB-BCRC) and the clustering-based inter-route neighborhood search (CB-Inter-Route-NS). Both operators create new solutions by relocating customers between routes passing close to each other. To identify such routes, these operators use the information gathered by a clustering procedure that is applied to customers' spatial data.

The cartographic approach to the DVRPTWSC is incorporated into a multiagent system, where a dispatcher receives customer requests and plans the routes for several vehicles. Prior to the start of the planning horizon, the dispatcher performs a cartographic processing of customers' spatial data. The cartographic processing divides the region where customers are located into a hierarchy of nested regions that resembles the hierarchy of administrative divisions

of geographical areas (e.g., counties, states, and countries). First, a region is created for each customer and then, hierarchical clustering is used to merge the regions associated to close customers, thus creating larger regions. Skupin (2002) described a similar process to create scale-dependent visualizations of non-geographic information. The output of the hierarchical clustering procedure is saved, and each customer is associated with a hierarchy of nested regions that contains the regions where the customer is situated.

The creation of the initial routing plan involves the optimization of a pool of routing scenarios, i.e., solutions to VRPTW instances containing all known requests and potential requests sampled from known probability distributions. The routing scenarios are initially solved by a construction heuristic and then they are reoptimized by the search operators of the MA, which use the output of the hierarchical clustering procedure to perform the search. When the reoptimization process finishes, the routing scenario most similar to the other routing scenarios in the pool is selected as the initial routing plan. By anticipating likely requests, the initial routing plan is expected to be more robust, such that it requires fewer modifications over the planning horizon. Approaches to dynamic VRPs that exploit stochastic information of the problem to anticipate future events have been shown to perform better than approaches ignoring any stochastic information, see Bent and Van Hentenryck (2004c), Hvattum, Løkketangen and Laporte (2006), and Barkaoui (2017) for some examples.

Over the planning horizon, the dispatcher updates the routing plan as new information is revealed. Potential requests that were included in the routing plan and do not materialize are removed and new requests are included in the routing plan if they can be feasibly served. The assignation of requests based on nested regions (ARNR) is used to process new requests. The aim of the ARNR procedure is to identify the vehicles yielding the lowest detour costs to serve new requests without having to consider all active vehicles, because calculating the detour cost that a vehicle would incur to serve a request can be a computationally expensive operation. The ARNR procedure scans the hierarchy of nested regions associated with the customer requiring service, looking for vehicles that currently are or will be close to the customer and can feasibly serve the request. It is expected that those vehicles can serve the request at lower detour costs than vehicles located farther from the customer.

Computational experiments were carried out to assess the performance of the proposed solution methodology. Regarding the MA, the results show that (1) the proposed MA performs competitively with state-of-the-art heuristics for the VRPTW; (2) the CB-BCRC operator can produce solutions identical to those produced by a similar crossover operator based on exhaustive search, while requiring shorter execution times; and (3) the CB-BCRC operator is the main feature responsible for the quality of the solutions found by the MA. Regarding the cartographic approach to the DVRPTWSC, the results show that (1) the quality of the routing scenarios, which initially are solved by using a construction heuristic, is significantly improved by the reoptimization process; (2) the solutions are significantly improved by exploiting stochastic

information on potential requests; (3) the use of the ARNR procedure significantly reduces the number of vehicles considered for serving new requests and it leads to solutions similar to those produced by considering all vehicles in operation; and (4) the proposed approach performs similarly under three different levels of dynamism: low, medium, and high.

The rest of this dissertation is organized as follows: Chapter 2 presents the literature review. Chapter 3 describes the MA for the VRPTW. Chapter 4 describes the cartographic approach to the DVRPTWSC. Chapter 5 presents concluding remarks and directions for further research.

CHAPTER

2

LITERATURE REVIEW

Case studies involving vehicle routing problems (VRPs) with time-window constraints were first considered by Pullen and Webb (1967), Knight and Hofer (1968), and Cook and Russell (1978). The study of the vehicle routing problem with time windows (VRPTW) has been largely influenced by the work of Solomon (1987), who proposed a set of benchmark instances that became the standard for assessing the performance of optimization algorithms. Currently, the VRPTW is one of the most studied NP-hard problems in transportation and logistics, due to its practical applications and its difficulty (VIDAL *et al.*, 2015).

The study of dynamic VRPs can be traced back to an MIT technical report by Wilson and Colvin (1977), who described a dial-a-ride system. The research on dynamic VRPs intensified in late 1990s, largely due to the advances in information and communication technologies as mobile computing, Internet, cellphone networks, geographic information systems, and global positioning systems, which are critical for this class of problems (PSARAFTIS; WEN; KONTOVAS, 2016).

There is a vast literature on vehicle routing with time windows and dynamic vehicle routing. For comprehensive surveys on the VRPTW, the reader is referred to Bräysy and Gendreau (2005a), Bräysy and Gendreau (2005b), and Desaulniers, Madsen and Ropke (2014). Comprehensive surveys on dynamic VRPs can be found in Pillac *et al.* (2013), Bektaş, Repoussis and Tarantilis (2014), Ritzinger, Puchinger and Hartl (2016), and Psaraftis, Wen and Kontovas (2016). This chapter reviews some important contributions from the literature on VRPTW and dynamic VRPs.

2.1 The Vehicle Routing Problem with Time Windows

The classical VRPTW aims to define a routing plan for serving a set of geographically scattered customers requiring goods to be either delivered or picked up. A fleet of vehicles is available for serving the customers. The routes should be planned ensuring that vehicle capacities

are not exceeded and all customers are visited within their period of availability, called a *time window*. As pointed out by Alvarenga, Mateus and Tomi (2007), the selection of the problem's objective depends largely on the characteristics of the situation being considered. In some cases, the minimization of the total distance traveled by the vehicles is more suitable, for example when the quantity of products to be transported is smaller than the total load capacity of the fleet. If the load capacity of the fleet is just barely big enough to satisfy the customers' demand, the minimization of the number of vehicles would be the main objective. Many real-life problems can be modeled as VRPTWs, including postal deliveries (MECHTI *et al.*, 1999), industrial gas distribution (CHIANG; RUSSELL, 2004), waste collection (KIM; KIM; SAHOO, 2006), food distribution (OSVALD; STIRN, 2008), and distribution in the retail industry (BELFIORE; YOSHIZAKI; TSUGUNOBU, 2009).

2.1.1 Solution methods

Exact approaches have commonly adopted the objective of minimizing the total travel distance. Desaulniers, Madsen and Ropke (2014) pointed out that the performance of exact algorithms for the VRPTW is largely influenced by the instance's structure. Instances with wide time windows are more difficult to solve for exact methods since the number of feasible routes increases jointly with the number of customers per feasible route, making finding the optimal solution more difficult. Instances with randomly-located customers are harder to solve that instances with clustered customers. The state-of-the-art exact method for the VRPTW is the branch-price-and-cut (BPC) algorithm of Pecin *et al.* (2017). This algorithm solved all 56 Solomon (1987) instances with 100 customers and 51 out of 60 Gehring and Homberger (1999) instances with 200 customers.

Traditionally, heuristic methods for the VRPTW have adopted a hierarchical objective, consisting of minimizing the number of vehicles used as their primary objective, and minimizing the total travel distance as their secondary objective (DESAULNIERS; MADSEN; ROPKE, 2014). Some heuristics that seek to minimize the total travel distance as their primary objective have been proposed by Jung and Moon (2002), Alvarenga, Mateus and Tomi (2007), Oliveira and Vasconcelos (2010), Ursani *et al.* (2011), and Vidal *et al.* (2015). Multiobjective heuristics for the VRPTW have been proposed by Ombuki, Ross and Hanshar (2006) and Garcia-Najera and Bullinaria (2011).

Heuristic methods, although without ensuring optimality, can often find good-quality solutions requiring relatively low computation times. Several of the state-of-the-art heuristics for the VRPTW are memetic algorithms (MAs). In general terms, MAs are evolutionary algorithms that incorporate local search procedures to refine the quality of the solutions (MOSCATO, 1989). Regarding the traditional hierarchical objective, the MAs proposed by Nagata, Bräysy and Dullaert (2010), Blocho and Czech (2013), Vidal *et al.* (2013), and Nalepa and Blocho (2015) found the current best known solutions for several of the benchmark instances of Solomon (1987)

(100 customers) and Gehring and Homberger (1999) (200, 400, 600, 800, and 1000 customers). Regarding the distance minimization objective, the heuristics proposed by Vidal *et al.* (2015) found the current best known solutions for the Solomon's instances.

The MA of Nagata, Bräysy and Dullaert (2010) uses a version of the edge assembly crossover (EAX) adapted to the VRPTW. An efficient procedure restores the feasibility of the solutions produced by the crossover operator. Feasible solutions generated by the repair procedure are further refined by a local search procedure.

The algorithm of Blocho and Czech (2013) executes parallel processes. A process runs either a heuristic algorithm or an MA and some local search procedures. The processes cooperate periodically using a randomized scheme. A process cooperates with others by sending them the best solution it has found so far. If a process receives a better solution during a cooperation phase, the received solution replaces its current solution. Otherwise, the EAX crossover operator creates a new solution from the current and received solutions. The new solution is repaired if necessary. If the new solution is feasible, it replaces the current solution.

The MA of Vidal *et al.* (2013) maintains feasible and infeasible solutions in two separate subpopulations. The ordered crossover (OX) is used to create new solutions and two local search operators are applied to improve them. With a certain probability, a repair procedure is applied to infeasible solutions. A diversification mechanism is executed after a certain number of successive iterations without improvement.

Nalepa and Blocho (2015) described a parallel MA that evolves several populations in parallel processes. The EAX crossover operator is used to create new solutions. The repair, local search, and mutation procedures are based on a hill-climbing approach that uses traditional inter-route and intra-route neighborhoods. The former consist of interchanging customers between different routes and the latter consist of relocating customers within a single route. The processes cooperate and exchange solutions according to a cooperation scheme. The authors investigated the impact of different cooperation schemes on the performance of the MA.

A part of this study is devoted to the development of an MA that addresses the VRPTW aiming to minimize the total travel time. The results of the proposed MA were compared with those of heuristics that address the VRPTW seeking to minimize the total travel distance or time (travel times are equal to their corresponding distances in the considered benchmark instances). The heuristics considered for comparison were those of Jung and Moon (2002), Alvarenga, Mateus and Tomi (2007), Oliveira and Vasconcelos (2010), Ursani *et al.* (2011) and Vidal *et al.* (2015).

Jung and Moon (2002) proposed an MA that uses 2D images to represent the solutions. The crossover operator cuts the 2D space where customers are located with different curves and figures. The route segments located in the areas created with the cuts are combined. A repair procedure adds possible missing route segments. The mutation procedure splits each route of a

solution into at most three routes. Three local search procedures are used to refine the solutions.

Alvarenga, Mateus and Tomi (2007) proposed a column generation heuristic method (CGH). The VRPTW is formulated as a set partitioning problem (SPP), where each column in the SPP model represents a route. A genetic algorithm (GA) is used to generate routes that are inserted into the SPP model. CGH consists of two phases that are executed iteratively. In the first phase, the GA is used to generate routes for the original problem and the SPP model is solved with the columns generated. In the second phase, the solution created in the previous phase is divided into several subsets of routes. The GA is applied to each subset of routes to generate more routes. The routes created during phases one and two are saved and the procedure continues until the execution time limit is reached. At the end of the search process, the final solution is obtained by solving the SPP model with all the routes generated during the search.

Oliveira and Vasconcelos (2010) presented a hybrid heuristic that combines simulated annealing (SA) with non-monotonic temperature control, random restart and hill-climbing (HC). The SA procedure uses five different neighborhood operators. The neighborhood operator used at each iteration is randomly selected. The SA temperature is increased after a certain number of iterations without improvement of the incumbent solution. An HC algorithm is applied to improve the best solution found by the SA. The HC algorithm uses the same neighborhood operators as the SA procedure. The hybrid heuristic is run 30 times and the best solution found over all runs is selected as the final solution.

Ursani *et al.* (2011) proposed a localized optimization framework (LOF). At the start of the search, an initial solution is created by using a split procedure. The optimization procedure consists of the following steps: (1) overlapping subproblems are generated considering all possible pair of routes from the current solution; (2) the overlapping subproblems are optimized separately by using a GA; (3) a non-bipartite matching problem is solved with aid of a GA to select the pairs of routes that produce the largest improvement on the objective function; with the selected routes, the procedure is repeated from step (1). When the optimization procedure does not produce any improvement, a de-optimization procedure is executed. This procedure modifies the solutions by successively reversing the sequences of customers, seeking to escape local optima. The optimization and de-optimization procedures are repeated during a fixed number of iterations.

Vidal *et al.* (2015) studied the impact of four different relaxation schemes (penalized late arrivals, both early and late arrivals, returns in time, and flexible speed) on the performance of three vehicle routing heuristics: a multistart local improvement (MS-LI) procedure, a multistart iterated local search (MS-ILS) and an MA. MS-LI consecutively runs a local improvement procedure based on traditional inter-route and intra-route neighborhoods. MS-ILS works in the following manner: First, an initial solution is created randomly. During a fixed number of iterations, new solutions are generated by a shaking operator that applies two swap moves to randomly-selected customers and the improvement method used in the MS-LI procedure. The

process finishes after a given number of consecutive iterations without improvement of the best solution. The MA considered in this study was that of Vidal *et al.* (2013).

2.2 Dynamic Vehicle Routing Problems

According to Psaraftis (1988), a VRP can be considered as dynamic if the inputs of the problem are updated concurrently with the creation and execution of the routing plan. In dynamic VRPs, the creation and execution of the routing plan are processes that are executed in parallel. In static VRPs, the routing plan is created prior to its execution and both processes do not overlap.

As reported by Pillac *et al.* (2013) and Psaraftis, Wen and Kontovas (2016), dynamic requests are the most common source of dynamism addressed in the literature on dynamic VRPs. Dynamic requests refer to the arrival of new requests, cancellation of requests, and changes in customer locations and/or demands. Dynamic travel times have also been addressed in several studies. Bektaş, Repoussis and Tarantilis (2014) remarked that the literature is scarce regarding less-predictable dynamic events as vehicle breakdowns, unexpected traffic congestion, accidents, and cargo damages.

Figure 1 illustrates the creation and execution of a routing plan in a dynamic VRP. The depot is labeled by number 0 and customers are labeled by numbers $1, \ldots, 10$. In this case, new requests arrive in real time while the vehicles are traversing their routes. At time t_1, an initial routing plan is defined with the customers that made their requests in advance. By time t_2, new requests have been received from customers 9 and 2 and the original routing plan has been updated accordingly. Customer 9's request was assigned to vehicle 1 and customer 2's request was assigned to vehicle 2. By time t_3, new requests have been received from customers 5 and 6 and the routing plan has been updated again. Customer 5's request was assigned to vehicle 2 and customer 6's request was assigned to vehicle 3. In this case, an extra vehicle was needed to serve all customers. At time t_4, the routing plan has been executed completely.

The main characteristics that differentiate the dynamic VRPs from their static counterparts are the following (PILLAC *et al.*, 2013):

1. In dynamic VRPs, routing plans must be updated on a continuous basis. This leads to a trade-off between solution quality and reactiveness. Whereas when solving static VRPs, several hours of computation would be required to obtain a solution, in dynamic VRPs fast solution methods are essential, since solutions often need to be updated within minutes or seconds after the occurrence of a dynamic event.

2. In dynamic VRPs, a real-time communication system is required to allow the interaction between the actors involved in the problem.

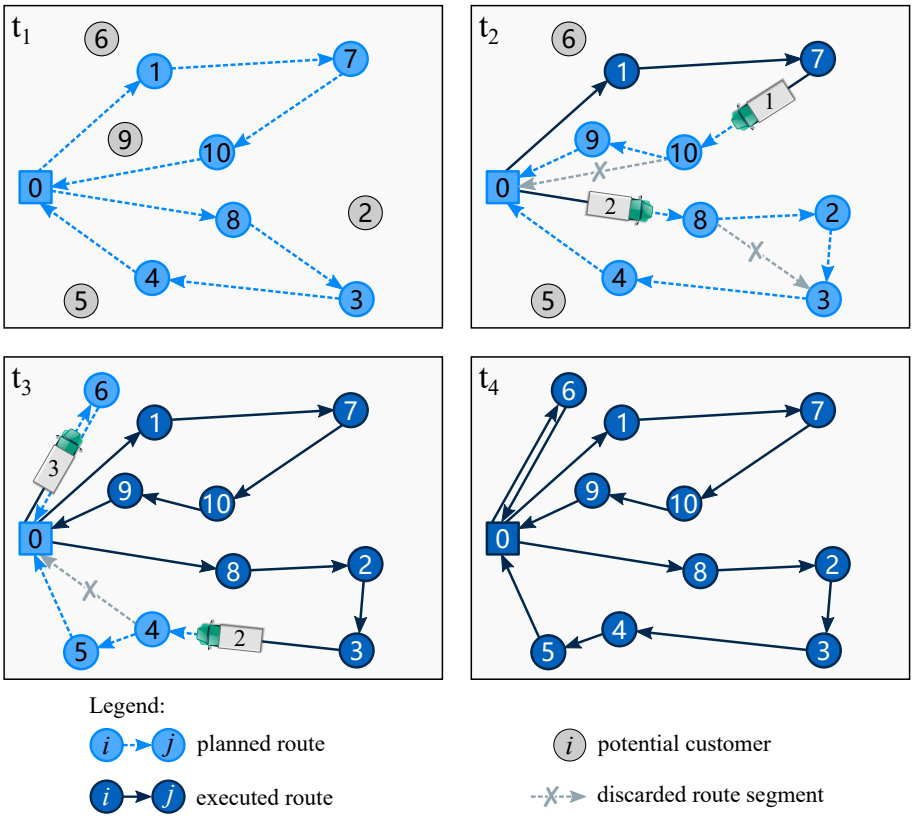


Figure 1 – Example of a dynamic VRP.

3. Due to operational limitations, rejection of requests is more common in dynamic VRPs than in static VRPs. For example, when customers must be served within a hard time window and the number of vehicles is limited, some requests would be rejected since they simply could not be feasibly served.

4. Objective functions can be different in dynamic VRPs. Whereas in static VRPs the aim generally is to minimize operational costs, measured in terms of total travel time or number of vehicles used, the aim in dynamic VRPs can be to assess the effectiveness of the responses to dynamic events. Objective functions can be expressed in terms of customers served per unit time, requests rejected per unit time, or time elapsed between the arrival of a request and its service.

A dynamic VRP is stochastic if some probabilistic information is known about the inputs that are updated dynamically. For example, customer demands may be assumed to follow a known probability distribution, although the actual demands will only be revealed when the vehicles visit the customers. Similarly, the probabilities that customers make requests could be estimated based on historical records of previous requests, although the actual number of requests will only be known at the end of the planning horizon. Available stochastic information can be integrated in the solution process to foresee likely events.

Pillac *et al.* (2013) and Bektaş, Repoussis and Tarantilis (2014) have noted that the

exploitation of stochastic information is a powerful tool to address dynamic VRPs since it can significantly improve the robustness and reliability of optimization approaches. Pillac *et al.* (2013) further remarked that the development of algorithms that make use of stochastic information to solve dynamic VRPs should become a priority.

Most real-life VRPs are dynamic, although they are often addressed as static VRPs (PSARAFTIS, 1995). An approach to cope with dynamism is to generate a static routing plan assuming that it will remain valid over a certain period of time. Whenever the inputs of the problem change, the plan is updated by performing a replanning from scratch with the updated information. This approach can work well for VRPs with low levels of dynamism (BRÄYSY; HASLE, 2014). Nevertheless, highly dynamic and/or uncertain problems should be modeled and addressed as dynamic VRPs. Some examples of highly dynamic problems are airport shuttle scheduling (KOHOUT; EROL, 1999), courier services (CORDEAU *et al.*, 2007), ambulance routing (DOERNER; HARTL, 2008), drayage operations (MÁHR *et al.*, 2010), and same-day delivery operations (SAVELSBERGH; WOENSEL, 2016).

2.2.1 Solution methods

In their comprehensive survey on dynamic VRPs, Psaraftis, Wen and Kontovas (2016) pointed out that most of the approaches to these problems are heuristic. This is because heuristic approaches usually require low computation times to find solutions, which makes these methods suitable to address problems that need high levels of reactiveness. From the survey, it is possible to conclude that the heuristic methods most commonly used to address dynamic VRPs are tabu search, GAs, MAs, insertion heuristics, and routing policies.

Both exact and heuristic approaches to optimization problems generally are centralized systems since all the information available about a given problem is the input for a single method that optimizes one or more objectives.

Multiagent systems are systems composed of multiple computing elements, called *agents*. Agents can autonomously take actions to reach their objectives. Furthermore, agents can interact with other agents, emulating social behaviors as cooperation, coordination, and negotiation (WOOLDRIDGE, 2009).

Multiagent systems can represent decentralized and centralized systems. A system is decentralized if all agents follow their own objectives using only the information they have available locally. In a centralized system, a coordinator agent has access to global information about the environment and guides other agents to achieve an objective.

Fischer, Müller and Pischel (1996) and Gath (2016) have argued that multiagent systems are particularly well-suited to address dynamic transportation problems, given that a decentralized system can cope with multiple dynamic events. Gath (2016) also presented a comprehensive survey on applications of multiagent systems to transportation problems.

A dynamic VRP can be modeled by a decentralized multiagent system in the following manner: Customer agents make requests. Vehicle agents organize an auction upon the arrival of a request. The bids represent the costs that vehicles would incur to serve the request. The winner vehicle is responsible for serving the request. Vehicles use fast heuristics to calculate the cost of serving a request. In addition, vehicles may cooperate between them to reduce their transport costs. For example, they can interchange their assigned requests with other vehicles, emulating in this way inter-route moves that are commonly used in centralized solution methods.

Although agents generally use simple heuristics to be responsive, decentralized multiagent systems can find solutions competitive with those found by more complex heuristic methods, as pointed out by Fischer, Müller and Pischel (1996). This fact was corroborated by Máhr *et al.* (2010), who compared the performances of a centralized algorithm based on mixed integer programming and a decentralized multiagent system. The authors applied both solution methods to a highly dynamic VRP arising in drayage operations and concluded that the multiagent system performed competitively with the centralized algorithm.

As pointed out by Máhr *et al.* (2010) and Gath (2016), the research on dynamic VRPs has been focused on the development of centralized solvers both exact and heuristic. Several decades of extensive research have consolidated these methods. Multiagent approaches to dynamic VRPs are less common, despite the suitability of these methods to model dynamic environments. Nevertheless, multiagent technologies to address dynamic VRPs can become more popular in the nearby future. New logistic paradigms as crowdshipping, where common citizens serve as couriers (Buldeo Rai *et al.*, 2017), bring forth a new class of decentralized and highly-dynamic routing problems, where not only requests are uncertain but also the transport resources to serve them. Certainly, multiagent systems could be very useful to address problems like those.

Along the same lines as Pillac *et al.* (2013), Máhr *et al.* (2010) and Gath (2016) stated that a very promising direction for further research on agent-based approaches to dynamic VRPs is the development of agents capable of exploiting probabilistic information about the problem, because this would allow them to anticipate future events and be more proactive.

2.2.2 The dynamic vehicle routing problem with time windows and stochastic customers

The dynamic vehicle routing problem with time windows and stochastic customers (DVRPTWSC) is a VRPTW where some customers make their service requests before the start of the planning horizon and other customers make their service requests over the planning horizon, while vehicles are traversing their routes. Thus, it is necessary to update the initial routing plan in real time to accommodate the new requests. Stochastic information about the new requests is available to be used by the solution method. This subsection briefly reviews some studies that have addressed DVRPTWSCs.

Bent and Van Hentenryck (2004c) addressed the DVRPTWSC aiming to maximize the number of served customers. To solve the problem, the authors presented a multi scenario approach (MSA). MSA continuously updates a pool of routing plans that include known and future requests. Future requests are sampled from known probability distributions. The hybrid local search of Bent and Van Hentenryck (2004a) is used to optimize the routing plans. Before adding the routing plans to the pool, future requests are removed. The resulting routing plans are expected to leave room for accommodating future requests if they materialize. When an event occurs, the pool of routing plans is updated accordingly and the plans that become inconsistent are deleted. New dynamic requests are accepted if they can be feasibly included in at least one routing plan. The plan to be executed is chosen with aid of a consensus function that selects the routing plan most similar to other plans in the pool. The authors evaluated the performance of MSA on instances with a degree of dynamism varying between 30% and 80%. The instances were generated from the benchmark set of Solomon (1987). MSA achieved significant improvements over approaches that do not exploit stochastic information. The benefits of MSA were more notorious on instances with high levels of dynamism.

As an alternative to the consensus function, Bent and Van Hentenryck (2004b) proposed the regret function, which approximates the value of scheduling each request next on the vehicles. Experiments conducted showed that the regret function produced noticeable improvements on instances with high level of dynamism (about 70%). Later, Bent and Van Hentenryck (2007) incorporated waiting and relocation strategies in MSA. Both strategies improved the quality of the solutions obtained, mainly when used along with the regret function. The relocation strategy dominated the waiting strategy as the level of dynamism increased.

Hvattum, Løkketangen and Laporte (2006) addressed a real-world DVRPTWSC arising in the operations of a distribution company in Norway. The objective of the problem is to minimize the number of vehicles used plus the total travel time. Customers can appear at unknown locations with random demands at any time over the planning horizon. Around 50% of the requests are known by the start of the planning horizon, the remaining requests must be processed in real time. To model the problem, the authors divided the geographical area where customers were located in $n \times n$ sectors, and they assumed that customers were uniformly distributed within each sector. Historical customer locations were used to estimate the probability of receiving requests from a given sector. The Poisson distribution was used to represent the number of requests received at a given time interval over the planning horizon. Customers' demands were drawn with equal probabilities from a historical database of demands. The problem was formulated as a multistage stochastic model with recourse. To solve the problem, the authors developed the dynamic stochastic hedging heuristic (DSHH). For each time interval in which the planning horizon is divided, DSHH solves a set of sampled scenarios using an insertion heuristic and uses common features from the solutions to build a "good plan". Each scenario contains all known customers and a set of sampled potential customers. The authors conducted experiments with DSHH and a myopic dynamic heuristic (MDH) that does not take into account the available

stochastic information. DSHH was able to reduce travel distances by more than 15% with respect to MDH, but it occasionally produced solutions requiring extra vehicles.

In a subsequent study, Hvattum, Løkketangen and Laporte (2007) extended their previous work: They proposed the branch-and-regret heuristic (BRH), which is an improved version of DSHH. In BRH, the process for selecting a plan from the pool of sampled scenarios is modified. In this case, a plan is created by a procedure that iteratively merges the sampled scenarios. The authors also extended the original problem by introducing the possibility of stochastic demands for known customers. BRH was shown to be effective in tackling different types of stochasticity.

Barkaoui, Berger and Boukhtouta (2015) addressed a DVRPTWSC with soft time windows, i.e., time windows that can be violated incurring in a penalty. In the problem considered, customers are associated with a satisfaction level that varies in function of the time they are visited. Customers who are served within their time windows have high satisfaction levels, and customers visited earlier or later than expected have low satisfaction levels. Customers may need several visits to achieve a desired level of satisfaction. The objectives considered were to maximize the number of satisfied customers and minimize the total travel distance and violations of temporal constraints. The authors developed a GA that continuously generates routing plans considering known and potential requests. The information related to customer satisfaction is used to anticipate future requests. Insertion and exchange procedures are used to process new requests. The authors used instances derived from the Solomon's (1987) benchmark set to test their solution approach. The authors compared their algorithm with a myopic version of it, which does not consider future potential requests, and a greedy method. The proposed GA performed well against the other algorithms over the range of satisfaction thresholds the authors considered. The authors remarked that by anticipating future requests, the GA was able to react more quickly to dynamic requests and customer satisfaction increased.

Barkaoui (2017) addressed the DVRPTWSC with soft time windows seeking to minimize the number of routes as first objective and minimize the total travel distance as second objective. The author presented an adaptive hybrid genetic algorithm (AHGA) that consists of two GAs. The first GA evolves a population of solutions to the original problem and the second GA evolves a population of operator combinations that are used by the first GA. A co-evolutionary genetic algorithm (CGA) is responsible for generating waiting strategies. As AHGA, CGA consists of two GAs. One GA continuously generates scenarios that include possible dynamic requests. These scenarios are used by the second GA to generate waiting strategies. The best waiting strategies generated are sent to AHGA and they are used to solve the routing problem in real time. The reported results showed that AHGA with waiting strategies significantly outperformed other heuristic approaches.

Zou and Dessouky (2018) addressed the DVRPTWSC aiming to minimize the total travel distance. The authors developed an optimization-based look-ahead dynamic routing framework. The planning horizon is divided into a fixed number of intervals of equal length. Periodically, at

fixed time intervals, the current routing plan is updated by solving a VRP that considers both known and anticipated customers. At each update, a new set of anticipated customers is included in the routing plan. An iterative insertion-based heuristic is used to solve the VRPs and an SA heuristic is used to refine the solutions. A hybrid waiting time adjustment heuristic optimizes the time slots in the current routing plan to maximize the chances of accommodating potential requests. When a customer makes a request, the algorithm checks whether the customer's request was anticipated. If so, the customer is only marked as confirmed. Otherwise, a cheapest insertion heuristic is used to process the request. To test their approach, the authors generated different dynamic instances from the Solomon's (1987) RC201 instance. The proposed approach was shown to be more reliable than a purely reactive approach on instances with different proportions of realized dynamic requests.

CHAPTER

3

THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Before dealing with the allocation of the dynamic requests, it is necessary to focus our attention on the static customers, i.e., the customers that have made their requests in advance and are known at planning time. The first problem to be solved is the creation of an initial routing plan for serving those customers. In other words, it is necessary to solve an instance of the vehicle routing problem with time windows (VRPTW).

In this case, the VRPTW is addressed seeking to minimize the total travel time. To this aim, a simple yet effective memetic algorithm (MA) is developed. A clustering procedure is applied to customers' spatial data at the beginning of the MA execution. The crossover and local search operators take advantage of the information gathered by the clustering procedure. Both operators relocate customers between routes that pass close to each other, seeking to minimize the detour costs associated with the modifications. The clusters "guide" the search since they are used to identify which routes pass close to each other.

Experiments conducted on the Solomon's (1987) benchmark set show a competitive performance of the proposed MA. The proposed MA outperforms four out of six solution approaches considered for comparison (when comparing the total cumulative travel distance/time), and it finds the best-known solutions for several instances.

The main contributions of this chapter are (1) the introduction of two search procedures, the clustering-based best cost route crossover (CB-BCRC) and the clustering-based inter-route neighborhood search (CB-Inter-Route-NS); (2) computational experiments that show that the CB-BCRC procedure can produce solutions competitive with those produced by an approach based on exhaustive search, while requiring lower execution times; and (3) computational experiments that show that the proposed MA can produce solutions competitive with those produced by state-of-the-art methods.

The remainder of this chapter is organized as follows: Section 3.1 presents the definition of the VRPTW. Section 3.2 details the proposed MA. Section 3.3 presents the experimental results. Finally, Section 3.4 concludes the chapter.

3.1 Problem Definition

The VRPTW consists of determining a set of routes for distributing goods from a single depot to a set of n geographically scattered customers. The problem can be defined on a complete directed graph $G = (V, E)$. The set of nodes is given by $V = \{0, 1, \ldots, n\}$. Node 0 represents the depot, whereas $N = \{1, \ldots, n\}$ represents the set of customer nodes. Each node $i \in N$ is associated with a demand $q_i > 0$, a service time $s_i > 0$, and a time window $[e_i, l_i]$. The depot has no demand or service time, i.e., $q_0 = 0$ and $s_0 = 0$. The depot's time window represents the planning horizon and is given by $[e_0, l_0]$, where e_0 and l_0 represent the earliest possible departure time from the depot and the latest possible arrival time at the depot, respectively. The set of edges is given by $E = \{(i, j) | i, j \in V, i \neq j\}$. Each edge (i, j) is associated with a distance $d_{ij} \geq 0$ and a travel time $t_{ij} \geq 0$. A fleet of M vehicles is available for serving the customers. The fleet is assumed to be homogeneous, since all vehicles have the same capacity Q, and they operate at the same cost.

A route is defined by a sequence of nodes $r^v = \langle r_0^v, r_1^v, \ldots, r_{n_v}^v, r_{n_v+1}^v \rangle$, where v is the vehicle responsible for traversing the route and r_k^v represents the kth node visited by vehicle v. It should be noted that a vehicle can perform only one route over the planning horizon. Given that every route starts and ends at the depot, $r_0^v = r_{n_v+1}^v = 0$. For $1 \leq k \leq n_v$, $r_k^v \in N$ and all r_k^v are distinct, since all customers are visited at most once.

The travel time of a route r^v is given by

$$f(r^v) = \sum_{k=0}^{n_v} t_{r_k^v, r_{k+1}^v}. \tag{3.1}$$

Note that the travel time of a route is different from the total route time, which is the sum of total travel time, total waiting time, and total service time.

The start of service time $T_{r_k^v}$ at node r_k^v is given by the following recursive equation (DESAULNIERS; MADSEN; ROPKE, 2014):

$$T_{r_k^v} = \begin{cases} e_0 & \text{if } k = 0 \\ \max\left\{ e_{r_k^v}, T_{r_{k-1}^v} + s_{r_{k-1}^v} + t_{r_{k-1}^v, r_k^v} \right\} & \text{otherwise.} \end{cases} \tag{3.2}$$

If vehicle v arrives at node r_k^v before $e_{r_k^v}$, it should wait to start the service. Arriving after $l_{r_k^v}$ is not allowed. Formally, the route r^v is feasible regarding the time-window constraints if $T_{r_k^v} \leq l_{r_k^v}$ for $k = 1, \ldots, n_v + 1$. The vehicle-capacity constraint is satisfied if $\sum_{k=1}^{n_v} q_{r_k^v} \leq$ Q. If both time-window and vehicle-capacity constraints are satisfied, the route r^v is said to be feasible.

Violations of time-window and vehicle-capacity constraints must be calculated to penalize infeasible routes. For a route r^v, time-window violations are given by

$$f_{tw}(r^v) = \sum_{k=1}^{n_v+1} \max\left\{T_{r_k^v} - l_{r_k^v}, 0\right\}. \tag{3.3}$$

Vehicle-capacity violations are given by

$$f_q(r^v) = \max\left\{\sum_{k=1}^{n_v} q_{r_k^v} - \mathrm{Q}, 0\right\}. \tag{3.4}$$

A solution for the VRPTW is defined as a set of m routes $S = \left\{r^1, \ldots, r^m\right\}$ where each customer is served exactly once and $m \leq \mathrm{M}$.

The objective function to be minimized is

$$F(S) = \sum_{v=1}^{m} \left[f(r^v) + \alpha_1 f_{tw}(r^v) + \alpha_2 f_q(r^v)\right]. \tag{3.5}$$

Function (3.5) calculates the total travel time taken by the vehicles for serving all customers; α_1 and α_2 are weights used to penalize the violations to time-window and vehicle-capacity constraints, respectively. A solution is said to be feasible if

$$f_{tw}(r^v) = f_q(r^v) = 0, \text{ for } v = 1, \ldots, m. \tag{3.6}$$

The objective of the problem is to find a feasible solution S^* that minimizes $F(S)$.

3.2 Memetic Algorithm

The proposed MA performs a clustering analysis phase where the customers are clustered based on their spatial distance. The clusters obtained from that process have two purposes: they are used to create some initial solutions, and more importantly, they serve as a guide for the search operators of the MA. The details of the clustering process and the search methodology are presented below.

3.2.1 Solution Representation

Given that a solution S for a VRPTW instance is a set of routes, a solution in the MA is represented as a set of lists, where each list represents a route. Customers are visited according to the order in which their identifiers appear within the list. Since every route starts and ends at the depot, the identifier of the depot, 0, is not considered in this representation. An example of this representation is shown in Figure 2.

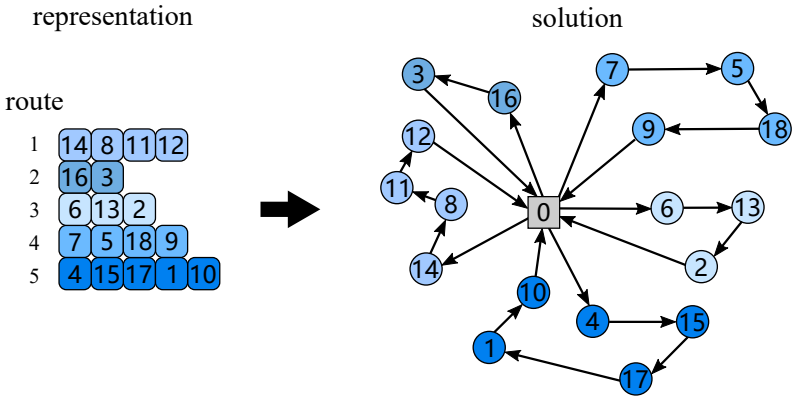


Figure 2 – Representation of a solution with five routes.

3.2.2 Fitness Function

The fitness of a solution is calculated using Function (3.5).

3.2.3 Generation of the initial population

The initial population of the MA is composed of two kinds of solutions. Solutions of *kind 1* are created by performing hierarchical agglomerative clustering (HAC) on the matrix of distances between customers. To create a solution, the dendrogram obtained from the clustering process is cut at one point, so that a clustering arrangement is obtained. The solution is put together by defining a route on each cluster and letting the visit order be random. Several clustering arrangements, with different numbers of clusters, are obtained by cutting the dendrogram at different points (see Xu and Wunsch (2008) for further details). The number of clusters created varies between two values: C_{min} and C_{max}. The aim of this approach is to define routes composed of close customers. These solutions do not necessarily satisfy the time-window and vehicle-capacity constraints.

In addition to being used to create solutions of *kind 1*, the clustering arrangements generated are saved in the set A, which is used afterward by the search operators of the MA.

Solutions of *kind 2* are created by forming routes with customers selected randomly, checking that the vehicle-capacity constraint is not violated. These solutions are created to increase the diversity of the initial population. Solutions of *kind 1* and *kind 2* are generated with probabilities ρ and $(1-\rho)$, respectively.

Figure 3 shows three examples of solutions created for the same VRPTW instance. Figure 3.1 and Figure 3.2 show two solutions of *kind 1*, with 6 and 8 clusters, respectively. Figure 3.3 shows a solution of *kind 2*, generated randomly.

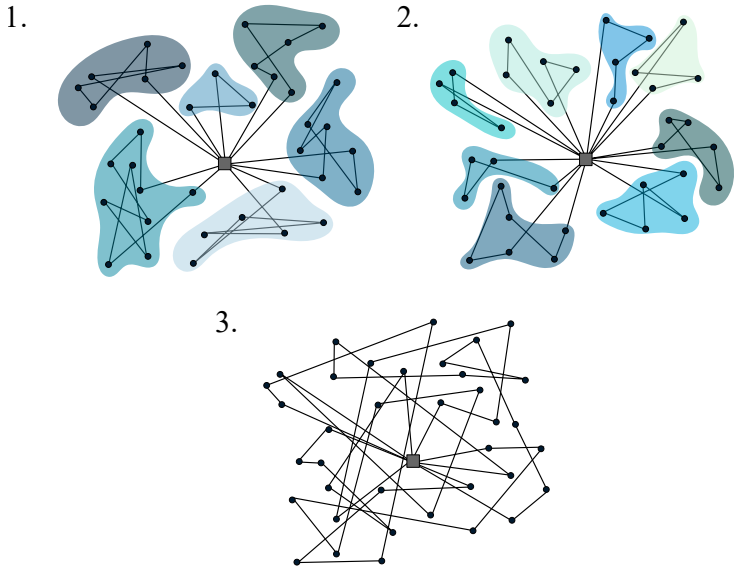


Figure 3 – Examples of solutions created for the initial population.

3.2.4 Crossover

The crossover operator is responsible for the exploration of the search space. This operator takes available solutions and combines the information encoded by them to create new ones. Problem-specific crossover operators can use knowledge of the problem to increase the performance of the algorithm.

A problem-specific crossover operator used in evolutionary algorithms for the VRPTW is the best cost route crossover (BCRC) (OMBUKI; ROSS; HANSHAR, 2006). This operator takes two solutions and creates a new one by removing customers and reinserting them at the best feasible locations. Given two solutions S_1 and S_2, the process begins by selecting randomly a route from S_1. The customers belonging to the selected route are removed from S_2. Finally, each missing customer is reinserted at a feasible location that yields the minimum detour cost. The BCRC has been used successfully in evolutionary algorithms for the VRPTW; however, the reinsertion phase is based on an exhaustive search procedure that can be very time consuming (PIERRE; ZAKARIA, 2017). This happens because all routes are considered for the reinsertion.

Seeking to reduce the time required by the reinsertion phase of the BCRC, the clustering-based best cost route crossover (CB-BCRC) was developed. The CB-BCRC takes two solutions and a clustering arrangement as input and creates a new solution following a process similar to the original BCRC procedure: Given two solutions S_1 and S_2, the process begins by selecting a route from S_1 at random. Customers belonging to the selected route are removed from S_2. The new solution is obtained by reinserting the removed customers at the best feasible locations in S_2.

The main difference between the original BCRC procedure and the proposed one lies in the reinsertion phase. In the original procedure, the reinsertion phase seeks the best feasible

location through all routes, considering all positions within them. Such operation turns out to be rather expensive computationally. In the CB-BCRC, the reinsertion phase is modified such that only locations within routes close to removed customers are considered. The clustering arrangements saved in set A are used to identify candidate routes for the reinsertion process. The rationale behind this approach is that removed customers can be accommodated at a lower cost within the routes passing close to them, because the detour costs are reduced. Furthermore, a performance boost is obtained, since the search is now restricted to a subset of possible locations.

Figure 4 illustrates the application of the CB-BCRC operator on a given VRPTW instance with 25 customers. Let S_1 and S_2 be two solutions (Figure 4.1), and let C be a clustering arrangement of customers containing six clusters (Figure 4.2). The procedure begins by selecting randomly a route from S_1; in this case, the route $0 \rightarrow 18 \rightarrow 8 \rightarrow 17 \rightarrow 16 \rightarrow 13 \rightarrow 0$ is selected. This route passes through clusters c_2, c_1, and c_6; a route is said to pass through a cluster if both have at least one common customer (Figure 4.3). The routes from S_2 passing through those clusters are found, those routes are $0 \rightarrow 6 \rightarrow 18 \rightarrow 8 \rightarrow 7 \rightarrow 0$, $0 \rightarrow 5 \rightarrow 17 \rightarrow 16 \rightarrow 14 \rightarrow 13 \rightarrow 0$, and $0 \rightarrow 2 \rightarrow 15 \rightarrow 22 \rightarrow 23 \rightarrow 4 \rightarrow 21 \rightarrow 0$ (Figure 4.4). Customers belonging to the route selected from S_1 are removed from S_2 (Figure 4.5). The customers removed are reinserted at the best feasible locations on the routes identified in S_2; this operation produces three new routes: $0 \rightarrow 6 \rightarrow 8 \rightarrow 7 \rightarrow 0$, $0 \rightarrow 18 \rightarrow 5 \rightarrow 17 \rightarrow 16 \rightarrow 14 \rightarrow 0$, and $0 \rightarrow 2 \rightarrow 15 \rightarrow 22 \rightarrow 23 \rightarrow 4 \rightarrow 21 \rightarrow 13 \rightarrow 0$ (Figure 4.6). After the reinsertion process, the new solution is obtained (Figure 4.7).

The pseudocode for the CB-BCRC operator is presented in Figure 5. The CB-BCRC procedure takes as input two solutions S_1 and S_2, and a clustering arrangement C. In line 1, a route is selected randomly from S_1. Clusters through which the selected route passes are identified in line 2. Line 3 finds the routes of S_2 passing through the clusters identified previously. In line 4, the customers belonging to the chosen route from S_1 are removed from S_2. The reinsertion phase is performed in line 5: All removed customers are reinserted on the routes identified in line 3, seeking the feasible locations yielding the lowest detour costs. If no feasible location exists, a new route, containing only the processed customer is created. The new route is considered for future reinsertions. Customers are reinserted in random order. Finally, line 6 returns the resulting solution.

The pseudocode for the crossover operator is presented in Figure 6. The CROSSOVER procedure takes as input two solutions S_1 and S_2, and the set of clustering arrangements A, which was created during the generation of the initial population. A clustering arrangement C is randomly selected in line 1. C is used for calling the CB-BCRC procedure in line 2.

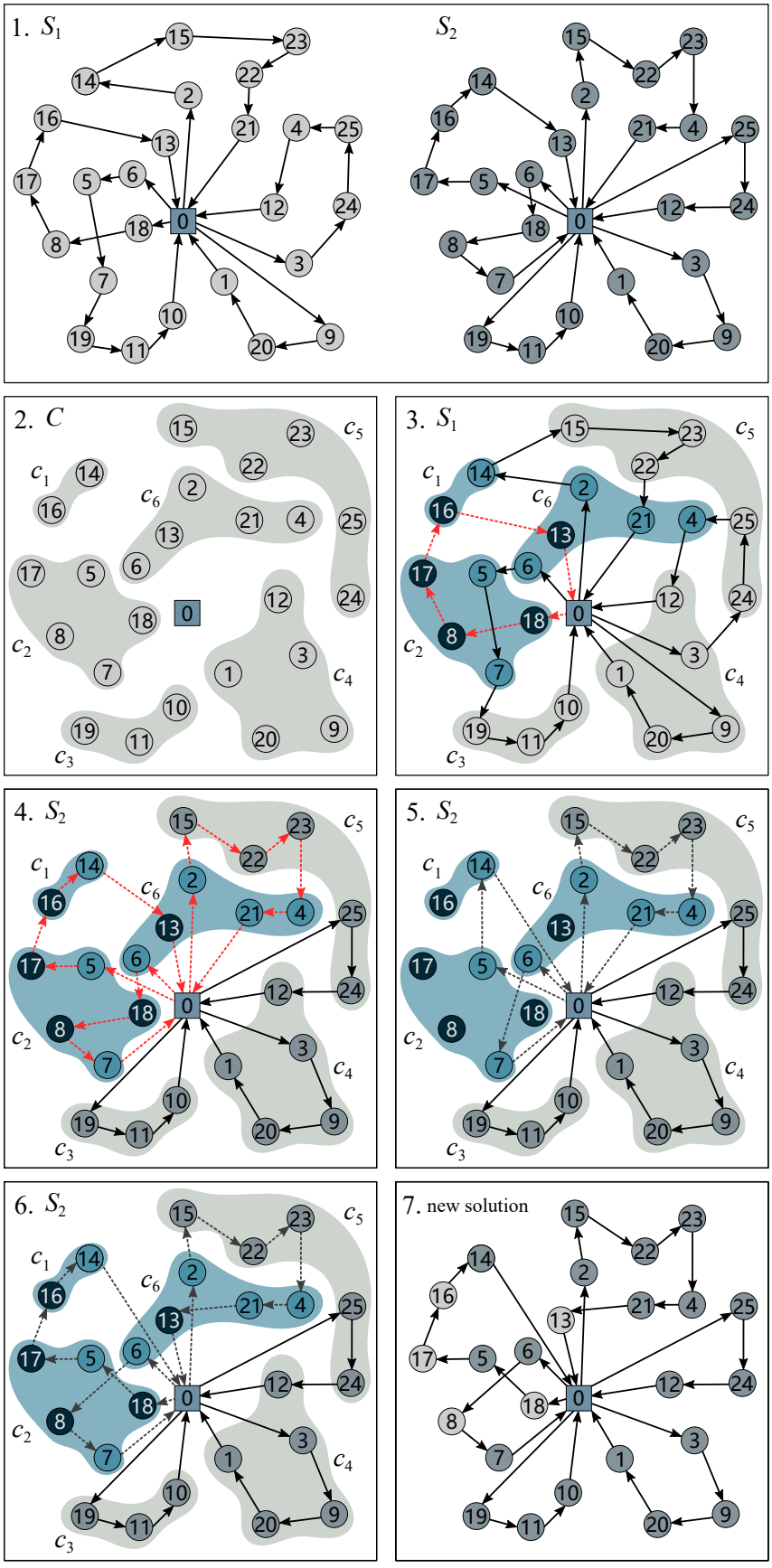


Figure 4 – Example: CB-BCRC operator.

CB-BCRC(S_1, S_2, C)
1 $route$ = SELECT-A-ROUTE(S_1)
2 $clusters$ = FIND-CLUSTERS$(route, C)$
3 $routes$ = FIND-ROUTES$(clusters, S_2)$
4 REMOVE-CUSTOMERS$(route, S_2)$
5 REINSERT-CUSTOMERS$(routes, S_2)$
6 **return** S_2

Figure 5 – Pseudocode for the CB-BCRC operator.

CROSSOVER(S_1, S_2, A)
1 C = RANDOM(A)
2 **return** CB-BCRC(S_1, S_2, C)

Figure 6 – Pseudocode for the crossover operator.

3.2.5 Local Search

The local search operator is responsible for performing an exploitative search that seeks to refine the solutions found by the crossover operator. Two kinds of neighborhoods are considered within the search: *inter-route* neighborhoods, where solutions are obtained by moving one or more customers between routes; and *intra-route* neighborhoods, where solutions are obtained by modifying a single route. The clustering arrangements saved in set A are used for guiding the search through the inter-route neighborhoods. Details on the local search process are given below.

3.2.5.1 Inter-route neighborhood search

Since the inter-route neighborhoods are composed of solutions obtained by moving customers between two or more routes, a question that must be addressed is which routes are good candidates for performing such moves.

The search procedure begins by identifying a set of routes passing close to each other; this is achieved using the clustering arrangements saved in set A. Once the candidate routes have been identified, traditional inter-route neighborhood moves are applied to modify them (reviews on traditional neighborhoods for the VRP can be found in Desaulniers, Madsen and Ropke (2014) and Funke, Grünert and Irnich (2005)).

By only moving customers between routes passing close to each other, the search procedure seeks to minimize the detour costs associated with such moves, thus producing good-quality solutions. On the other hand, a blind exchange of customers between routes passing far apart from each other could lead to low-quality solutions, since the detour costs of the exchange

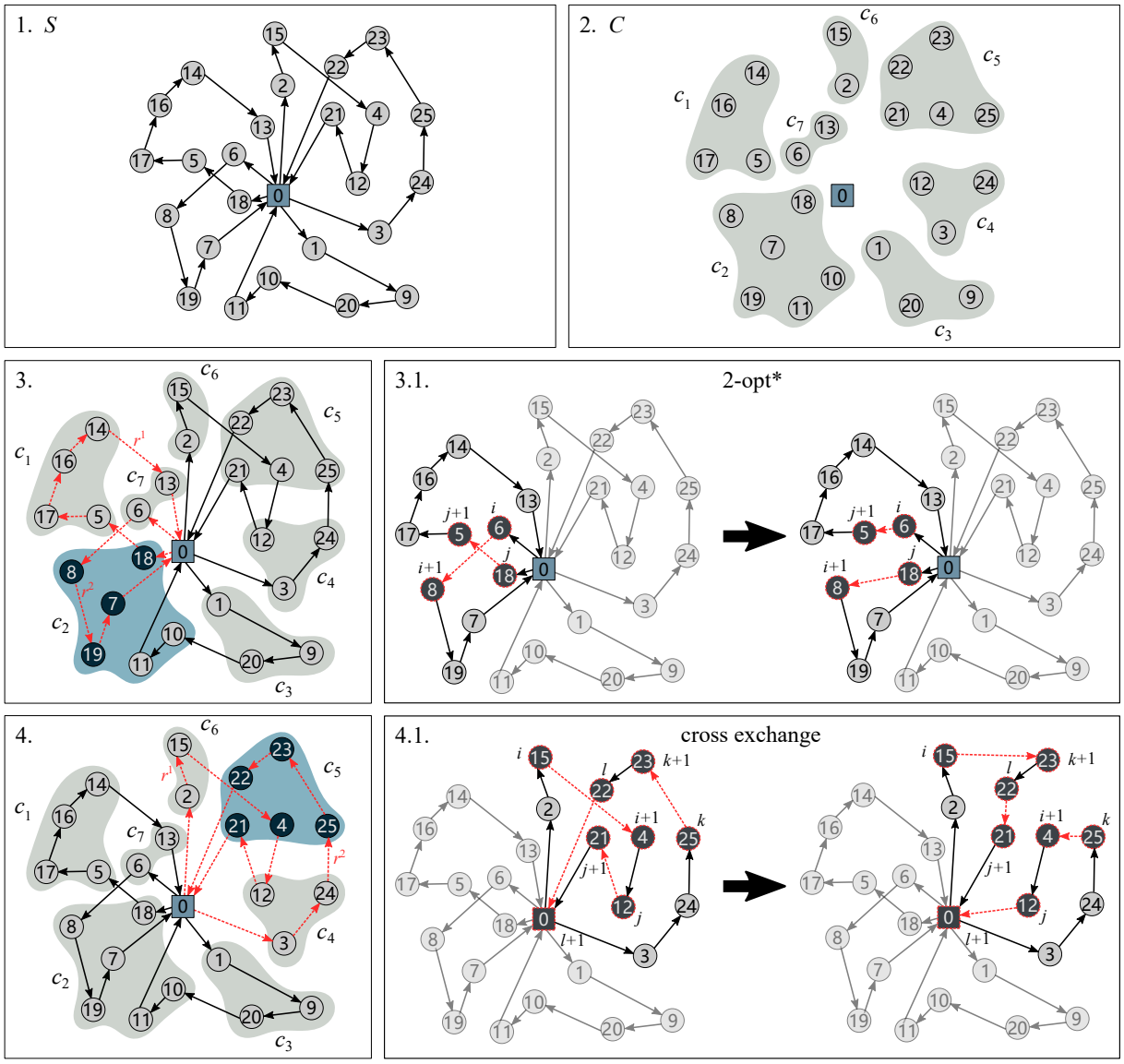


Figure 7 – Example: inter-route neighborhood search.

could be high.

Figure 7 shows an example of how the routes are selected and modified. Let S be a solution (Figure 7.1) and let C be a clustering arrangement of customers containing seven clusters (Figure 7.2). The procedure begins by selecting a cluster and identifying the routes passing through it (a route is said to pass through a cluster if both have at least one common customer). The routes found are considered to visit a common region, and thus they pass close to each other. At least two different routes should be found in order to continue. From the routes identified, two are randomly selected. In the example, clusters c_2 and c_5 were chosen (Figure 7.3 and Figure 7.4, respectively), and the routes selected are shown by dashed lines. A new solution is obtained by applying traditional inter-route moves to the selected routes. In the example, the *2-opt** move is applied to the routes found in Figure 7.3.: the arcs $(i, i+1)$ and $(j, j+1)$ are replaced by the arcs $(i, j+1)$ and $(j, i+1)$ (Figure 7.3.1). Similarly, the *Cross Exchange* move

```
CB-INTER-ROUTE-NS(S, A, 𝒩)
 1  if |S| < 2
 2      return S
 3  S_best = S
 4  C = RANDOM(A)
 5  for each cluster c ∈ C
 6      routes = FIND-ROUTES(S_best, c)
 7      if |routes| < 2
 8          continue
 9      r^1, r^2 = RANDOM(routes)
10      S' = 𝒩(S_best, r^1, r^2)
11      if F(S') < F(S_best)
12          S_best = S'
13  return S_best
```

Figure 8 – Pseudocode for the clustering-based inter-route neighborhood search procedure.

is applied to the routes found in Figure 7.4.: the arcs $(i,i+1)$, $(j,j+1)$, $(k,k+1)$, and $(l,l+1)$ are replaced by the arcs $(i,k+1)$, $(l,j+1)$, $(k,i+1)$, and $(j,l+1)$ (Figure 7.4.1). In both cases, the arcs to be modified are selected randomly.

The inter-route neighborhood search consists of applying iteratively the process illustrated in Figure 7. The pseudocode for the clustering-based inter-route neighborhood search (CB-Inter-Route-NS) is presented in Figure 8. The CB-INTER-ROUTE-NS procedure takes as input a solution S, a set of clustering arrangements A, and a neighborhood function $\mathscr{N}$. The latter takes as input a solution S and the indexes of two of its routes, r^1 and r^2, and it creates a new solution by applying an inter-route move to a set of arcs randomly selected from r^1 and r^2. Since the inter-route moves can be applied only to solutions with two or more routes, line 1 checks whether S meets this requirement. If S has only one route, it is returned without changes (line 2). The incumbent is initialized in line 3. A clustering arrangement C is randomly selected from A in line 4. The loop on lines 5-12 performs the search process; basically, it executes the process illustrated in Figure 7 with each cluster $c \in C$. Line 6 finds the routes of the incumbent passing through cluster c. If there are not at least two routes passing through c, the search continues with the next cluster (lines 7 and 8). Optionally, the routes found can be checked for ensuring they meet some criteria, for example having a minimum number of customers. From the routes identified, r^1 and r^2 are randomly selected in line 9. Line 10 creates a neighbor solution S' by applying the function $\mathscr{N}$ to the routes r^1 and r^2 from the incumbent. Lines 11 and 12 update the incumbent when a better solution is found. Finally, line 13 returns the best found solution.

3.2.5.2 Intra-route neighborhood search

Seeking to further improve the solutions obtained by the CB-INTER-ROUTE-NS procedure, an intra-route neighborhood search procedure is applied to one randomly selected route. In this case, a traditional simulated annealing (SA) heuristic is responsible for performing the search. The SA heuristic randomly applies the intra-route neighborhood moves *2-opt*, *or-opt*, and *swap* to perform the search (for further details on SA and the mentioned neighborhood moves, the reader is referred to Kirkpatrick, Gelatt and Vecchi (1983) and Funke, Grünert and Irnich (2005), respectively).

3.2.5.3 Variable neighborhood search procedure

The inter-route and intra-route neighborhood searches are carried out within a variable neighborhood search (VNS) framework (MLADENOVIĆ; HANSEN, 1997). The pseudocode for the VNS procedure is presented in Figure 9. In line 1, the list of neighborhood functions $\mathcal{N}_l$ is initialized. In this case, the neighborhoods considered are *2-opt** and *Cross Exchange*. The incumbent is initialized in line 2. Line 3 sets the neighborhood function to start the search with. Lines 4-10 comprise the search procedure. The length of list $\mathcal{N}_l$ is denoted by len($\mathcal{N}_l$). The INTER-ROUTE-NS procedure in line 5 performs the inter-route neighborhood search. This procedure takes as input the incumbent, the set of clustering arrangements, a neighborhood function, and a number of iterations I_{max}. It applies the procedure CB-INTER-ROUTE-NS to the incumbent during I_{max} iterations to obtain a new solution S'. This step can be considered a combination of the shaking and local search steps of the traditional VNS. The intra-route phase of the local search is applied in line 6. The INTRA-ROUTE-NS procedure takes as input a solution S', a temperature value γ, and a cooling rate value β. It applies the SA heuristic to one randomly chosen route from S'. If the updated S' turns out to be a better solution, it becomes the

```
VNS(S, A, I_max, γ, β)
 1  N_l = ⟨2-opt*, cross-exchange⟩
 2  S_best = S
 3  k = 1
 4  while k ≤ len(N_l)
 5      S' = INTER-ROUTE-NS(S_best, A, N_l[k], I_max)
 6      S' = INTRA-ROUTE-NS(S', γ, β)
 7      if F(S') < F(S_best)
 8          S_best = S'
 9          k = 1
10      else k = k + 1
11  return S_best
```

Figure 9 – Pseudocode for the VNS procedure.

incumbent, and the search process is restarted with the first neighborhood function. If no better solution is found, the algorithm continues the search with the next neighborhood function (lines 7-10). Finally, line 11 returns the best found solution.

3.2.6 MA main procedure

The pseudocode for the proposed MA is presented in Figure 10. The MA procedure takes as input a VRPTW instance, X; the population size, $\mathscr{L}$; the maximum number of generations, G_{max}; the crossover rate, R_{rate}; the local search rate, L_{rate}; the parameters that control the local search: I_{max}, γ, and β; and the parameters used for creating the initial population of solutions: C_{min}, C_{max}, and ρ. The population of solutions P and a set of clustering arrangements A are created in line 1. The incumbent is initialized in line 2. The loop on lines 3-7 performs the search by updating the population of solutions and the incumbent at each iteration.

The way in which the solutions are updated is presented in Figure 11. The NEW-POP procedure seeks to replace each solution in P with a better one. The procedure begins with the creation of an empty population of solutions Q, which is filled by the loop on lines 2-13. This loop goes through each solution S_j in P and creates an updated version of it in the following manner: First, an additional solution S_o is taken at random from P (line 4). Both solutions S_j and S_o undergo crossover, and a child solution S_c is created (line 7). If the cost of S_c is more favorable than that of S_j, S_c is kept; otherwise S_j becomes S_c (lines 8 and 9). Next, the local search procedure is applied to S_c, and the solution obtained from this process takes the place of S_j in the next generation (lines 12 and 13). Finally, the updated population of solutions is returned in line 14.

It is necessary to note that as the search goes on, the MA is expected to deal with multiple locally minimal solutions that will survive several generations until better solutions replace them. For that reason, the intra-route local search operator is applied to only one route. Since a solution can remain for several successive generations, the intra-route search will eventually be performed

```
MA(X, 𝓛, G_max, R_rate, L_rate, I_max, γ, β, C_min, C_max, ρ)
1  P,A = INITIAL-POP(X, 𝓛, C_min, C_max, ρ)
2  S_best = GET-BEST-SOLUTION(P)
3  for i = 1 to G_max
4       P = NEW-POP(P,A, 𝓛, R_rate, L_rate, I_max, γ, β)
5       S' = GET-BEST-SOLUTION(P)
6       if F(S') < F(S_best)
7            S_best = S'
8  return S_best
```

Figure 10 – Pseudocode for the main procedure of the MA.

```
NEW-POP(P, A, 𝓛, R_rate, L_rate, I_max, γ, β)
 1  let Q[1..𝓛] be a new array
 2  for j = 1 to 𝓛
 3      S_j = P[j]
 4      S_o = RANDOM(P)
 5      let S_c be a new solution
 6      if U(0,1) ≤ R_rate
 7          S_c = CROSSOVER(S_o, S_j, A)
 8          if F(S_c) > F(S_j)
 9              S_c = S_j
10      else S_c = S_j
11      if U(0,1) ≤ L_rate
12          S_c = VNS(S_c, A, I_max, γ, β)
13      Q[j] = S_c
14  return Q
```

Figure 11 – Pseudocode for the creation of a new population of solutions.

on several routes of the solution.

3.3 Computational Results

All the procedures described in the previous section were implemented using the Kotlin programming language. The experiments were conducted on an Intel Core i7-4790U PC with 16 GB memory.

The well-known Solomon's (1987) VRPTW benchmark set was selected for testing the proposed MA. This benchmark set includes 56 instances with 100 customers. Depending on the spatial distribution of the customers, time-window characteristics, and vehicle capacity, the instances are classified into classes C1, R1, RC1, C2, R2, and RC2. The customers are clustered in instances of type C and randomly located in instances of type R. In instances of type RC, there is a combination of clustered and randomly located customers. In instances of type 1, the customers have narrow time windows and the vehicle capacity is small, thus more vehicles are required for serving all customers. In instances of type 2, the customers have wider time windows and the vehicle capacity is larger, thus fewer vehicles are needed to serve all customers. The service time is the same for all customers; in instances of type C, the service time is equal to 90, whereas in instances of type R and RC, the service time is equal to 10.

Two sets of computational experiments were conducted to assess the performance of the proposed solution approach. In the first set of experiments, the performance of the CB-BCRC and the BCRC was compared in terms of solution quality and execution time. In the second set of experiments, the performance of the proposed MA was assessed by comparing the obtained

results against those of other heuristics, as traditionally done in the VRPTW literature. The results of both sets of experiments were also considered to assess the impact of the local search on solution quality.

3.3.1 The CB-BCRC vs. the BCRC

Two genetic algorithms (GAs), $GA_{CB-BCRC}$ and GA_{BCRC}, were developed to compare the performance of the CB-BCRC and the BCRC. The main procedure in both GAs is the same as in the proposed MA (Figure 10). Nevertheless, the local search phase is skipped in the GAs, i.e., the parameters that control the local search are excluded and lines 11-12 of the NEW-POP procedure (Figure 11) are omitted when creating a new population of solutions. In addition, GA_{BCRC} uses the original BCRC instead of the CB-BCRC, i.e., the line 7 in the NEW-POP procedure is changed to $S_c = \text{BCRC}(S_o, S_j)$.

Both GAs were run 10 times on each instance of the benchmark set, using double-digit precision. The parameters of the algorithms were set to the values presented in Table 1. The parameter values were selected after preliminary testing, where a wide range of parameter settings was explored seeking a balanced trade-off between fast execution times and solution quality. The hierarchical clustering phase in the $GA_{CB-BCRC}$ was performed using the unweighted pair group method with arithmetic mean (UPGMA) algorithm.

The results obtained are summarized in Table 2. The average travel time, the average number of vehicles, and the average execution time (in seconds) are reported for each data set. The cumulative total travel time, the cumulative total number of vehicles, and the cumulative execution time over the 56 instances are reported at the bottom of the table. Let $x_{CB-BCRC}$ and x_{BCRC} denote the results obtained by the algorithms $GA_{CB-BCRC}$ and GA_{BCRC}, respectively. The performance gap between both algorithms is calculated as follows: $100 \cdot (x_{CB-BCRC} - x_{BCRC})/x_{BCRC}$. A negative gap indicates a performance improvement achieved by the $GA_{CB-BCRC}$ algorithm.

Assessing the solutions in terms of travel time and number of vehicles, it can be seen that both GAs produced pretty similar solutions. The average travel time and average number of vehicles were the same on data sets C1 and C2. On the other data sets, the gaps related to the

Table 1 – GA Parameters

Parameter	Value
Population size, $\mathscr{L}$	100
Maximum number of generations, G_{max}	1000
Crossover rate, R_{rate}	0.9
Minimum number of clusters, C_{min}	2
Maximum number of clusters, C_{max}	25*
Weight used to penalize violations to time-window constraints, α_1	1000
Weight used to penalize violations to vehicle-capacity constraints, α_2	1000
Probability of creating solutions of *kind 1* in the initial population, ρ	0.6

* The fleet size in the considered instances.

Table 2 – Results of experiments: CB-BCRC vs. BCRC

Data set	GA_{BCRC}	$GA_{CB-BCRC}$	Gap (%)
C1			
travel time	828.38	828.38	0.00
number of vehicles	10.00	10.00	0.00
execution time (s)	11.68	3.48	-70.21
C2			
travel time	589.86	589.86	0.00
number of vehicles	3.00	3.00	0.00
execution time (s)	71.98	34.20	-52.49
R1			
travel time	1202.20	1202.62	0.03
number of vehicles	13.59	13.66	0.52
execution time (s)	8.69	4.54	-47.76
R2			
travel time	882.89	882.96	0.01
number of vehicles	5.39	5.43	0.74
execution time (s)	35.03	29.37	-16.16
RC1			
travel time	1374.47	1374.79	0.02
number of vehicles	13.34	13.44	0.75
execution time (s)	8.51	4.18	-50.88
RC2			
travel time	1010.26	1010.24	0.00
number of vehicles	6.30	6.20	-1.59
execution time (s)	29.20	21.42	-26.64
Cumulative			
travel time	55390.35	55398.55	0.01
number of vehicles	493.50	494.70	0.24
execution time (s)	1472.26	887.16	-39.74

average travel time were very close to 0.0%. Regarding the number of vehicles, the gaps were less than 1.0% on all data sets but RC2, where $GA_{CB-BCRC}$ outperformed GA_{BCRC} by a slightly greater margin. Overall, the quality of the solutions produced by both algorithms was practically the same. Although GA_{BCRC} outperformed $GA_{CB-BCRC}$, the gap between the cumulative travel times was 0.01%, practically negligible. The difference in the number of vehicles was 0.24%, which is still very low.

Table 3 and Figure 12 summarize the descriptive statistics of the experiments carried out on all instances. The statistics related to travel time and the number of vehicles indicate that there are no significant differences between the solutions produced by both algorithms. Thus, it is possible to infer that the CB-BCRC operator focuses the search effort on the most promising regions of the search space.

Table 3 – Descriptive statistics: CB-BCRC vs. BCRC

Variable	Algorithm	Statistics							
		Obs	Min	Max	1st Q.	Median	3rd Q.	Mean	St Dv.
Travel time	GA_{BCRC}	560	588.29	1700.96	820.63	947.09	1162.60	989.11	286.21
	$GA_{CB-BCRC}$	560	588.29	1708.06	822.19	942.96	1161.20	989.26	286.31
Numer of vehicles	GA_{BCRC}	560	3.00	20.00	5.00	10.00	12.00	8.81	4.40
	$GA_{CB-BCRC}$	560	3.00	21.00	5.00	10.00	12.00	8.83	4.43
Execution time (sec)	GA_{BCRC}	560	5.80	76.12	9.61	12.22	35.88	26.29	22.08
	$GA_{CB-BCRC}$	560	2.76	54.97	3.97	5.70	28.03	15.84	13.55

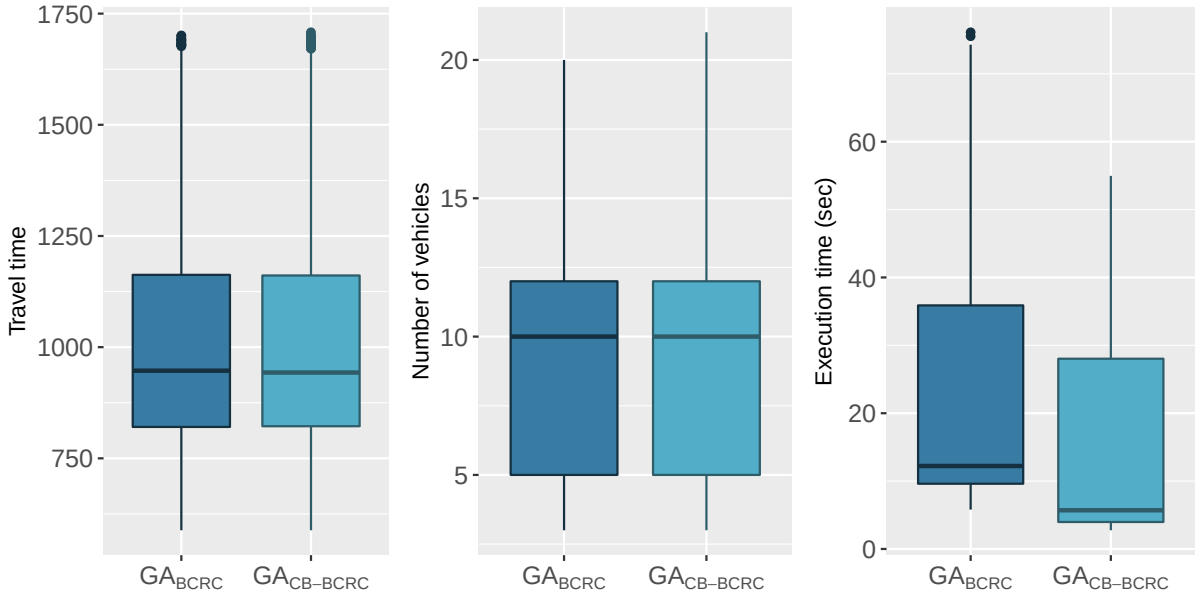


Figure 12 – Box plots: CB-BCRC vs. BCRC.

The descriptive statistics also indicate that $GA_{CB-BCRC}$ required shorter execution times. Analyzing the results class-wise, it is possible to observe that the execution times of the algorithm $GA_{CB-BCRC}$ were lower for all data sets. Overall, the execution time was reduced by 39.74%. For the data set C1, the reduction in the execution time even achieved 70.21%. It is also possible to observe that $GA_{CB-BCRC}$ performed better on instances with clustered customers or with narrow time windows (i.e. instances from data sets C1, C2, R1, and RC1). On instances with wide time windows (i.e. instances from data sets R2 and RC2), the performance of $GA_{CB-BCRC}$ decreased.

Statistical tests were conducted to confirm the similarity observed between the solutions produced by both algorithms and the differences observed in the execution times. The Shapiro–Wilk test was applied to test the normality of the results. For both algorithms, the distributions of travel time, number of vehicles, and execution time did not conform to normality, since all p-values were less than 0.05. Thus, the results were compared by applying the Mann–Whitney U test. At a 95% confidence level, the Mann–Whitney U tests indicated that there is no statistically significant difference between the results of both algorithms in terms of travel time (p-value = $1.0 > 0.05$) and number of vehicles (p-value = $0.99 > 0.05$). However, there is a statistically significant difference between the execution times of the algorithms (p-value = $5.6 \times 10^{-27} < 0.05$).

The experimental results offer evidence that the CB-BCRC can produce solutions of similar quality to those produced by the BCRC, while requiring shorter execution times. In the experiments conducted, the average execution time of the GA was reduced by 39.74%.

3.3.2 Performance of the proposed MA

A set of experiments was conducted to establish whether the solutions produced by the proposed MA are competitive with the best-known solutions of the literature. The focus is now on assessing the solutions in terms of travel time.

The MA parameters were set to the values presented in Table 4. These values were selected after preliminary testing, where a wide range of parameter settings was explored, seeking a balanced trade-off between fast execution times and solution quality. The hierarchical clustering was performed using the UPGMA algorithm.

Each instance of the Solomon's benchmark set was solved 10 times, using double-digit precision. The solutions obtained were compared with the results of other heuristics. Since the objective of the proposed MA is the minimization of total travel time, the heuristics selected for comparison were those whose main objective is to minimize the total travel distance or total travel time (travel times are equal to their corresponding distances in Solomon's instances). The heuristics selected were those of Jung and Moon (2002) (JU), Alvarenga, Mateus and Tomi (2007) (AL), Oliveira and Vasconcelos (2010) (OL), Ursani *et al.* (2011) (UR), Garcia-Najera and Bullinaria (2011) (GN), and Vidal *et al.* (2015) (VI). The heuristic of Garcia-Najera and Bullinaria (2011) is indeed a multiobjective GA; nevertheless, a comparison against this heuristic is fair, since the authors reported the solutions with the best travel distances that their algorithm found. It should also be noted that Vidal *et al.* (2015) conducted experiments with two heuristics (HGA and MS-ILS) using different relaxation schemes, and reported the best solutions found during all experiments.

Table 5 summarizes the results for each Solomon data set following the format commonly adopted in the VRPTW literature. Considering the best solutions found by the heuristics after being run several times on each instance, two values are reported for each data set. The upper value indicates the average travel distance (or travel time), and the lower value indicates the average number of vehicles. The values CTC and CNV indicate the cumulative total travel distance (or travel time) and the cumulative number of vehicles, respectively (Vidal *et al.* (2015)

Table 4 – MA Parameters

Parameter	Value
Population size, $\mathscr{L}$	100
Maximum number of generations, G_{max}	1000
Crossover rate, R_{rate}	0.9
Local search rate, L_{rate}	0.03
Number of iterations of the inter-route neighborhood search, I_{max}	50
Initial temperature of the intra-route neighborhood search, γ	100000
Cooling rate of the intra-route neighborhood search, β	0.001
Minimum number of clusters, C_{min}	2
Maximum number of clusters, C_{max}	25*
Weight used to penalize violations to time-window constraints, α_1	1000
Weight used to penalize violations to vehicle-capacity constraints, α_2	1000
Probability of creating solutions of *kind 1* in the initial population, ρ	0.6

* The fleet size in the considered instances.

Table 5 – Comparative results: best solutions for each data set

Data Set	JU	AL	OL	UR	GN	VI	Proposed MA		
							Best	Average	Gap (%)
C1	828.38	828.38	828.38	828.38	828.38	828.38	828.38	828.38	0.00
	10.00	10.00	10.00	10.00	10.00	-	10.00	10.00	
C2	589.86	589.86	589.86	589.86	589.86	589.86	589.86	589.86	0.00
	3.00	3.00	3.00	3.00	3.00	-	3.00	3.00	
R1	1179.95	1183.38	1186.94	1188.85	1187.32	1178.98	1184.95	1191.39	0.51
	13.25	13.25	13.33	13.5	13.08	-	13.42	13.54	
R2	878.41	899.90	878.79	885.78	897.95	877.2	878.74	881.44	0.18
	5.36	5.55	5.36	5.55	4.00	-	5.36	5.35	
RC1	1343.65	1341.67	1362.44	1360.96	1348.22	1338.18	1351.43	1364.25	0.99
	13.00	12.88	13.25	13.25	12.63	-	13.13	13.26	
RC2	1004.21	1015.90	1004.59	1013.29	1036.65	1003.95	1004.77	1008.01	0.08
	6.25	6.50	6.13	6.88	5.38	-	6.25	6.20	
CTC	54779.02	55134.27	55020.00	55137.04	55378.61	54708	54909.37	55144.87	0.37
CNV	486	489	488	498	459	-	489	491	
runs	100	3	15	30	30	82	10	10	

JU: Jung and Moon (2002) AL: Alvarenga, Mateus and Tomi (2007) OL: Oliveira and Vasconcelos (2010)
UR: Ursani *et al.* (2011) GN: Garcia-Najera and Bullinaria (2011) VI: Vidal *et al.* (2015)

did not report the information related to the number of vehicles of their solutions). The number of runs is reported in the last row. For the proposed MA, the values corresponding to the average solution quality are reported as well. The cost gaps between the best results of the cited heuristics and the best results obtained by the proposed MA are reported in the last column of the table. Let x_{tt} be the best travel cost achieved by the cited heuristics and let x^*_{tt} be the best travel cost achieved by the proposed MA. The cost gap is given by $100 \cdot (x^*_{tt} - x_{tt})/x_{tt}$.

The heuristics of Vidal *et al.* (2015) produced the best-known results for the Solomon's instances regarding distance minimization. These heuristics achieved the shortest travel distance on all data sets and the shortest cumulative travel distance. The overall performance of the proposed MA can be assessed by checking the CTC value corresponding to the best solutions. It is possible to see that the proposed MA outperformed four out of six cited heuristics, whereas the gap with respect to the results of Vidal *et al.* (2015) was 0.37%.

Analyzing the results category-wise, it can be seen that the proposed MA achieved the best solution quality on instances of type C. Remarkable results were obtained on instances RC2. In this case, the gap with respect to the best results was less than 0.1%. On the other data sets, the gaps were a bit greater; in any case, the gaps with respect to the best results were less than 1.0%. These results show that the proposed MA is able to produce high-quality solutions, which are competitive with those produced by the other heuristics.

The results obtained for all the instances are presented in Table 6. Two values are reported for each solution: the total travel cost in terms of distance or time (TC) and the number of vehicles (NV). The best published solutions are listed with the references to the papers that recorded them. Most of the best-known solutions have been found by some of the heuristics cited previously; the remaining best-known solutions were found by the heuristics of Rochat and Taillard (1995) (RO), Chiang and Russell (1997) (CH), Tan *et al.* (2001) (TA), and Cordeau, Laporte and Mercier

Table 6 – Results for each instance

Instance	Other Heuristics		Proposed MA					
	Best Solution		Best Solution		Average	St Dv.	CV	Avg. time (s)
	TC | NV	Ref.	TC | NV	Gap (%)	TC | NV	TC | NV	TC | NV	
C101	828.94 | 10	JU	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	19.82
C102	828.94 | 10	JU	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	21.87
C103	828.06 | 10	JU	**828.06 | 10**	**0.00**	828.06 | 10.00	0.00 | 0.00	0.00 | 0.00	23.85
C104	824.78 | 10	JU	**824.78 | 10**	**0.00**	824.78 | 10.00	0.00 | 0.00	0.00 | 0.00	24.93
C105	828.94 | 10	JU	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	20.88
C106	828.94 | 10	JU	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	21.09
C107	828.94 | 10	JU	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	21.26
C108	828.94 | 10	JU	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	22.37
C109	828.94 | 10	JU	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	23.64
C201	591.56 | 3	JU	**591.56 | 3**	**0.00**	591.56 | 3.00	0.00 | 0.00	0.00 | 0.00	38.10
C202	591.56 | 3	JU	**591.56 | 3**	**0.00**	591.56 | 3.00	0.00 | 0.00	0.00 | 0.00	39.14
C203	591.17 | 3	JU	**591.17 | 3**	**0.00**	591.17 | 3.00	0.00 | 0.00	0.00 | 0.00	38.92
C204	590.6 | 3	JU	**590.6 | 3**	**0.00**	590.60 | 3.00	0.00 | 0.00	0.00 | 0.00	38.13
C205	588.88 | 3	JU	**588.88 | 3**	**0.00**	588.88 | 3.00	0.00 | 0.00	0.00 | 0.00	38.09
C206	588.49 | 3	JU	**588.49 | 3**	**0.00**	588.49 | 3.00	0.00 | 0.00	0.00 | 0.00	38.31
C207	588.29 | 3	JU	**588.29 | 3**	**0.00**	588.29 | 3.00	0.00 | 0.00	0.00 | 0.00	38.67
C208	588.32 | 3	JU	**588.32 | 3**	**0.00**	588.32 | 3.00	0.00 | 0.00	0.00 | 0.00	37.78
R101	1642.88 | 20	JU	**1642.88 | 20**	**0.00**	1645.51 | 20.00	3.02 | 0.00	0.18 | 0.00	30.04
R102	1472.62 | 18	AL	1473.62 | 18	0.07	1478.39 | 18.00	2.67 | 0.00	0.18 | 0.00	28.65
R103	1213.62 | 14	AL	1217.46 | 14	0.32	1223.28 | 14.90	2.42 | 0.32	0.20 | 2.12	28.03
R104	976.61 | 11	JU	985.17 | 11	0.88	992.44 | 10.80	4.89 | 0.42	0.49 | 3.90	27.06
R105	1360.78 | 15	JU	1365.66 | 15	0.36	1371.97 | 15.70	2.85 | 0.48	0.21 | 3.08	28.01
R106	1239.37 | -	VI	1245.35 | 13	0.48	1250.83 | 13.10	4.57 | 0.32	0.37 | 2.41	28.09
R107	1072.12 | -	VI	1081.7 | 12	0.89	1089.60 | 11.70	5.07 | 0.48	0.47 | 4.13	27.45
R108	938.2 | -	VI	951.24 | 10	1.39	953.87 | 10.40	2.14 | 0.52	0.22 | 4.97	27.22
R109	1013.16 | 12	CH	1155.46 | 13	14.05	1161.32 | 13.00	5.00 | 0.00	0.43 | 0.00	27.60
R110	1072.41 | 12	JU	1082.47 | 12	0.94	1092.04 | 12.00	5.04 | 0.00	0.46 | 0.00	26.81
R111	1053.5 | 12	JU	1055.6 | 12	0.20	1063.82 | 12.00	3.95 | 0.00	0.37 | 0.00	27.18
R112	953.63 | 10	JU	962.82 | 11	0.96	973.55 | 10.90	5.27 | 0.32	0.54 | 2.90	26.46
R201	1147.8 | 8	OL	1148.09 | 8	0.03	1149.68 | 8.30	1.17 | 0.48	0.10 | 5.82	31.07
R202	1034.35 | 8	JU	1034.82 | 8	0.05	1040.16 | 6.90	2.85 | 0.74	0.27 | 10.69	32.72
R203	874.87 | 6	JU	875.12 | 6	0.03	876.57 | 6.00	1.29 | 0.00	0.15 | 0.00	34.85
R204	735.8 | 5	OL	736.27 | 5	0.06	737.61 | 4.60	1.38 | 0.52	0.19 | 11.23	37.31
R205	954.16 | 5	OL	**954.16 | 5**	**0.00**	958.09 | 5.30	3.05 | 0.48	0.32 | 9.11	33.85
R206	879.89 | 5	JU	**879.89 | 5**	**0.00**	883.06 | 5.00	2.75 | 0.00	0.31 | 0.00	35.94
R207	797.99 | 4	OL	800.83 | 4	0.36	803.06 | 4.00	1.88 | 0.00	0.23 | 0.00	38.47
R208	705.33 | -	VI	707.58 | 3	0.32	711.39 | 3.50	2.45 | 0.53	0.34 | 15.06	41.84
R209	859.39 | 5	JU	**859.39 | 5**	**0.00**	861.02 | 5.00	1.90 | 0.00	0.22 | 0.00	33.85
R210	904.78 | -	VI	910.83 | 6	0.67	913.54 | 6.20	1.99 | 0.63	0.22 | 10.20	33.38
R211	753.15 | -	VI	759.14 | 4	0.80	761.62 | 4.00	2.15 | 0.00	0.28 | 0.00	35.12
RC101	1623.58 | 15	RO	1646.53 | 16	1.41	1657.52 | 16.30	6.46 | 0.48	0.39 | 2.96	26.14
RC102	1461.23 | 14	JU	1480.46 | 15	1.32	1487.73 | 15.00	6.06 | 0.00	0.41 | 0.00	27.36
RC103	1249.86 | 11	TA	1279.15 | 12	2.34	1292.62 | 12.00	11.83 | 0.00	0.91 | 0.00	27.32
RC104	1135.48 | 10	CO	1139.14 | 10	0.32	1154.03 | 10.40	6.49 | 0.52	0.56 | 4.97	26.82
RC105	1518.58 | 16	JU	1542.37 | 16	1.57	1552.54 | 15.80	5.69 | 0.63	0.37 | 4.00	27.01
RC106	1376.26 | -	VI	1383.85 | 13	0.55	1404.15 | 13.60	11.87 | 0.52	0.85 | 3.80	27.01
RC107	1211.11 | -	VI	1216.65 | 12	0.46	1232.31 | 12.00	8.78 | 0.00	0.71 | 0.00	26.93
RC108	1117.53 | 11	AL	1123.26 | 11	0.51	1133.14 | 11.00	4.33 | 0.00	0.38 | 0.00	26.52
RC201	1265.56 | 9	JU	1269.07 | 9	0.28	1273.45 | 9.10	3.38 | 0.88	0.27 | 9.62	29.89
RC202	1095.64 | 8	JU	**1095.64 | 8**	**0.00**	1098.73 | 7.90	2.23 | 0.32	0.20 | 4.00	31.62
RC203	926.82 | -	VI	**926.82 | 5**	**0.00**	933.13 | 5.00	4.18 | 0.00	0.45 | 0.00	33.22
RC204	786.38 | 4	OL	**786.38 | 4**	**0.00**	787.70 | 4.00	1.31 | 0.00	0.17 | 0.00	35.84
RC205	1157.55 | 7	OL	**1157.55 | 7**	**0.00**	1158.98 | 7.00	1.52 | 0.00	0.13 | 0.00	29.28
RC206	1054.61 | 7	JU	1057.65 | 7	0.29	1062.57 | 6.10	2.79 | 0.74	0.26 | 12.10	31.97
RC207	966.08 | 6	JU	**966.08 | 6**	**0.00**	966.91 | 6.00	1.65 | 0.00	0.17 | 0.00	31.08
RC208	778.93 | -	VI	**778.93 | 4**	**0.00**	782.65 | 4.50	2.24 | 0.53	0.29 | 11.71	32.61

RO: Rochat and Taillard (1995) CH: Chiang and Russell (1997) TA: Tan *et al.* (2001)
CO: Cordeau, Laporte and Mercier (2001) JU: Jung and Moon (2002) AL: Alvarenga, Mateus and Tomi (2007)
OL: Oliveira and Vasconcelos (2010) VI: Vidal *et al.* (2015)

(2001) (CO). The best solutions found by the proposed MA are reported along with the travel cost gaps attained with respect to the best published solutions. The average, standard deviation, and coefficient of variation of the TC and NV values gathered during 10 runs are reported to provide further details on the MA performance. The average running time (in seconds) for each instance is reported as well. The proposed MA found the best-known solution for 27 out of 56 instances (marked in bold), including all instances of type C and six out of eight instances of the data set RC2. For most of the other instances, the cost gaps were less than 1.0%. Overall, a solution with a cost gap less than 1.0% with respect to the best-known solution was found for 50 out of 56 instances.

The values of the coefficients of variation associated with travel time were less than 1.0% for all instances, which suggests that the proposed MA performed consistently in all experiments. Statistical tests were carried out to confirm this fact. Since the results did not conform to normality (Shapiro-Wilk test p-value < 0.05), the Kruskal-Wallis test was employed to compare the travel-time distributions obtained from all experiments. At a 95% confidence level, the Kruskal-Wallis test indicated that the travel-time distributions did not significantly differ across experiments (p-value $= 1.0 > 0.05$). Thus, it is possible to conclude that the proposed MA consistently found solutions of similar quality.

3.3.3 Impact of the local search on solution quality

Comparing the average and cumulative travel times attained by the proposed MA (presented in Table 5) with the average and cumulative travel times attained by the $GA_{CB-BCRC}$ algorithm (presented Table 2), it is possible to observe that there was a quality improvement in all cases, except for data sets C, where the results obtained without applying the local search procedure were the same. Even though quality improvements were obtained by applying the local search operator, it seems that the local search has a low impact on solution quality, since the quality difference between the solutions obtained by the $GA_{CB-BCRC}$ algorithm and the proposed MA is not substantial. The improvements were more noticeable on data sets R1 and RC1, whereas on data sets R2 and RC2 the improvements were less evident. This suggests that the CB-BCRC operator is the main factor responsible for the quality of the solutions produced by the proposed MA.

Table 7 and Figure 13 present the descriptive statistics of the results obtained using the local search procedure (MA) and omitting it ($GA_{CB-BCRC}$). As can be seen, the results are fairly similar. Statistical tests were conducted to confirm whether the local search significantly influences the quality of the solutions. The solutions were compared by applying the Mann–Whitney U test. Since several statistical tests were performed, a false discovery rate correction (FDR) procedure was performed on the p-values. The tests were performed on the solutions grouped by data set and on the aggregate results. At a 95% confidence level, the Mann–Whitney U tests indicated that there is no statistically significant difference between the solutions produced by

Table 7 – Descriptive statistics: impact of the local search on solution quality

Data Set	Algorithm	Statistics							
		Obs	Min	Max	1st Q.	Median	3rd Q.	Mean	St Dv.
C1	$GA_{CB-BCRC}$	90	824.78	828.94	828.94	828.94	828.94	828.38	1.31
	MA	90	824.78	828.94	828.94	828.94	828.94	828.38	1.31
C2	$GA_{CB-BCRC}$	80	588.29	591.56	588.45	589.74	591.27	589.86	1.41
	MA	80	588.29	591.56	588.45	589.74	591.27	589.86	1.41
R1	$GA_{CB-BCRC}$	120	946.42	1708.06	1046.38	1130.83	1298.89	1202.62	217.14
	MA	120	951.24	1652.02	1042.15	1127.44	1286.58	1191.39	206.37
R2	$GA_{CB-BCRC}$	110	707.88	1155.76	762.67	876.45	958.91	882.96	127.14
	MA	110	707.58	1151.21	764.07	876.30	955.60	881.44	126.58
RC1	$GA_{CB-BCRC}$	80	1134.82	1680.39	1224.20	1364.24	1516.59	1374.79	181.52
	MA	80	1123.26	1665.88	1203.27	1350.37	1510.23	1364.25	181.04
RC2	$GA_{CB-BCRC}$	80	780.72	1280.57	893.05	1015.96	1116.47	1010.24	164.05
	MA	80	778.93	1279.19	892.54	1014.21	1117.18	1008.01	163.48
All instances	$GA_{CB-BCRC}$	560	588.29	1708.06	822.19	942.96	1161.20	989.26	286.31
	MA	560	588.29	1665.88	820.32	944.55	1156.92	984.73	280.73

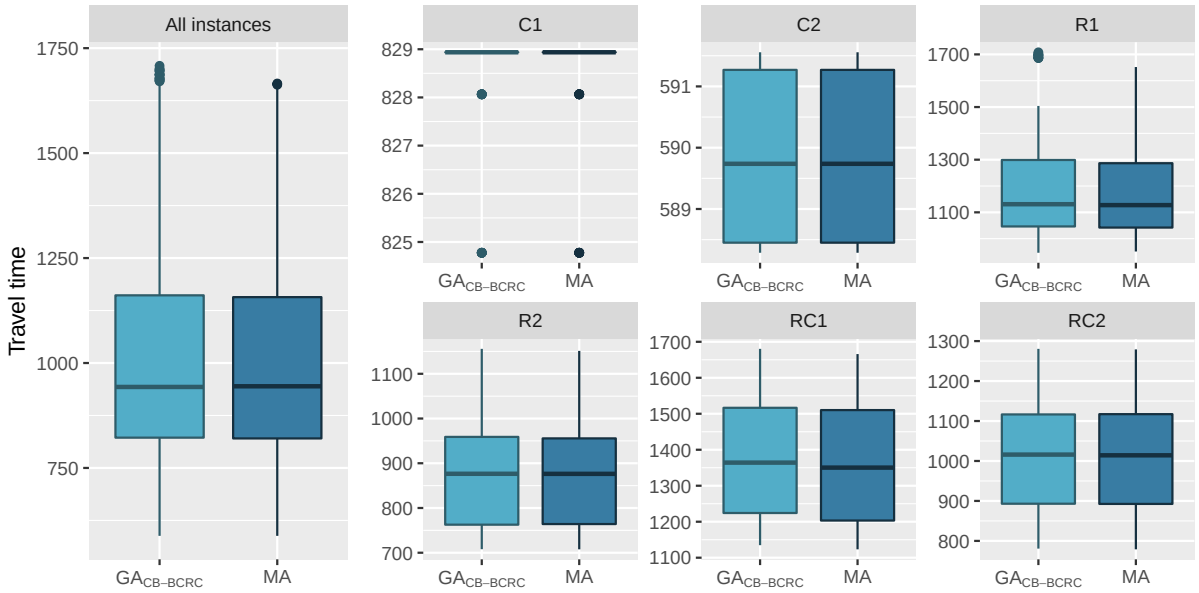


Figure 13 – Box plots: impact of the local search on solution quality.

the $GA_{CB-BCRC}$ algorithm and the proposed MA. All the p-values obtained were greater than 0.05.

From the statistical analysis of the results, it is possible to conclude that the refinement performed by the local search procedure slightly improves the quality of the solutions produced by the crossover operator alone. This allows the proposed MA to produce solutions more competitive with the best-known solutions of the literature. Nevertheless, when it comes to solution quality, the performance of the proposed MA relies on the crossover operator.

3.4 Conclusions

This chapter presented a simple yet effective MA for solving the VRPTW. Two search procedures were proposed: the CB-BCRC and the CB-Inter-Route-NS. When the initial population of solutions is being created, a clustering procedure is applied to the customers' spatial data. The output of this procedure is used by the search operators to identify which routes pass close to each other. New solutions are created by relocating customers between the routes identified, seeking to minimize the detour costs associated with the modifications.

Computational experiments on the Solomon's benchmark set show that the CB-BCRC operator can produce solutions similar to those created by the BCRC operator, and it requires lower execution times. The results also show that the proposed MA can produce high-quality solutions, which are competitive with those produced by other heuristics. The proposed MA outperformed four out of six solution approaches considered for comparison (when assessing the results in terms of the travel cost obtained over all instances), and it found the best-known solutions for several instances. Although the local search procedure slightly improves the quality of the solutions produced by the CB-BCRC operator, the latter is the main factor responsible for the quality of the produced solutions.

This chapter presented a comparison between the CB-BCRC and the BCRC. The performance comparison of the CB-BCRC and other crossover operators and the performance comparison of the CB-Inter-Route-NS and other local search approaches proposed for the VRPTW remain topics for further research.

Further research could assess the performance of the proposed MA in solving other variants of the VRP and related problems. The development of new clustering approaches to guide the search is another possible direction of further research.

CHAPTER

4

THE DYNAMIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS AND STOCHASTIC CUSTOMERS

This chapter addresses the dynamic vehicle routing problem with time windows and stochastic customers (DVRPTWSC) seeking to minimize the total travel time and maximize the number of dynamic requests served. A cartographic approach to the DVRPTWSC is presented. The proposed approach is incorporated into a multiagent system that considers three kinds of agents: dispatcher, vehicle, and customer. The dispatcher is responsible for receiving service requests from the customers and planning the routes for the vehicles. The first action the dispatcher takes is to perform a cartographic processing of the customers' spatial data. This process uses hierarchical clustering to divide the region where customers are located into a hierarchy of nested regions. Each customer is associated with a hierarchy of nested regions that contains the regions where the customer is situated.

A slightly modified version of the memetic algorithm (MA) presented in section 3.2 is used to create the initial routing plan. The MA optimizes a pool of routing scenarios that contain all known requests and potential requests sampled from known probability distributions. The MA uses the output of the hierarchical clustering procedure to perform the search. The initial routing plan is selected from the pool of routing scenarios by using a similarity measure.

Over the planning horizon, the dispatcher updates the routing plan by removing the potential requests that do not materialize and by adding new requests. The assignation of requests based on nested regions (ARNR) is used to process new requests. The ARNR procedure aims to assign new requests among the vehicles that currently are or will be near the customer requiring service, since those vehicles are more likely to serve the request at a low detour cost. The hierarchy of nested regions associated with the customer requiring service is used to identify a set of candidate vehicles to perform the service.

Experiments conducted on instances generated from the Solomon's (1987) benchmark set show that (1) the proposed MA significantly improves the quality of the routing scenarios created by a construction heuristic; (2) the exploitation of stochastic information about potential requests significantly improves the quality of the solutions; (3) the ARNR procedure attains results similar to those produced by an exhaustive search-based approach, which considers all active vehicles as candidates for serving new requests; (4) the ARNR procedure significantly reduces the number of vehicles considered for serving new requests in comparison with the exhaustive search-based approach; and (5) the proposed approach performs consistently under three levels of dynamism: low, medium, and high.

The remainder of this chapter is organized as follows: Section 4.1 presents the definition of the DVRPTWSC. Section 4.2 describes the solution approach. Section 4.3 gives details on the implementation of the solution approach. Section 4.4 presents the experimental results. Section 4.5 concludes the chapter.

4.1 Problem Definition

The DVRPTWSC consists of defining a decision-making process to deliver goods (in this case, a unique product) from a single depot to a set of n geographically scattered customers. The customers are labeled by numbers $1,\ldots,n$ and the depot is labeled by number 0 (the depot is considered as a special customer). Each customer i is located at coordinates (x_i, y_i) and is associated with a demand q_i, a service time s_i, and a time window $[e_i, l_i]$. The depot has neither demand nor service time, i.e., $q_0 = s_0 = 0$. The planning horizon is given by the depot's time window $[e_0, l_0]$, where e_0 and l_0 represent the earliest possible departure time from the depot and the latest possible arrival time at the depot, respectively.

The customers are classified as *static customers* or *dynamic customers* depending on the moment they make their service requests. Static customers make their requests in advance, i.e., they are known before e_0. Dynamic customers may or may not make requests over the planning horizon. It is assumed that the probability that a dynamic customer makes a request is known (this probability could be estimated based on historical records of previous requests). When a dynamic customer j makes a service request, the arrival time of the request, R_j, is represented by a continuous random variable that takes values in the interval $[e_0, e_j)$. This means that dynamic customers can only make service requests before the start of their time windows.

A fleet of homogeneous vehicles $v = 1,\ldots,\mathrm{M}$, each one with capacity Q, is available for serving the customers. The vehicles depart from the depot, visit a set of assigned customers and return to the depot before the end of the planning horizon. A vehicle can only traverse one route over the planning horizon. Customers can make at most one service request over the planning horizon.

It is assumed that a real-time dispatching system is available to allow the dispatcher to

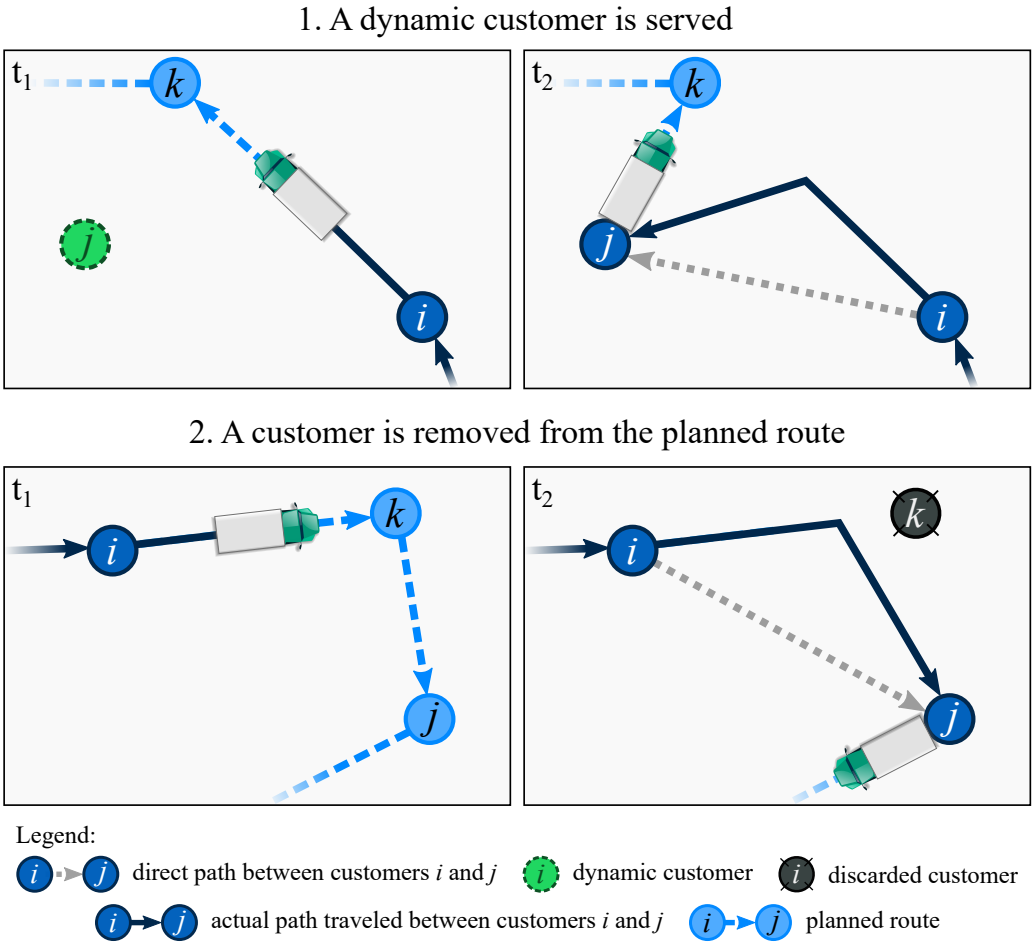


Figure 14 – Examples of possible route diversions.

instantly locate each vehicle and check its updated information, including served customers, remaining customers, elapsed travel time, and remaining capacity. Furthermore, a real-time communication system allows the dispatcher to instruct vehicles about changes on their routes when necessary. When a dynamic request becomes known, the dispatcher must decide whether to accept it or reject it. Dynamic requests that cannot be feasibly served by any vehicle are rejected.

At any moment during the planning horizon, the dispatcher can order the vehicles to perform changes to their current routes by either adding or removing customers. If the dispatcher requires it and route feasibility holds, a vehicle can divert to serve customer j while en route to customer i; in this case, the service of customer i is deferred. Similarly, while a vehicle is en route to customer i, the dispatcher can order to skip the service of that customer; in that case, the vehicle diverts to the next customer on its route.

The vehicles travel at the same constant speed and measure their travel times. Whenever possible, they travel in a straight line between two customers. Nevertheless, the travel time between two customers can vary due to possible route diversions, as illustrated in Figure 14. In Figure 14.1, the vehicle is heading toward customer k when the dynamic customer j makes a request at time t_1. The dispatcher instructs the vehicle to serve customer j first and postpone the

service at customer k. In Figure 14.2, the vehicle is heading toward customer k at time t_1 when the dispatcher instructs it to skip that service and continue to customer j. In both examples, the time it took for the vehicle to travel between the customers was different from the travel time the vehicle would have needed if it would have traveled in a straight line.

For every vehicle v that arrives at the depot after completing its journey, the sequence of served customers is given by $u^v = \langle u_0^v, u_1^v, \ldots, u_{n_v}^v, u_{n_v+1}^v \rangle$, where $u_0^v = u_{n_v+1}^v = 0$ since every route starts and ends at the depot. All u_k^v must be distinct for $1 \leq k \leq n_v$, since customers are visited at most once. The set of customers served by vehicle v is $\mathscr{U}^v = \{u_k^v | 1 \leq k \leq n_v\}$. Let $\tau_{u_i^v, u_j^v}$ be the time it took for vehicle v to travel between customers u_i^v and u_j^v. The cumulative travel time of vehicle v to the kth customer on its route is given by

$$\tau_{v,k}^+ = \begin{cases} 0 & \text{if } k = 0 \\ \tau_{v,k-1}^+ + \tau_{u_{k-1}^v, u_k^v} & \text{if } k > 0. \end{cases} \tag{4.1}$$

The total travel time taken by vehicle v to complete its route is $ftt_v = \tau_{v,n_v+1}^+$.

The accumulated load delivered by vehicle v after serving the kth customer on its route is given by

$$q_{v,k}^+ = \begin{cases} 0 & \text{if } k = 0 \\ q_{v,k-1}^+ + q_{u_k^v} & \text{if } k > 0. \end{cases} \tag{4.2}$$

The total load delivered by vehicle v after finishing its journey is $fdl_v = q_{v,n_v}^+$. Vehicle v's route is feasible regarding the capacity constraint if $fdl_v \leq \mathrm{Q}$.

The vehicles record the times at which they start serving each customer and the time they arrive back at the depot. Let $\mathscr{T}_{u_k^v}$ be the time at which vehicle v started serving the kth customer on its route, and let $\mathscr{T}_{u_{n_v+1}^v}$ be the time at which vehicle v arrived at the depot after completing its journey. Vehicle v's route is feasible regarding the time-window constraints if (1) all customers were served within their time windows, i.e., $e_{u_k^v} \leq \mathscr{T}_{u_k^v} \leq l_{u_k^v}$ for $k = 1, \ldots, n_v$ (vehicles may have to wait until the beginning of a customer's time window to start the service); and (2) vehicle v arrived at the depot by the end of the planning horizon, i.e., $\mathscr{T}_{u_{n_v+1}^v} \leq l_{u_{n_v+1}^v} = l_0$.

The results of the decisions made by the dispatcher are evaluated once all vehicles that were assigned a route have returned to the depot after serving their assigned customers. Let SC be the set of static customers and let AC be the set of dynamic customers that made requests during the planning horizon. Let $U_V = \{1, \ldots, m'\}$, with $m' \leq \mathrm{M}$, be the set of all vehicles that traversed a route and returned to the depot. The objective of the problem is to define *a set of actions* that allows the dispatcher to assign customers to the vehicles, in such a way that the final

executed routing plan conforms to the following criteria:

$$min \sum_{v \in U_V} ftt_v \qquad \text{(primary objective)} \tag{4.3}$$

$$max \left| AC \cap \bigcup_{v \in U_V} \mathscr{U}^v \right| \qquad \text{(secondary objective)} \tag{4.4}$$

subject to

$$SC \subseteq \bigcup_{v \in U_V} \mathscr{U}^v \tag{4.5}$$

$$\mathscr{U}^v \cap \mathscr{U}^{v'} = \emptyset \qquad 1 \leq v < v' \leq m' \tag{4.6}$$

$$u_k^v \neq u_{k'}^v \qquad 1 \leq k < k' \leq n_v, \forall v \in U_V \tag{4.7}$$

$$u_k^v = 0 \qquad k = 0, k = n_v + 1, \forall v \in U_V \tag{4.8}$$

$$e_{u_k^v} \leq \mathscr{T}_{u_k^v} \leq l_{u_k^v} \qquad 1 \leq k \leq n_v, \forall v \in U_V \tag{4.9}$$

$$\mathscr{T}_{u_k^v} \leq l_{u_k^v} \qquad k = n_v + 1, \forall v \in U_V \tag{4.10}$$

$$fdl_v \leq \mathrm{Q} \qquad \forall v \in U_V. \tag{4.11}$$

The primary objective (4.3) is to minimize the total travel time. The secondary objective (4.4) is to maximize the number of dynamic requests served. Constraint (4.5) ensures that all static requests are served. Constraints (4.6) and (4.7) ensure that customers are served at most once. Constraints (4.8) ensure that all vehicles start and end their routes at the depot. Constraints (4.9) guarantee that customers are served within their time windows. Constraints (4.10) guarantee that all vehicles arrive at the depot by the end of the planning horizon. Constraints (4.11) state that no vehicle can be overloaded.

4.2 Solution Approach

A multiagent system was developed to address the DVRPTWSC. The model considers three kinds of agents: dispatcher, vehicle, and customer. The dispatcher receives service requests from the customers and plans the routes for the vehicles. The vehicles just obey the dispatcher's instructions and update their routing information. The approach used by the dispatcher to plan the routes consists of two stages. In the first stage, the geographical area where the customers are located is divided into several regions, each of them clustering together close customers. Additionally, an initial routing plan, which contains the static customers and some potential dynamic customers is created. The second stage starts at the same time as the planning horizon starts. In this stage, the dispatcher updates the initial routing plan according to the occurrence of dynamic events. The geographical regions previously defined are considered by the dispatcher to take some decisions while the routing plan is being executed.

4.2.1 Cartographic processing

The first step of the solution approach consists of dividing the given geographical region into a *hierarchy of nested subregions*. The spatial division obtained by this process resembles the hierarchy of administrative divisions of geographical areas (e.g., counties, states, and countries).

Skupin (2002) described a process to create scale-dependent visualizations of non-geographic information. The hierarchical division is performed following a similar process: First, a subregion is defined for each customer, including the depot. This step is accomplished by constructing the Euclidean Voronoi diagram of the customers' locations. Alternatively, an ad hoc geographical division could be performed if real geographic data are available. For example, an urban area could be split into several subregions by using blocks and streets as guides. Once the regions for the customers have been defined, the next step consists of creating larger regions, clustering close customers. Hierarchical agglomerative clustering (HAC) is performed on the matrix of Euclidean distances between customers, without taking into account the depot (for further details on HAC, the reader is referred to Xu and Wunsch (2008)). The dendrogram obtained from the clustering process is cut at different points; as the height of the cuts increases, larger clusters are obtained. The regions associated with the customers belonging to a given cluster are merged to obtain a larger region. The cutting process is repeated until a clustering arrangement composed of two clusters is obtained.

A region is defined by a *list of customers* and a *polygon* enclosing the area inside which the customers are located. Let θ_k denote the region k. At the lowest level of aggregation, the geographical area is divided into regions $\theta_1, \ldots, \theta_n$; each customer $1, \ldots, n$ belongs to one region. As the dendrogram is cut at higher heights, larger regions are obtained. At the highest level of aggregation, the geographical area is divided into two regions: θ_{z-1} and θ_z.

The creation of a hierarchy of nested subregions is illustrated in Figure 15. Given a DVRPTWSC instance, HAC is applied to the spatial data of all customers, both static and dynamic. Seven geographical divisions are obtained depending on the point where the dendrogram is cut. At the lowest level of aggregation, the geographical area is divided into regions $\theta_1, \ldots, \theta_{10}$; each customer belongs to one region. As the dendrogram is cut at higher heights, regions containing more than one customer are obtained. At the highest level of aggregation, the geographical area is divided into regions θ_{17}, which contains customers 2, 3, 4, 6, and 9; and θ_{18}, which contains customers 1, 5, 7, 8, and 10.

The dendrogram obtained from the clustering process, D, is saved to be used later. The customers are assigned a sequence containing all the regions where they are located. In such sequence, the regions are sorted from smallest to largest. Let Θ^i denote the sequence of regions associated with customer i. Then, $\Theta^i = \langle \Theta^i_1, \ldots, \Theta^i_{z_i} \rangle = \langle \theta_i, \ldots, \theta_{\wp} \rangle$, where θ_i is the region containing only customer i and $\theta_{\wp}$ is the largest region where customer i belongs. In the example presented in Figure 15, the sequence of regions where customer 1 belongs would be

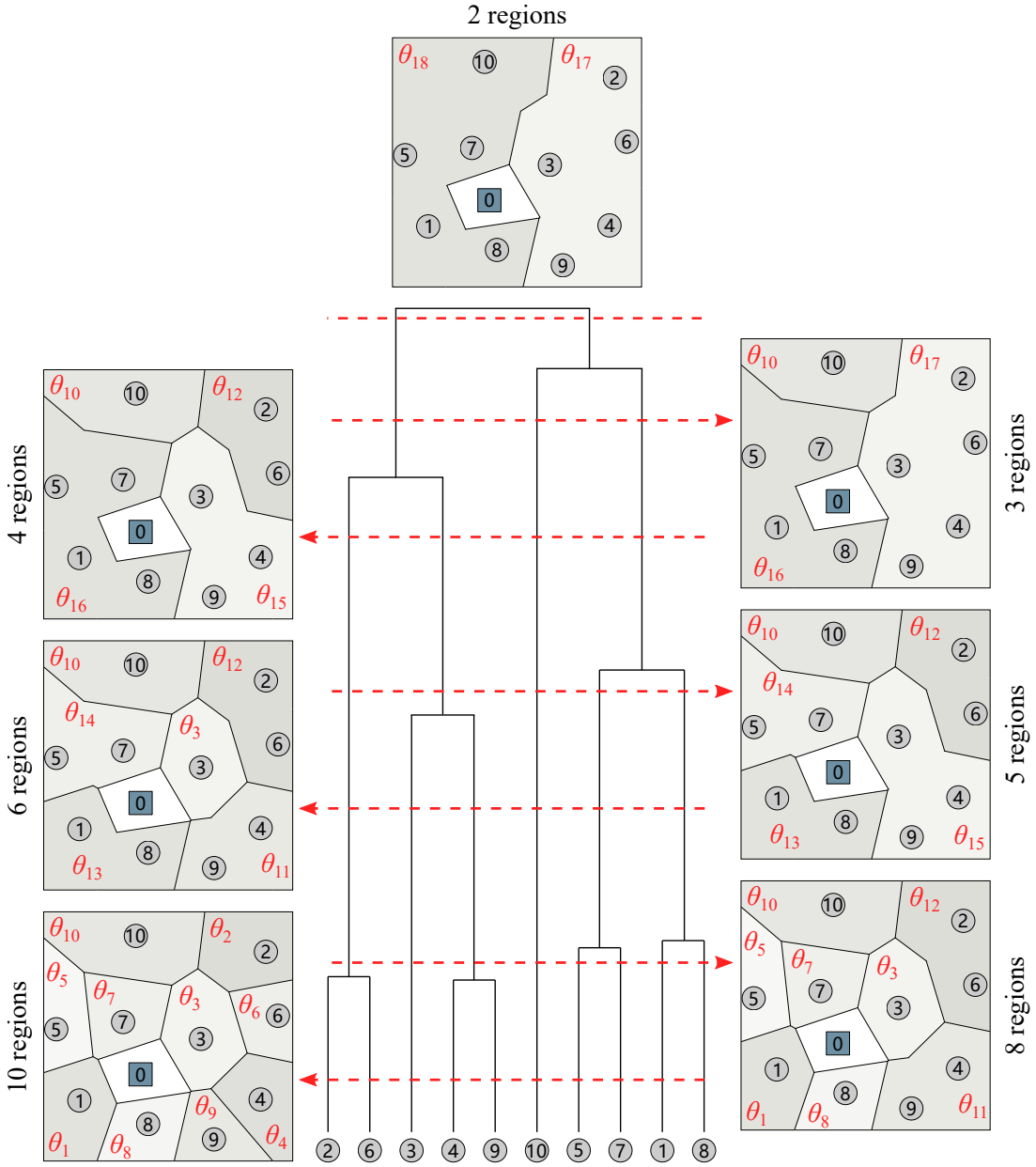


Figure 15 – Example: cartographic processing.

$\Theta^1 = \langle \theta_1, \theta_{13}, \theta_{16}, \theta_{18} \rangle$.

4.2.2 Initial routing plan

Routing scenarios are used to tackle the uncertainty of the DVRPTWSC. A routing scenario contains a possible realization of the random variables involved in the problem and a solution to the static and deterministic problem generated from that realization. In this case, each routing scenario is a solution to an instance of the vehicle routing problem with time windows (VRPTW) that contains all static customers and a subset of the dynamic customers, who are obtained by sampling their probability distributions. Formally, a routing scenario is represented as a set of m routes $S = \{r^1, \ldots, r^m\}$ where each customer is served exactly once, each route is

traversed by a different vehicle, and $m \leq \mathrm{M}$. A route $r^v = \langle r_0^v, r_1^v, \ldots, r_{n_v}^v, r_{n_v+1}^v \rangle$ is the sequence of customers served by the vehicle $v = 1, \ldots, m$. Given that every route starts and ends at the depot, $r_0^v = r_{n_v+1}^v = 0$ for $v = 1, \ldots, m$. Several routing scenarios are generated and one of them is selected as the initial routing plan.

A slightly modified version of the memetic algorithm (MA) proposed in section 3.2 is employed to create the initial routing plan. In this case, the MA population becomes a pool of routing scenarios. Nevertheless, the search operators of the MA are not modified. The routing scenarios are optimized during a fixed number of iterations. At the end of the optimization process, one routing scenario is chosen based on a similarity measure.

The first step consists of creating the pool P, containing $\mathscr{L}$ routing scenarios. A routing scenario is built by solving a VRPTW instance containing all static customers and a set of sampled dynamic customers. In this case, the nearest neighbor (NN) heuristic, proposed by Solomon (1987), is used to build the first routing scenarios of the pool.

The NN heuristic starts every route with the unrouted customer closest to the depot. At each iteration, the heuristic appends the unrouted customer closest to the last customer on the route. All insertions are performed preserving the route feasibility regarding time-window and vehicle-capacity constraints. A new route is started when there are no feasible customers remaining to be appended to the route. The process is repeated until there are no customers unrouted. The proximity between two customers is calculated with the metric $\hat{t}_{ij}$. Let i be the last customer on the current partial route and let j be an unrouted customer that could be added to the route. The metric $\hat{t}_{ij}$ is given by:

$$\hat{t}_{ij} = \omega_1 t_{ij} + \omega_2 \dot{t}_{ij} + \omega_3 \ddot{t}_{ij} \tag{4.12}$$

with

$$\dot{t}_{ij} = T_j - (T_i + s_i)\,, \tag{4.13}$$

$$\ddot{t}_{ij} = l_j - \left(T_i + s_i + t_{ij}\right). \tag{4.14}$$

Where t_{ij} is the direct travel time between customers i and j. $T_j = \max\left\{e_j, T_i + s_i + t_{ij}\right\}$ is the time at which service can begin at customer j following service at customer i. $\dot{t}_{ij}$ is the time difference between the completion of service at customer i and the start of service at customer j. $\ddot{t}_{ij}$ is the urgency of delivery to customer j, expressed in terms of the time remaining until the vehicle's last possible service start. ω_1, ω_2, and ω_3 are weights satisfying $\omega_1 \geq 0, \omega_2 \geq 0, \omega_3 \geq 0$ and $\omega_1 + \omega_2 + \omega_3 = 1$.

The routing scenarios are reoptimized using the crossover and local search operators described in section 3.2. Despite being a binary operator, the clustering-based best cost route crossover (CB-BCRC) can be used to optimize the routing scenarios regardless of whether each routing scenario potentially defines a different VRPTW instance (since the sampled dynamic customers may vary from one routing scenario to another). For optimizing a routing scenario S_j, an additional routing scenario S_o is randomly chosen from the pool; the CB-BCRC procedure

will randomly select a route r from S_o and it will seek the customers of r that are also present in S_j. The CB-BCRC procedure will only process the customers of r that are found in S_j. All customers present in r but not in S_j are omitted. It even might be possible that r turns out to be composed entirely of dynamic customers and none of them is present in S_j. In such a situation, S_j would not be modified by the CB-BCRC operator.

In the original MA, a set of clustering arrangements A is generated during the creation of the initial population of solutions. Both crossover and local search operators need the set A to operate. In the modified MA, A is not generated during the creation of the initial pool of routing scenarios. A is obtained from the dendrogram D, which was previously generated during the cartographic processing.

An example of the optimization procedure is presented in Figure 16. Given a DVRPTWSC instance with 10 static customers (1, 2, 3, 5, 7, 8, 9, 11, 12, and 13) and five dynamic customers (4, 6, 10, 14, and 15), a pool containing $\mathscr{L}$ routing scenarios is created. Each routing scenario is initially solved by using the NN heuristic. The optimization process consists of applying iteratively the crossover and local search operators to each routing scenario S_j of the pool. Each time the crossover operator is applied, an additional routing scenario S_o is randomly chosen from the pool and a route is randomly selected from S_o. In the example, the route $0 \rightarrow 2 \rightarrow 1 \rightarrow 13 \rightarrow 6, \rightarrow 5 \rightarrow 11 \rightarrow 0$ was selected. The customers belonging to the selected route are sought and removed in S_j. In this case, customer 6 is not present in S_j and thus, only customers 1, 2, 5, 11, and 13 are removed. A new solution for S_j is obtained by reinserting the removed customers according to the reinsertion procedure of the CB-BCRC operator. Finally, the local search operator refines the solution produced by the crossover operator. The optimization process is applied to all routing scenarios of the pool during a fixed number of iterations. Once all iterations are completed, the solution for each routing scenario is expected to be better than that produced initially by the NN heuristic.

Once the routing scenarios have been reoptimized, one of them is selected as the initial routing plan. Several common characteristics can be identified among the routing scenarios of the pool. For example, it is expected that those dynamic customers who are more likely to make requests will be visited in several routing scenarios, whereas the dynamic customers who are less likely to make requests will appear less often. Moreover, the sequences of visits can partially match between different routing scenarios.

The routing plan is expected to be robust enough to not require major modifications over the course of the planning horizon. It is presumed that by selecting a routing scenario that is compatible with most of the situations represented by the routing scenarios in the pool, the number of actions needed to tackle the dynamic requests will be reduced. The routing scenario most similar to the other routing scenarios in the pool becomes the initial routing plan. Similar approaches have been previously used in solution methods for dynamic vehicle routing problems (VRPs). The multiple scenario approach (MSA) proposed by Bent and Van Hentenryck (2004c)

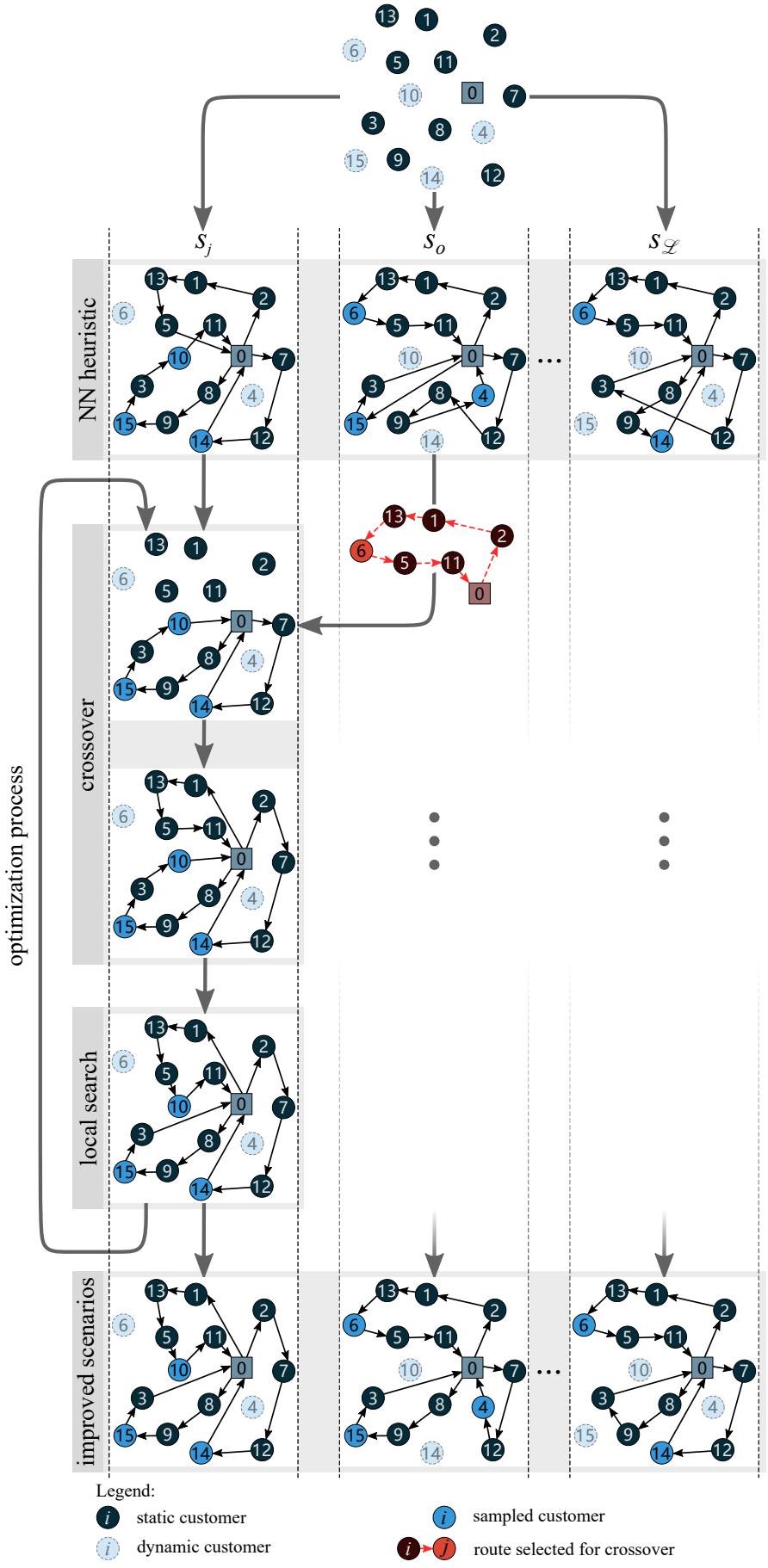


Figure 16 – Example: optimization of the routing scenarios.

keeps a pool of routing plans that are constantly updated. MSA selects the plan to be executed using a consensus function that identifies the plan most similar to the current pool of plans. The dynamic stochastic hedging heuristic (DSHH) proposed by Hvattum, Løkketangen and Laporte (2006) solves a set of sample scenarios and then seeks the common features among the resulting solutions to build a "good plan".

The similarity between two routing scenarios is calculated based on the number of route segments (edges) they have in common. Considering a routing scenario $S = \{r^1, \ldots, r^m\}$, the set of route segments of S is given by:

$$\Gamma(S) = \bigcup_{v=1}^{m} \bigcup_{k=0}^{n_v} \left\{ \left(r_k^v, r_{k+1}^v \right) \right\} \tag{4.15}$$

To calculate the similarity between two routing scenarios S_i and S_j, the Sørensen-Dice similarity coefficient (DICE, 1945) is applied to sets $\Gamma(S_i)$ and $\Gamma(S_j)$:

$$\mathrm{SD}(S_i, S_j) = \frac{2\left|\Gamma(S_i) \cap \Gamma(S_j)\right|}{|\Gamma(S_i)| + \left|\Gamma(S_j)\right|} \tag{4.16}$$

Function (4.16) returns a value in the range $[0, 1]$; the value will be closer to 1 if the routing scenarios are similar, and closer to 0 otherwise.

The average similarity between a routing scenario S_i and the rest of routing scenarios in the pool is given by the following equation:

$$\mu_P(S_i) = \frac{1}{|P| - 1} \sum_{S_j \in P \setminus \{S_i\}} \mathrm{SD}(S_i, S_j) \tag{4.17}$$

The routing scenario with the highest value of μ_P is selected as the initial routing plan. It is relevant to mention that a similar approach, based on the Jaccard similarity coefficient, was used by Garcia-Najera and Bullinaria (2011) to estimate the average similarity between a solution and the other solutions in the population of a multiobjective genetic algorithm for the VRPTW.

The pseudocode for the MA employed to create the initial routing plan is presented in Figure 17. The procedure MA' is a slightly modified version of the MA described in section 3.2, with which the reader should be familiar. This procedure takes as input a DVRPTWSC instance, Y; the pool size, $\mathscr{L}$; the maximum number of generations, G_{max}; the crossover rate, R_{rate}; the local search rate, L_{rate}; the parameters that control the local search: I_{max} (number of iterations), γ (temperature value), and β (cooling rate value); and the parameters used to create the set of clustering arrangements A: C_{min} (the minimum number of clusters), C_{max} (the maximum number of clusters), and D (the dendrogram produced in the cartographic processing). The weights used by the NN heuristic are set in line 1. A pool containing $\mathscr{L}$ routing scenarios is created in line 2. Set A is created in line 3, the CUT procedure obtains a set of different clustering arrangements

from dendrogram D. The number of clusters varies between C_{min} and C_{max}. The optimization process illustrated in Figure 16 is applied to each routing scenario of the pool through G_{max} iterations (lines 4-5). For a detailed description of the crossover and local search operators used by the NEW-POP procedure, the reader is referred to section 3.2. Line 6 sorts the routing scenarios in ascending order according to their value of μ_P. Line 7 returns the routing scenario with the the highest value of μ_P.

MA'$(Y, \mathscr{L}, G_{max}, R_{rate}, L_{rate}, I_{max}, \gamma, \beta, C_{min}, C_{max}, D)$
1 $\omega_1 = 0.333, \omega_2 = 0.333, \omega_3 = 0.333$
2 $P =$ CREATE-SCENARIOS$(Y, \mathscr{L}, \omega_1, \omega_2, \omega_3)$
3 $A =$ CUT(D, C_{min}, C_{max})
4 **for** $i = 1$ **to** G_{max}
5 $\quad P =$ NEW-POP$(P, A, \mathscr{L}, R_{rate}, L_{rate}, I_{max}, \gamma, \beta)$
6 P.sort-by$\{S \rightarrow \mu_P(S)\}$
7 **return** P.last()

Figure 17 – Pseudocode for the MA employed to create the initial routing plan.

4.2.3 Updating the routing plan in response to dynamic events

Over the planning horizon, the dispatcher updates the routing plan as the actual requests become known. The routing plan contains a set of potential dynamic requests that have not materialized at the start of the planning horizon, it is possible that some of them do not materialize at all and need to be removed from the routing plan. Furthermore, some actual dynamic requests that were not anticipated in the initial routing plan must be assigned in real time, while the vehicles are traversing their routes.

The fleet of vehicles $v = 1, \ldots, \mathrm{M}$ is based at the depot at the start of the planning horizon. For each vehicle v, the dispatcher tracks the status of the following components in real time:

1. u^v, the sequence of customers already served by vehicle v.
2. h^v, the sequence of customers to be served by vehicle v.
3. $\eta^v = (x^v, y^v)$, vehicle v's current position.
4. $\tau^+_{v,lsc}$, the cumulative travel time that vehicle v has recorded until the latest served customer, it is computed with Equation (4.1); lsc is the index of the last customer in the current sequence u^v.
5. $q^+_{v,lsc}$, the accumulated load delivered by vehicle v after serving its latest customer, it is given by Equation (4.2).

6. ℓ^v, vehicle v's remaining capacity.

7. $\varepsilon^v \in \{idle, en\ route, waiting, serving\ customer\}$, vehicle v's current state.

Initially, $u^v = \langle u_0^v \rangle = \langle 0 \rangle$, $h^v = \langle\rangle$, $\eta^v = (x_0, y_0)$, $\tau_{v,0}^+ = q_{v,0}^+ = 0$, $\ell^v = \mathrm{Q}$, and $\varepsilon^v = idle$, for $v = 1, \ldots, \mathrm{M}$.

The vehicles wait at their current location when some waiting time is expected at their next destination. As pointed out by Gendreau *et al.* (1999), this is a form of least commitment strategy that allows the dispatcher to perform last minute changes to the planned routes.

The dispatcher uses the following data structures during the planning horizon:

1. A_V, a set that contains the vehicles that have been assigned a route and are in operation.

2. I_V, a set that contains the vehicles that remain idle at the depot.

3. U_V, a set that contains the vehicles that have completed their routes.

4. B_V, a set used by the process that assigns dynamic requests to vehicles. This set contains banned vehicles, i.e., vehicles that are not considered for serving a dynamic request.

5. *received-requests*, a list that contains the customers who already have made requests. Initially, it only contains the static customers. As the dynamic customers make their requests, they are added to the list.

The first action the dispatcher takes is to assign the routes obtained from the optimization process to a subset of vehicles of the fleet. Given an initial routing plan $S = \{r^1, \ldots, r^m\}$, $h^v = \langle r_k^v | 1 \leq k \leq n_v + 1 \rangle = \langle h_1^v, \ldots, h_{n_v+1}^v \rangle$ for $v = 1, \ldots, m$ and $A_V = \{v | 1 \leq v \leq m\}$. Vehicles that are not assigned a route remain idle at the depot waiting for further instructions, $I_V = \{v | m+1 \leq v \leq \mathrm{M}\}$. When a vehicle completes its route and arrives back at the depot, it is transferred from set A_V to set U_V. Note that sequence u^v is 0-indexed and sequence h^v is 1-indexed.

The dispatcher uses function cv to check whether a dynamic customer has already been included in the routing plan. This function associates dynamic customers with one vehicle from A_V. Let DC_Y be a set containing the dynamic customers present in the DVRPTWSC instance Y, then $cv : DC_Y \rightarrow A_V \cup \{-1\}$. For a dynamic customer $j \in DC_Y$,

$$cv(j) = \begin{cases} v & \text{if } \exists v \in A_V \text{ such that } h^v.contains(j) \\ -1 & \text{otherwise} \end{cases} \tag{4.18}$$

When the dynamic customer j is not included in the routing plan, cv returns -1.

At time e_0, the vehicles in A_V receive the order to start their journeys. Each vehicle v updates its parameters as it goes on its journey. When the vehicle starts serving a customer,

sequences u^v and h^v are updated. The served customer is removed from the first position of h^v and is added to the last position of u^v. $\tau^+_{v,lsc}$, and $q^+_{v,lsc}$ are updated at the same time as u^v. The quantity delivered to the customer is subtracted from the remaining capacity ℓ^v. η^v is updated continuously while the vehicle is en route. ε^v is updated according to the current vehicle's state. An example of how a vehicle updates its parameters along its journey is presented in Figure 18. At time t_1, the vehicle is about to depart from the depot. At time t_2, the vehicle has already visited customer 3 and is waiting at its current location before traveling to customer 2. At time t_3, the vehicle has already served customers 3 and 2 and is waiting at its current location before traveling to customer 1.

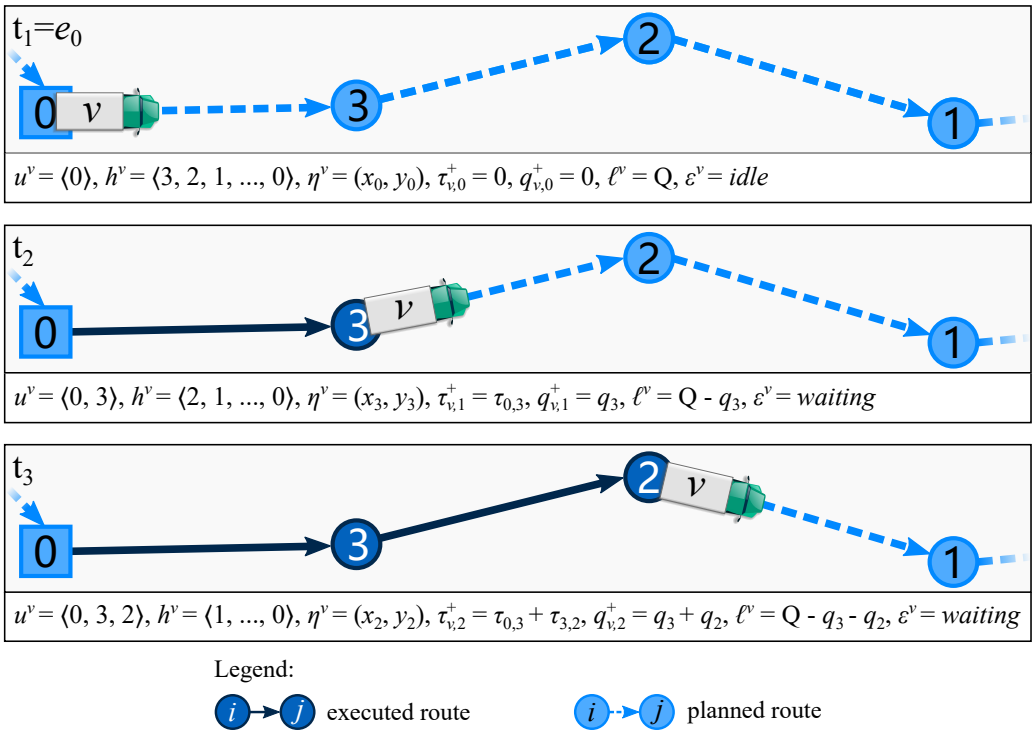


Figure 18 – Example of a vehicle updating its routing information.

The dispatcher sends a message to the vehicles ordering to change their routes in two cases: when a request considered in the initial routing plan does not materialize and needs to be removed, and when a new request needs to be added to the routing plan. Vehicles obey the orders just when they receive them. The procedures used to update the routes of the vehicles are described below.

4.2.3.1 Removing potential requests that did not materialize

For practical purposes, it is assumed that a dynamic customer j can only make a service request within the time interval $[e_0, e_j - \mathrm{t}]$, where t is a very short amount of time. The dispatcher can safely remove customer j from the routing plan if at time $e_j - \mathrm{t}$ the customer has not already placed a request.

Given a dynamic customer j, the dispatcher checks at time $e_j - \mathrm{t}$ if the customer has already placed a request. If not, the dispatcher checks whether customer j has already been included on the initial route of a vehicle. If so, the vehicle is ordered to remove customer j from its route. The procedure CHECK-DYNAMIC-REQUESTS, presented in Figure 19, performs this operation. The procedure takes as input a customer j. Line 1 checks whether the request from customer j has already been received. If not, lines 2-3 check if customer j has already been assigned to a vehicle v_a. If so, a message is sent to vehicle v_a, ordering it to remove customer j from its route (line 4).

```
CHECK-DYNAMIC-REQUESTS(j)
1  if ¬received-requests.contains(j)
2        v_a = cv(j)
3        if v_a ≠ −1
4              SEND-MESSAGE("remove customer j", v_a)
```

Figure 19 – Pseudocode for the procedure that checks if a dynamic customer has already made a request.

The procedure CHECK-DYNAMIC-REQUESTS is scheduled to be executed at each time $e_j - \mathrm{t}$, $\forall j \in DC_Y$. Whenever a vehicle v receives the order to remove a customer j from its route, it will perform the operation $h^v.remove(j)$ and then it will divert to its next destination if necessary.

Let j be the next customer that will be visited by vehicle v. Since v will wait at its current location if needed, it will arrive at customer j at time $\mathrm{t}' \geq e_j$. A vehicle can be asked to remove from its route the customer toward it is heading. In the worst case, the vehicle will have to wait, and it will start the trip toward customer j expecting to arrive at time e_j. The dispatcher will discard the request of customer j at time $e_j - \mathrm{t}$, and vehicle v will receive the order to remove that customer from its route when it has practically arrived at the customer's location. In any case, when it is necessary to remove the customer toward which a vehicle is heading, the vehicle will always be notified before arriving at the customer's location, and no unnecessary stops will be performed. A vehicle may need to divert its route if the next customer to be served is removed from its route, as illustrated in Figure 14.2. The vehicle would not need to divert its route if the next customer to be served remains the same and the removed customer is one that was scheduled to be visited later.

4.2.3.2 *Processing new requests that were not included in the initial routing plan*

Whenever a request from a dynamic customer is received, the dispatcher will check if such request has already been considered in the initial routing plan. If so, no further action is required, since a vehicle will serve that customer in due course. Otherwise, the dispatcher will assign the request to an available vehicle, or the request will be rejected if it cannot be feasibly

served. The procedure PROCESS-DYNAMIC-REQUEST, presented in Figure 20, is executed upon the arrival of a request. The procedure takes as input the customer who made the request, j. In line 1, the set of received requests is updated. Lines 2-3 check if the new request has already been assigned to a vehicle. If not, the assignation of requests based on nested regions (ARNR) is executed. This procedure will seek a suitable vehicle to serve the new request.

```
PROCESS-DYNAMIC-REQUEST(j)
1  received-requests.add(j)
2  v_a = cv(j)
3  if v_a == −1
4        ARNR(j)
```

Figure 20 – Pseudocode for the procedure that checks if a new request has already been assigned.

The ARNR procedure seeks to assign the new request to the vehicle that is capable of feasibly serving it at the lowest detour cost. Let j be the dynamic customer to be served. In the first stage, the procedure tries to assign the request to one vehicle that currently is or will be near customer j. If the request cannot be assigned, the procedure will try to assign it to one of the remaining active vehicles, those that were not considered in the first stage. If the request still cannot be assigned, the procedure will try to assign it to an idle vehicle. The request will be rejected if there is not a vehicle capable of feasibly serving it.

Before describing the ARNR procedure in detail, it is necessary to introduce some auxiliary methods. First, consider a vehicle v and a dynamic customer j, who just made a request. Vehicle v is said to be a *candidate* to serve customer j if it currently is or will be located close to that customer. The sequence of regions associated with customer j, Θ^j, is used to identify a set of candidate vehicles that could serve the request. Given a region belonging to Θ^j, the procedure IS-CANDIDATE, presented in Figure 21, checks whether a vehicle v is a candidate to serve customer j. This procedure takes as input a region Θ_a^j, with $a = 1,\ldots,z_j$, and a vehicle v. Line 1 checks if the vehicle's current coordinates, η^v, are inside the polygon associated with the region. If so, it is assumed that the vehicle is currently located close to customer j and the procedure returns TRUE. Line 3 checks if the list of customers that are inside the region and the sequence of customers to be served by the vehicle, h^v, have common elements. If so, it is assumed that the vehicle will visit the region later, and the procedure returns TRUE. If the vehicle is not inside the region and it has no plans to visit the region later, the method returns FALSE.

Figure 22 illustrates the two cases where a vehicle is considered as a candidate to serve a dynamic customer j. In Figure 22.1, the vehicle is currently located inside the region Θ_a^j, where the dynamic customer j is located. In Figure 22.2, the vehicle is about to visit some customers located in the same region as customer j.

IS-CANDIDATE(Θ_a^j, v)

1 **if** $\Theta_a^j.polygon.\text{contains}(\eta^v)$
2 **return** TRUE
3 **elseif** $\Theta_a^j.customers.\text{intersection-size}(h^v) > 0$
4 **return** TRUE
5 **return** FALSE

Figure 21 – Pseudocode for the procedure that checks if a vehicle is candidate to serve a customer.

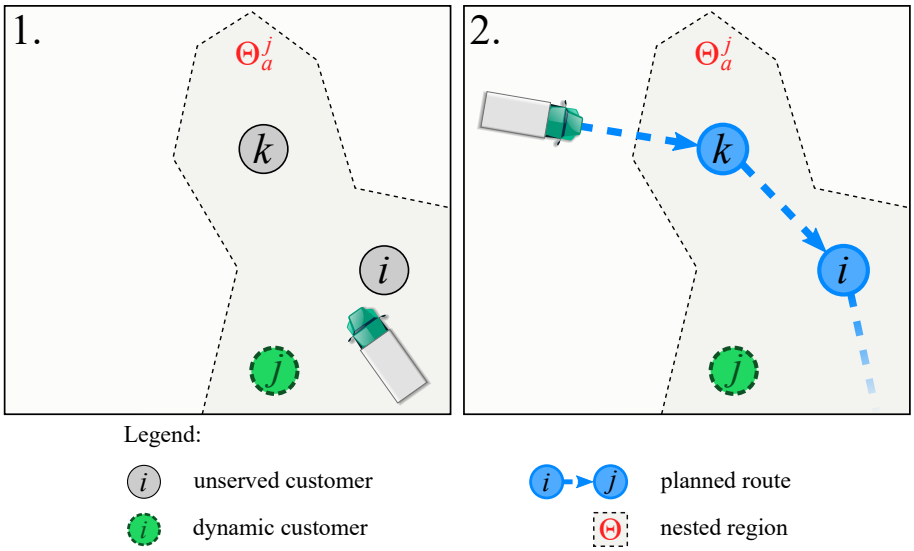


Figure 22 – Example: identifying a candidate vehicle for serving a dynamic customer.

Having identified a set of candidate vehicles to serve a dynamic customer j, the next step is to select one of them. The selection process resembles an auction, where two or more "bidder" vehicles offer a detour cost to serve the customer, and the winner is the vehicle offering the lowest detour cost.

The procedure SELECT-VEHICLE, presented in Figure 23, seeks a vehicle to serve a dynamic customer. This procedure takes as input a set of candidate vehicles, C_V; a dynamic customer, j; and a boolean flag. It returns the selected vehicle, v^*; the position at which the customer must be inserted on the route, v^*_{pos}; and the detour cost, v^*_{cost}. Normally, at least two candidate vehicles must be considered in the selection process. Nevertheless, this requirement can be relaxed in some cases. When the boolean flag is set to TRUE, the selection process can proceed with only one candidate vehicle. The selection process is only performed when C_V contains two vehicles or more or when the boolean flag is set to TRUE. These conditions are checked in line 1. If C_V contains less than two vehicles and the boolean flag is set to FALSE, the selection process is not performed and the values $v^* = -1$, $v^*_{pos} = -1$, and $v^*_{cost} = \infty$ are returned, meaning that no vehicle was selected (line 2).

SELECT-VEHICLE($C_V, j, flag$)
1 **if** $|C_V| < 2$ and $\neg flag$
2 **return** $-1, -1, \infty$
3 $v^* = -1$
4 $v^*_{pos} = -1$
5 $v^*_{cost} = \infty$
6 **for** each vehicle $v \in C_V$
7 v_{pos}, v_{cost} = CALCULATE-DETOUR-COST($h^v, \ell^v, \eta^v, \varepsilon^v, j$)
8 **if** v_{pos} == -1
9 $B_V = B_V \cup \{v\}$
10 **continue**
11 **if** $v_{cost} < v^*_{cost}$
12 $v^* = v$
13 $v^*_{pos} = v_{pos}$
14 $v^*_{cost} = v_{cost}$
15 **if** $v^* \neq -1$
16 **return** v^*, v^*_{pos}, v^*_{cost}
17 **else return** $-1, -1, \infty$

Figure 23 – Pseudocode for the procedure that selects a vehicle to serve a dynamic customer.

Variables v^*, v^*_{pos}, and v^*_{cost} are initialized in lines 3, 4, and 5, respectively. The loop on lines 6-14 selects the vehicle offering the best detour cost from the set of candidate vehicles. In line 7, the CALCULATE-DETOUR-COST procedure computes the additional travel time that vehicle v would have to spend to serve customer j. First, this procedure checks if the vehicle's current capacity minus the capacity reserved for the customers on the route is enough to serve the dynamic customer, i.e., $(\ell^v - \sum_{k=1}^{\text{len}(h^v)} q_{h^v_k}) \geq q_j$ must hold true ($\text{len}(h^v)$ denotes the length of sequence h^v). If so, the procedure seeks the feasible position in h^v where customer j can be inserted at the lowest detour cost. First, the travel time that vehicle v would spend to visit the customers in h^v is estimated considering the vehicle's current location η^v and state ε^v. Then, an exhaustive search-based method seeks the position in h^v where customer j could be inserted yielding the lowest detour cost regarding the estimated remaining travel time.

The CALCULATE-DETOUR-COST procedure returns the best insertion position for customer j, v_{pos}, and the associated detour cost, v_{cost}. If the vehicle's capacity is not enough for serving the customer or if there is not a feasible position for including the customer on the route, the procedure returns $v_{pos} = -1$ and $v_{cost} = \infty$. If vehicle v cannot feasibly serve customer j, it is added to the set of banned vehicles B_V and the search continues with another candidate vehicle (lines 8-10). If a vehicle is added to set B_V, it will no longer be considered as a candidate to serve customer j. Lines 11-14 update the selected vehicle, the corresponding insertion position, and the detour cost whenever a better candidate is found.

Line 16 returns the candidate vehicle offering the lowest detour cost to serve customer j,

v^*; the corresponding insertion position, v^*_{pos}; and the detour cost, v^*_{cost}. If none of the candidate vehicles can feasibly serve customer j, line 17 returns the values $v^* = -1$, $v^*_{pos} = -1$, and $v^*_{cost} = \infty$. Note that the selection procedure requires at least two candidate vehicles, but not necessarily all of them must be capable of feasibly serving the dynamic customer. One vehicle will be selected if, among the candidate vehicles, there is at least one vehicle capable of feasibly serving the customer.

Once a vehicle has been selected for serving a dynamic request, the dispatcher orders the vehicle to add the new customer to its route. This operation is performed by the ASSIGN-REQUEST procedure, presented in Figure 24. This procedure takes as input the selected vehicle, v^*; the position where the customer will be inserted on the route, v^*_{pos}; the detour cost associated with the insertion, v^*_{cost}; and the dynamic customer, j. Lines 1-3 calculate the value v^+_{cost}, which represents the cost of creating an additional route to serve customer j, i.e., the round-trip travel time between the depot and customer j. If there are no remaining idle vehicles, v^+_{cost} is set to ∞. If the detour cost offered by vehicle v^* to serve customer j is less than the cost of creating a new route, a message is sent to vehicle v^*, ordering it to add customer j to its route (line 5). Vehicle v^* possibly will divert its route if customer j is added at the first position on its route, as illustrated in Figure 14.1.

If customer j can be served at a lower cost by creating a new route, line 6 executes the CREATE-A-NEW-ROUTE procedure. This procedure first checks whether there are remaining idle vehicles (in this case, this condition is always met, since it was previously checked in line 2). If so, it further checks whether a vehicle departing from the depot at that moment can arrive at customer j's location before the end of the time window. Considering the arrival time of the request, R_j, and the direct travel time between the depot and customer j, $t_{0,j}$, $R_j + t_{0,j} \leq l_j$ must hold true. If both conditions are met, the procedure assigns the request of customer j to an idle vehicle $v_a \in I_V$, and it updates the sets of active and idle vehicles: $A_V = A_V \cup \{v_a\}$ and $I_V = I_V \setminus \{v_a\}$. Otherwise, the request of customer j is rejected.

```
ASSIGN-REQUEST(v*, v*_pos, v*_cost, j)
1  v+_cost = ∞
2  if |I_V| ≥ 1
3        v+_cost = CALCULATE-NEW-ROUTE-COST(j)
4  if v*_cost ≤ v+_cost
5        SEND-MESSAGE("insert customer j at position v*_pos", v*)
6  else CREATE-A-NEW-ROUTE(j)
```

Figure 24 – Pseudocode for the procedure that assigns a request to a previously selected vehicle.

The pseudocode for the ARNR procedure is presented in Figure 25. This procedure takes as input a dynamic customer j. In line 1, the contents of set B_V are removed; all active vehicles are potential candidates to serve customer j. The vehicle assigned to serve the dynamic request,

v^*; the position at which the customer must be inserted on its route, v^*_{pos}; and the associated detour cost, v^*_{cost} are initialized to the values -1, -1, and ∞, respectively (lines 2-4). These values indicate that no vehicle has been selected yet. The loop on lines 5-10 scans the nested regions associated with customer j, looking for candidate vehicles to serve the request. Line 6 creates the set C_V that will contain the candidate vehicles to serve customer j. In line 7, the active vehicles that, considering the region Θ^j_a, are candidates to serve customer j and have not been banned from serving this customer are copied to set C_V. In line 8, the SELECT-VEHICLE procedure selects a vehicle from the current set of candidate vehicles. The boolean flag used to call the SELECT-VEHICLE procedure is set to TRUE only if the region being scanned is the last (and largest) region in Θ^j. Line 9 checks whether a vehicle capable of feasibly serving customer j was found by the SELECT-VEHICLE procedure. If so, line 10 ends the search. Otherwise, the search continues with the next region in Θ^j.

At each iteration, the search moves up through the hierarchy of nested regions, and a larger region is scanned. As the scanned region grows, so too do the chances of finding a suitable vehicle for serving customer j. Nevertheless, candidate vehicles found in smaller regions should offer lower detour costs, since they currently are or will be nearer customer j. If a vehicle is added to set B_V by the SELECT-VEHICLE procedure, it will not be considered as a candidate vehicle in following iterations.

Aiming to take a better-informed decision, at least two vehicles are considered for serving a dynamic customer, even if this makes it necessary to scan additional regions. Consider the example in Figure 26. Figure 26.1 shows the dynamic customer 1 and two vehicles traversing their routes. The sequence of regions associated with customer 1 is $\Theta^1 = \langle \Theta^1_1, \ldots, \Theta^1_a, \ldots, \Theta^1_{z_1} \rangle$. Vehicle 2 is currently located very close to customer 1, whereas vehicle 1 is farther away. Seeking a vehicle for serving customer 1, Θ^1_1 would be the first region to be scanned. If only one vehicle were considered for serving the dynamic customer, vehicle 2 would have been selected for serving the request, since it is currently inside region Θ^1_1 and it can feasibly serve customer 1. Nevertheless, despite vehicle 2 being very close to customer 1, it only would serve the new request after visiting customer 3, since the customers' time windows do not allow a feasible insertion at another position on the route (Figure 26.2). Vehicle 1 is currently far from customer 1, this vehicle would only have been identified as a candidate to serve customer 1 by scanning the larger region Θ^1_a. If the request were assigned to the first suitable vehicle found, vehicle 1 would not even have been considered. Nevertheless, vehicle 1 should be selected, since it can feasibly serve customer 1 at a lower detour cost (Figure 26.3).

By comparing the detour costs offered by at least two vehicles, the selection procedure seeks to escape local minima. Nevertheless, when scanning the last (and largest) region associated with a dynamic customer j, the selection process can be carried out with only one vehicle. It is possible that only one candidate vehicle has been identified after scanning customer j's last region, and this would be the last chance to select a vehicle that fit the definition of *candidate*

vehicle for serving this customer.

If the search through the nested regions finds a suitable vehicle v^* to serve customer j, the ASSIGN-REQUEST procedure is called in line 12. Otherwise, all the remaining active vehicles, which were not considered in the search through the nested regions (and thus have not been banned yet), become candidates to serve customer j, and the SELECT-VEHICLE procedure tries to select one of them (lines 13-15). In this case, all active vehicles end up being considered for serving the dynamic request. If a suitable vehicle v^* is found, the ASSIGN-REQUEST procedure is called in line 17. If none of the active vehicles can serve the dynamic customer, line 18 calls the CREATE-A-NEW-ROUTE procedure, as mentioned before, this procedure first checks if there are still available idle vehicles and then checks if the customer can be served without lateness, considering the arrival time of the request, R_j. If both conditions are met, customer j's request is assigned to an idle vehicle and sets A_V and I_V are updated accordingly. Otherwise, customer j's request is rejected.

```
ARNR(j)
 1  B_V = ∅
 2  v* = −1
 3  v*_pos = −1
 4  v*_cost = ∞
 5  for a = 1 to z_j
 6       let C_V be a new set
 7       A_V.copy-to(C_V){v → IS-CANDIDATE(Θ^j_a, v) and ¬B_V.contains(v)}
 8       v*, v*_pos, v*_cost = SELECT-VEHICLE(C_V, j, a == z_j)
 9       if v* ≠ −1
10            break
11  if v* ≠ −1
12       ASSIGN-REQUEST(v*, v*_pos, v*_cost, j)
13  else let C_V be a new set
14       A_V.copy-to(C_V){v → ¬B_V.contains(v)}
15       v*, v*_pos, v*_cost = SELECT-VEHICLE(C_V, j, TRUE)
16       if v* ≠ −1
17            ASSIGN-REQUEST(v*, v*_pos, v*_cost, j)
18       else CREATE-A-NEW-ROUTE(j)
```

Figure 25 – Pseudocode for the assignation of requests based on nested regions (ARNR).

An example of how a dynamic request is assigned to a vehicle is presented in Figure 27. The cartographic processing has been applied to a DVRPTWSC instance with seven customers (Figure 27.1). When the dynamic customer 1 makes a request, three vehicles are currently traversing their routes. The first region associated with customer 1, Θ_1^1, is scanned; no candidate vehicle is found to serve the request (Figure 27.2). Moving one level further up the hierarchy of nested regions, region Θ_2^1 is scanned. In this case, vehicle 1 is identified as a candidate

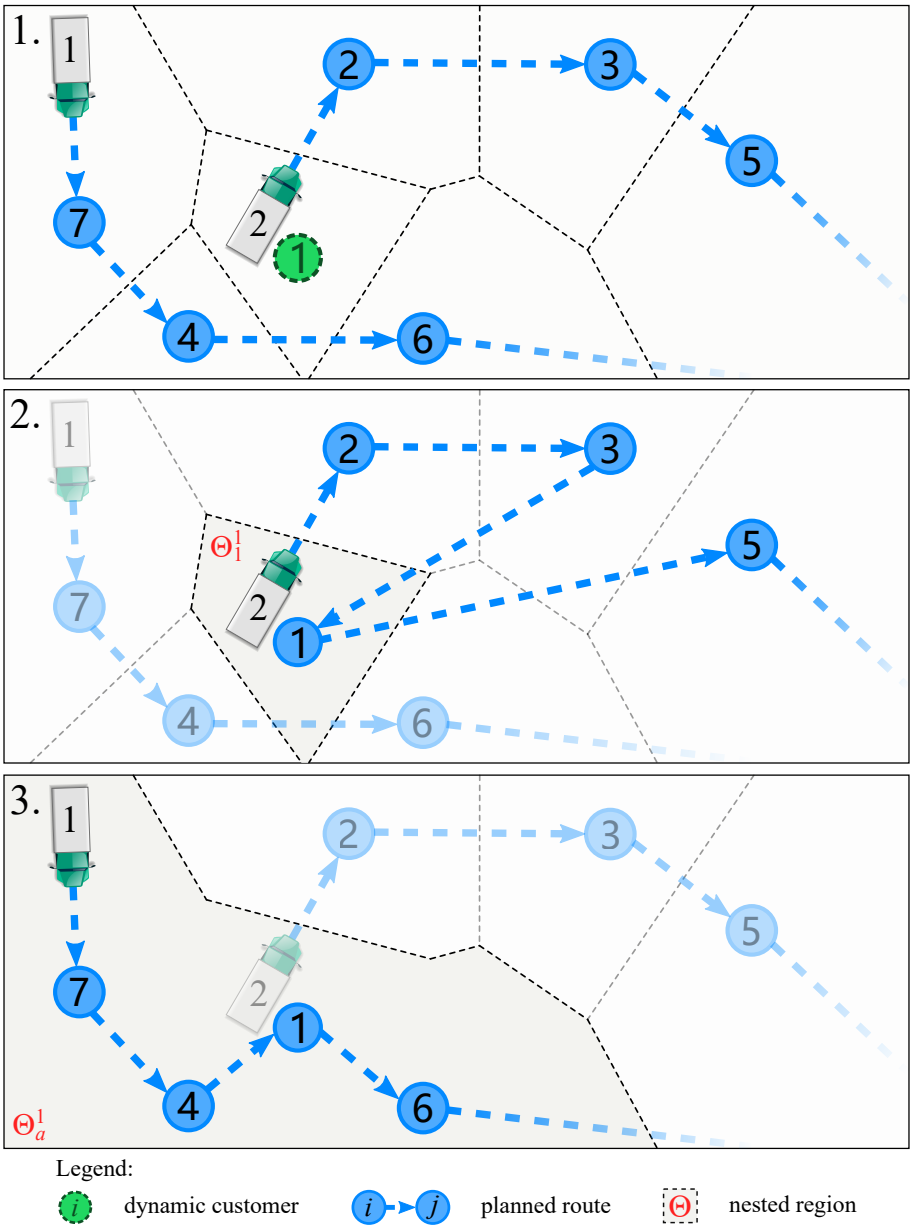


Figure 26 – Example: a vehicle traveling far from a dynamic customer could serve the request at a lower cost than one traveling nearby.

to serve customer 1 since it will visit customer 4, and both customers belong to the same region (Figure 27.3). Since there is only one candidate vehicle to serve the customer, the search continues, and region Θ_3^1 is scanned; again, only vehicle 1 is identified as a candidate to serve customer 1 (Figure 27.4). The search procedure finally scans region Θ_4^1, the largest region associated with customer 1. In this case, vehicle 1 and vehicle 2 are identified as candidates to serve customer 1. Vehicle 1 will visit the region, and vehicle 2 is currently inside the region (Figure 27.5). Assuming that vehicle 1 can feasibly serve customer 1 at a lower detour cost than vehicle 2, vehicle 1 is selected for serving the request (Figure 27.6). In this example, vehicle 3 was not even considered during the selection process.

The ARNR procedure focus the search effort on those vehicles that look more promising

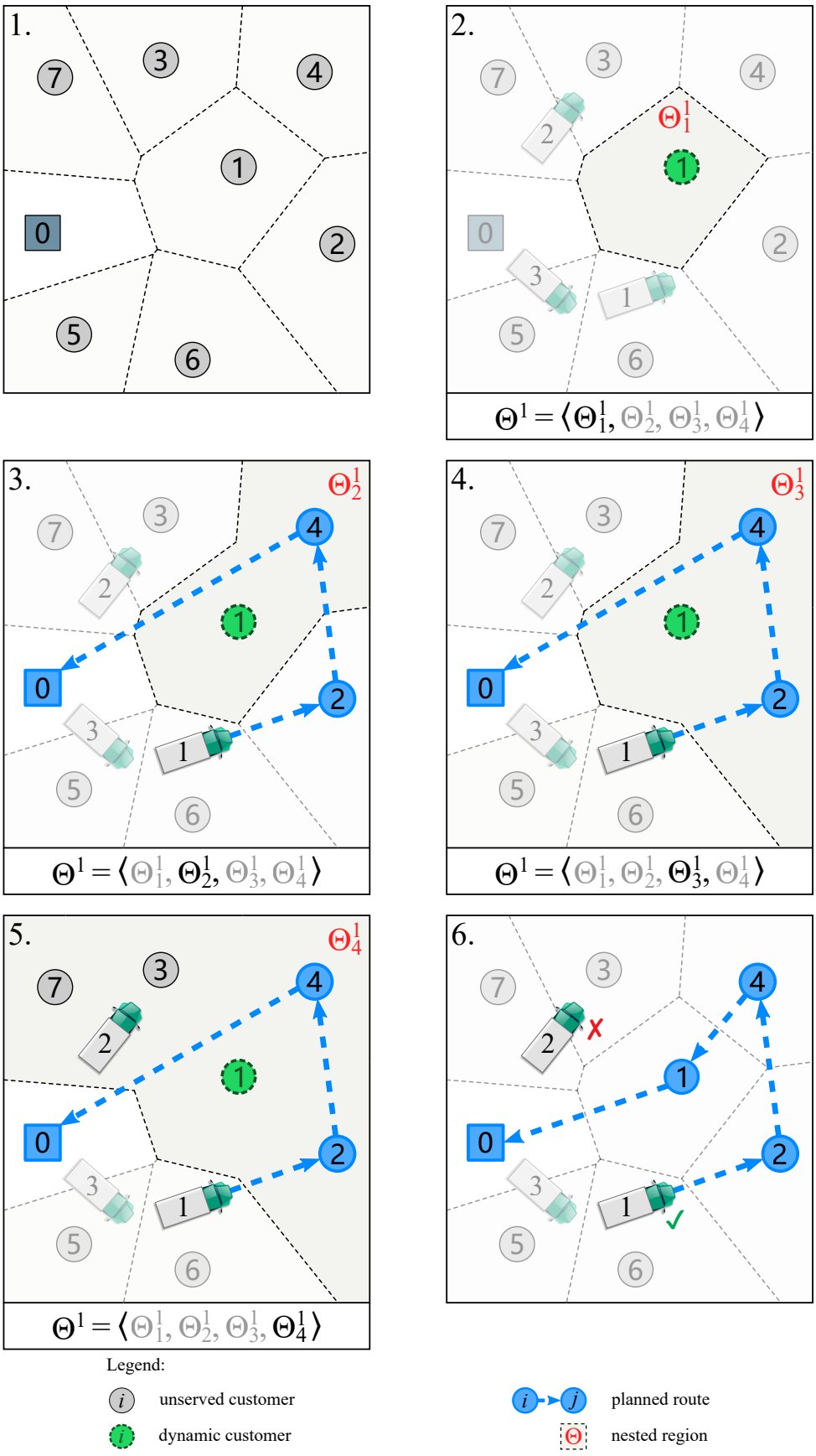


Figure 27 – Example: the operation of lines 5–12 in the ARNR procedure.

for serving a dynamic request. The aim is to reduce the number of vehicles for which the detour cost is calculated, without compromising the final solution quality. The ARNR procedure uses the spatial proximity between customers as a guide for reducing the computational burden of the search. The MA employed for optimizing the routing scenarios follows a similar rationale. Indeed, both the crossover and local search operators of the MA presented in subsection 4.2.2 can be considered by-products of the ARNR procedure, since the result of the hierarchical clustering, which originally is intended to be used by the ARNR procedure, is also used by these operators to solve a static problem.

Other multiagent approaches to VRPs have adopted mechanisms similar to the ARNR to reduce the number of vehicles considered for serving new requests. In the multiagent system proposed by Dan, Cai and Zheng (2009), the vehicles candidates for serving a new request are identified by using a fixed radius around the location of the customer requiring service.

Over the planning horizon, the routing plan is updated according to the information revealed to the dispatcher. When a vehicle v arrives at the depot after serving its customers, its state is set to *idle*, and it is transferred from set A_V to set U_V: $A_V = A_V \setminus \{v\}$, $U_V = U_V \cup \{v\}$. Vehicle v is not considered anymore for serving new requests. For every vehicle v in set U_v, the sequence of served customers is $u^v = \langle u_0^v, u_1^v, \ldots, u_{n_v}^v, u_{n_v+1}^v \rangle$, where $u_0^v = u_{n_v+1}^v = 0$; the cumulative travel time is $\tau_{v,lsc}^+ = \tau_{v,n_v+1}^+ = ftt_v$; and the accumulated delivered load is $q_{v,lsc}^+ = q_{v,n_v}^+ = fdl_v$.

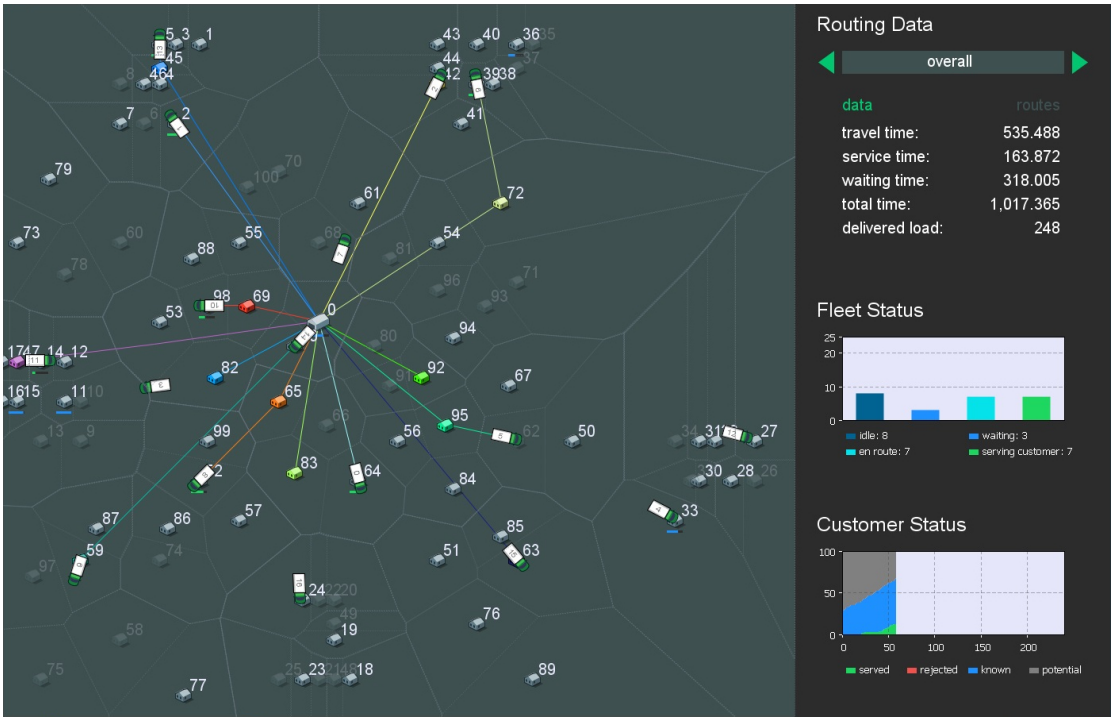
Once the m' vehicles assigned to a route arrive at the depot after serving their assigned customers, the set $U_V = \{1, \ldots, m'\}$ is obtained (with $m \leq m' \leq$ M). Then, the routing plan is assessed in terms of the criteria given at the end of section 4.1.

4.3 Implementation of the Proposed Approach

The MA employed for creating the initial routing plan was implemented using the Kotlin programming language. The multiagent system was developed and tested in Anylogic 7, university edition. The MA was imported into the model as an external library.

A benchmark set composed of 48 DVRPTWSC instances was generated to test the proposed approach. The generated instances represent situations with low, medium, and high dynamism, and they are based on the RC instances of Solomon (1987). The details regarding the creation of the instances are provided in Appendix A. First, the model reads the data from the instance. At that moment, the dispatcher knows who the static customers are and what are the probabilities of receiving requests from the dynamic customers. The customers are programmed to recreate the events defined by the instance. The static customers make their requests before the start of the planning horizon, some dynamic customers are programed to make their requests in due course, the remaining dynamic customers do not make requests. The customers make their requests by sending a message to the dispatcher. When the simulation starts, the dispatcher

1.



2.

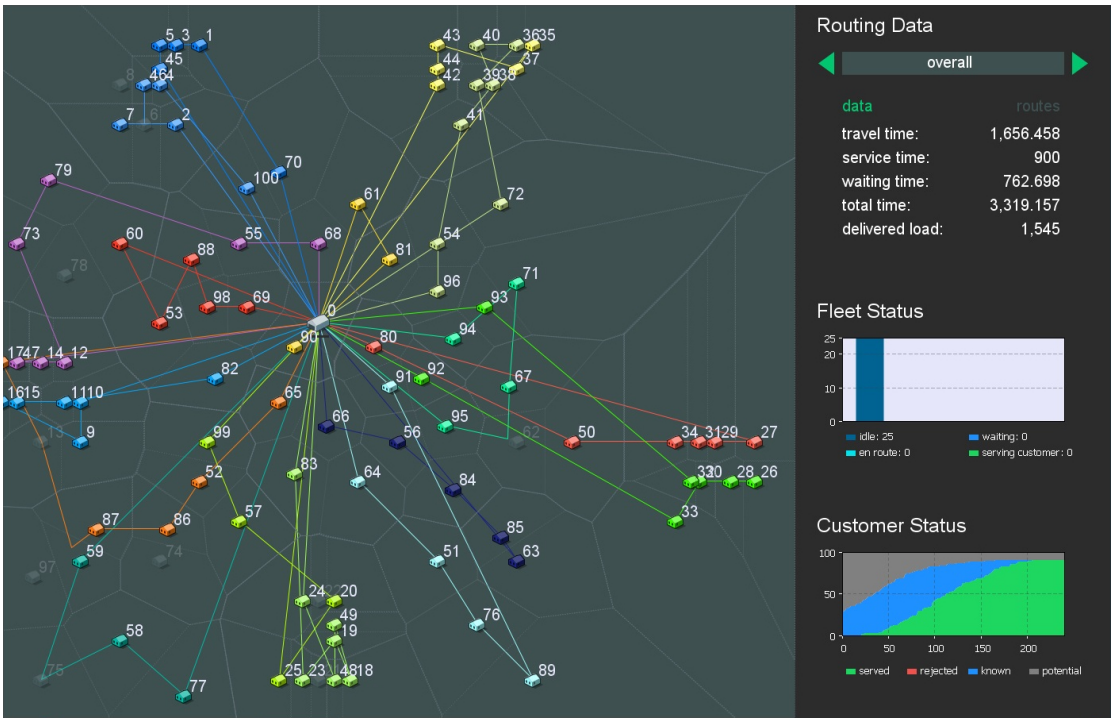


Figure 28 – A simulation run.

creates the initial routing plan and begins to manage the fleet following the procedures presented in subsection 4.2.3. The vehicles continuously update their routing information and follow the dispatcher's orders.

Figure 28 presents a simulation running at two different moments. Figure 28.1 shows a partially executed routing plan at time $t = 0.25(l_0 - e_0)$ and Figure 28.2 shows the routing

plan after being completely executed at the end of the planning horizon (i.e., at time t = l_0). The developed graphical interface allows to track the trips made by the vehicles on the map, as well as checking the sequences of customers served by the vehicles, the travel time (total and per vehicle), the delivered load (total and per vehicle), the overall fleet status, and the number of served and unserved customers.

4.4 Computational Results

The experiments were conducted on an Intel Core i7-4790U PC with 16 GB memory. All calculations were performed using double-digit precision. For all experiments, the MA parameters were set to the values presented in Table 8. These values were selected after preliminary testing, where a wide range of parameter settings was explored seeking a balanced trade-off between fast execution times and solution quality. The hierarchical clustering was performed using the unweighted pair group method with arithmetic mean (UPGMA) algorithm.

Table 8 – MA' Parameters

Parameter	Value
Population size, $\mathscr{L}$	100
Maximum number of generations, G_{max}	1000
Crossover rate, R_{rate}	0.9
Local search rate, L_{rate}	0.03
Number of iterations of the inter-route neighborhood search, I_{max}	50
Initial temperature of the intra-route neighborhood search, γ	100000
Cooling rate of the intra-route neighborhood search, β	0.001
Minimum number of clusters, C_{min}	2
Maximum number of clusters, C_{max}	25*
Weight used to penalize violations to time-window constraints, α_1^{**}	1000
Weight used to penalize violations to vehicle-capacity constraints, α_2^{**}	1000

* The fleet size in the considered instances.
** For further details on how infeasible solutions are penalized in the MA, the reader is referred to section 3.1.

4.4.1 Impact of the reoptimization on the quality of the routing scenarios

A set of experiments was conducted to assess the impact of the reoptimization performed by the MA on the routing scenarios created by the NN heuristic. A pool containing 100 routing scenarios was generated for each DVRPTWSC instance. In each experiment, the pools of routing scenarios underwent the reoptimization process performed by the loop on lines 4-5 of the MA' procedure (presented in Figure 17). 10 independent experiments were conducted. Since the benchmark set contains 48 instances, a total of 4800 routing scenarios underwent the reoptimization process.

The total expected number of requests of a DVRPTWSC instance could vary depending on its level of dynamism. Nevertheless, as shown in Appendix A, the total expected number of requests is equal to 90 for all instances, regardless their associated level of dynamism. Thus,

Table 9 – Descriptive statistics: impact of the reoptimization on the quality of the routing scenarios

Data Set	Run no.	Statistics							
		Obs	Min	Max	1st Q.	Median	3rd Q.	Mean	St Dv.
RC1	NN	2400	1273.30	2476.59	1724.49	1852.62	2010.83	1861.37	204.54
	MA - run 1	2400	950.27	1703.63	1136.60	1268.13	1434.77	1289.95	176.47
	MA - run 2	2400	939.50	1680.47	1136.00	1269.45	1427.96	1289.72	175.28
	MA - run 3	2400	944.09	1674.71	1137.98	1264.94	1432.32	1289.87	176.13
	MA - run 4	2400	946.65	1714.58	1134.06	1266.59	1431.94	1290.08	176.58
	MA - run 5	2400	932.73	1687.23	1135.95	1268.82	1432.45	1290.02	175.77
	MA - run 6	2400	940.90	1701.25	1135.42	1267.82	1431.28	1289.41	175.76
	MA - run 7	2400	925.74	1714.03	1135.46	1269.76	1435.26	1289.55	176.52
	MA - run 8	2400	933.25	1676.90	1135.30	1269.85	1433.15	1290.22	176.16
	MA - run 9	2400	945.25	1717.26	1137.70	1268.83	1431.58	1290.79	175.79
	MA - run 10	2400	942.89	1704.69	1137.81	1271.99	1435.24	1290.52	175.89
RC2	NN	2400	1068.06	3087.07	2031.68	2341.89	2576.38	2253.74	437.42
	MA - run 1	2400	645.79	1415.67	849.74	1009.99	1125.69	1003.37	173.60
	MA - run 2	2400	677.37	1410.96	852.88	998.36	1125.13	999.64	174.34
	MA - run 3	2400	662.71	1451.25	852.01	1004.21	1125.70	1002.13	173.06
	MA - run 4	2400	644.53	1455.04	854.21	1007.54	1129.36	1002.99	175.36
	MA - run 5	2400	675.67	1439.84	855.69	1008.22	1125.19	1002.21	172.97
	MA - run 6	2400	653.71	1410.00	852.82	1004.40	1130.34	1001.32	173.82
	MA - run 7	2400	670.87	1421.39	855.63	1005.53	1125.03	1001.61	173.28
	MA - run 8	2400	655.39	1449.55	853.77	1007.33	1126.66	1002.25	172.70
	MA - run 9	2400	645.67	1458.39	855.12	1008.35	1126.23	1002.03	174.18
	MA - run 10	2400	661.09	1416.29	855.20	1007.91	1123.75	1000.10	173.16
All r. scenarios	NN	4800	1068.06	3087.07	1773.16	2015.65	2343.67	2057.55	393.78
	MA - run 1	4800	645.79	1703.63	1002.82	1132.04	1301.20	1146.66	226.21
	MA - run 2	4800	677.37	1680.47	994.84	1130.90	1303.57	1144.68	227.14
	MA - run 3	4800	662.71	1674.71	999.61	1131.96	1302.84	1146.00	226.23
	MA - run 4	4800	644.53	1714.58	1000.02	1132.37	1307.66	1146.54	227.08
	MA - run 5	4800	675.67	1687.23	1002.83	1130.88	1306.29	1146.11	226.08
	MA - run 6	4800	653.71	1701.25	1000.57	1132.73	1299.14	1145.36	226.49
	MA - run 7	4800	670.87	1714.03	998.79	1130.58	1302.48	1145.58	226.54
	MA - run 8	4800	655.39	1676.90	1003.35	1131.16	1303.24	1146.24	226.18
	MA - run 9	4800	645.67	1717.26	999.95	1132.57	1305.29	1146.41	226.86
	MA - run 10	4800	661.09	1704.69	999.16	1132.00	1301.40	1145.31	227.04

the routing scenarios derived from the instances of the benchmark set are expected to contain a similar number of requests, regardless the instance they are derived from. Since, in this case, the level of dynamism of the instances does not cause the routing scenarios to have different characteristics, the results were analyzed grouping the instances only by their classification regarding time-window length and vehicle capacity.

The results obtained are presented in Table 9 and Figure 29. The travel time statistics that describe the pools of routing scenarios created by the NN heuristic are presented together with the travel time statistics that describe the pools of routing scenarios after being reoptimized in each experiment. For the aggregated and grouped results, the descriptive statistics show that noticeable improvements were achieved after each reoptimization run. Table 10 presents the gaps in average travel times between the original and reoptimized pools of routing scenarios. Let $\bar{x}_{tt}$ be the average travel time of the pools of routing scenarios created by the NN heuristic and let $\bar{x}^*_{tt}$ be the average travel time of the pools of routing scenarios after undergoing the reoptimization process. The gap is calculated as follows: $100 \cdot (\bar{x}^*_{tt} - \bar{x}_{tt})/\bar{x}_{tt}$.

Analyzing the gaps averaged over all experiments, it is possible to see that the reoptimization process reduced, on average, by 30.7% the average travel time of the routing scenarios

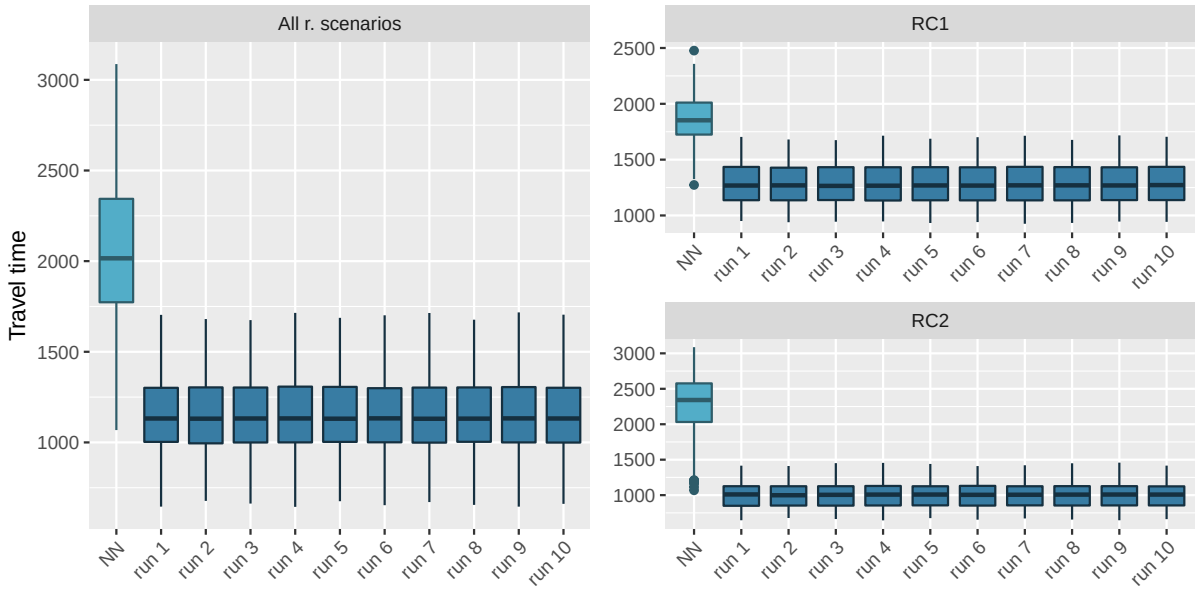


Figure 29 – Box plots: impact of the reoptimization on the quality of the routing scenarios.

Table 10 – Gaps in average travel times between the original and reoptimized pools of routing scenarios

Instance Class	Gap (%)										
	Run no.										
	1	2	3	4	5	6	7	8	9	10	Average
RC1	-30.70	-30.71	-30.70	-30.69	-30.70	-30.73	-30.72	-30.68	-30.65	-30.67	-30.70
RC2	-55.48	-55.65	-55.53	-55.50	-55.53	-55.57	-55.56	-55.53	-55.54	-55.62	-55.55
All r. scenarios	-44.27	-44.37	-44.30	-44.28	-44.30	-44.33	-44.32	-44.29	-44.28	-44.34	-44.31

for instances of class RC1. For instances of class RC2, the reduction in the average travel time was 55.55% on average. Considering all routing scenarios, the reduction in the average travel time was 44.31% on average. These results suggest that the reoptimization process noticeably improves the quality of the routing scenarios produced by the NN heuristic. The impact of the reoptimization process is more noticeable on instances with wide time windows and a larger vehicle capacity.

Statistical tests were conducted to confirm the results observed in Table 9 and Figure 29. First, the Shapiro–Wilk test was applied to test the normality of the aggregated and grouped results. In all cases, the data did not conform to normality, because all p-values were less than 0.05. The routing scenarios created by the NN heuristic and the routing scenarios obtained after each run of the reoptimization process were compared by Kruskal-Wallis tests with Dunn's post hoc tests. The p-values were corrected for multiple comparisons by a false discovery rate (FDR) correction procedure. At a 95% confidence level, the Kruskal–Wallis tests indicated that there is a significant difference in travel times between the obtained results (p-values < 0.05 for the results grouped by class and the aggregate results). The results of the Dunn's post hoc tests are presented in Table 11. Both for the results grouped by class and the aggregate results, the Dunn's tests showed that there is a statistically significant difference in travel times between the routing

Table 11 – Dunn's corrected p-values: impact of the reoptimization on the quality of the routing scenarios

Data Set		NN	run 1	run 2	run 3	run 4	run 5	run 6	run 7	run 8	run 9
RC1	run 1	< 0.05	-	-	-	-	-	-	-	-	-
	run 2	< 0.05	1.00	-	-	-	-	-	-	-	-
	run 3	< 0.05	1.00	1.00	-	-	-	-	-	-	-
	run 4	< 0.05	1.00	1.00	1.00	-	-	-	-	-	-
	run 5	< 0.05	1.00	1.00	1.00	1.00	-	-	-	-	-
	run 6	< 0.05	1.00	1.00	1.00	1.00	1.00	-	-	-	-
	run 7	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	-	-	-
	run 8	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-	-
	run 9	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-
	run 10	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
RC2	run 1	< 0.05	-	-	-	-	-	-	-	-	-
	run 2	< 0.05	1.00	-	-	-	-	-	-	-	-
	run 3	< 0.05	1.00	1.00	-	-	-	-	-	-	-
	run 4	< 0.05	1.00	1.00	1.00	-	-	-	-	-	-
	run 5	< 0.05	1.00	1.00	1.00	1.00	-	-	-	-	-
	run 6	< 0.05	1.00	1.00	1.00	1.00	1.00	-	-	-	-
	run 7	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	-	-	-
	run 8	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-	-
	run 9	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-
	run 10	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
All scenarios	run 1	< 0.05	-	-	-	-	-	-	-	-	-
	run 2	< 0.05	1.00	-	-	-	-	-	-	-	-
	run 3	< 0.05	1.00	1.00	-	-	-	-	-	-	-
	run 4	< 0.05	1.00	1.00	1.00	-	-	-	-	-	-
	run 5	< 0.05	1.00	1.00	1.00	1.00	-	-	-	-	-
	run 6	< 0.05	1.00	1.00	1.00	1.00	1.00	-	-	-	-
	run 7	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	-	-	-
	run 8	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-	-
	run 9	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-
	run 10	< 0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

scenarios created by the NN heuristic and the routing scenarios reoptimized in each experiment (p-values < 0.05 in all cases). Moreover, the improvements attained by the reoptimization process were consistent over all experiments, because there is no statistically significant difference in travel times between the routing scenarios obtained after each experiment run (all comparisons between experiment runs yielded p-values > 0.05). These results confirm that the reoptimization procedure significantly improves the quality of the routing scenarios.

4.4.2 Comparison of approaches to addressing dynamism

Let Sc-ARNR denote the described solution methodology, where the initial routing plan is created considering the static requests and some likely dynamic requests that are not confirmed at the start of the planning horizon, and the ARNR algorithm is used to assign the dynamic requests that were not included in the initial routing plan. The Sc-ARNR approach was compared with other approaches to the DVRPTWSC: Sc-AllV, Kw-ARNR, and Kw-AllV. In the Sc-AllV approach, the initial routing plan contains the static requests and some dynamic requests that are not confirmed at the beginning of the planning horizon, but in this case, all active vehicles are considered as candidates to serve the dynamic requests that were not included in the initial routing plan. Thus, the idea of focusing the search on a subset of vehicles is dropped in favor of

Table 12 – Descriptive statistics: overall results for all DVRPTWSC instances

Variable	Approach	Statistics							
		Obs	Min	Max	1st Q.	Median	3rd Q.	Mean	St Dv.
Travel time	Kw-AllV	480	857.71	2444.30	1318.06	1555.60	1910.03	1610.26	370.88
	Kw-ARNR	480	973.53	2909.15	1387.40	1578.44	1987.42	1657.31	410.71
	Sc-AllV	480	760.55	1979.10	1065.24	1245.80	1430.29	1244.96	260.58
	Sc-ARNR	480	760.55	1979.10	1066.19	1255.80	1432.91	1250.89	262.80
No. of vehicles	Kw-AllV	480	3.00	20.00	5.00	9.50	16.00	10.26	5.61
	Kw-ARNR	480	3.00	20.00	5.00	10.00	16.00	10.74	5.50
	Sc-AllV	480	3.00	19.00	5.00	9.00	13.00	9.46	4.80
	Sc-ARNR	480	3.00	19.00	5.00	9.50	13.00	9.53	4.76
Lateness,	Kw-AllV	480	0.00	0.00	0.00	0.00	0.00	0.00	0.00
overload,	Kw-ARNR	480	0.00	0.00	0.00	0.00	0.00	0.00	0.00
rejected requests	Sc-AllV	480	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Sc-ARNR	480	0.00	0.00	0.00	0.00	0.00	0.00	0.00

an exhaustive search-based approach. The Kw-ARNR approach consists of creating an initial routing plan considering only the static requests and assigning the dynamic requests using the ARNR algorithm. In the Kw-AllV approach, the initial routing plan contains only the static requests and all vehicles are considered as candidates to serve the dynamic requests. In both the Kw-ARNR and Kw-AllV approaches, the MA and the parameter settings presented in Chapter 3 were employed for creating the initial routing plan.

Each approach was applied 10 times to each instance of the benchmark set, rendering a total of $4 \times 48 \times 10 = 1920$ solutions. The initial routing plans created by the Sc-ARNR and Kw-ARNR approaches were also used by the Sc-AllV and Kw-AllV approaches, respectively. The solutions are assessed considering the total travel time, the number of vehicles used, the possible violations to time-window and vehicle-capacity constraints, and the number of rejected requests.

The overall results are presented in Table 12. Analyzing the results in terms of travel time, it is possible to observe that both the Sc-ARNR and the Sc-AllV approaches performed better than the other approaches. This suggests that by including likely dynamic requests in the initial routing plan, the final travel times spent by vehicles are reduced. Regarding the number of vehicles used, the results obtained by all approaches were similar. All obtained solutions represent feasible transport operations, since no violations were recorded to time-window and vehicle-capacity constraints. Furthermore, there were no rejected requests, the number of vehicles of the fleet was enough to serve both the static and dynamic requests in all solutions.

The results grouped by data set are presented in Table 13, Figure 30, and Figure 31. The best solutions found by each approach are presented in Table 14. TT and NV denote, respectively, travel time and number of vehicles used. Assessing the solutions in terms of travel time, it is possible to observe that, on all data sets, the Sc-AllV and Sc-ARNR approaches performed very similarly and both approaches outperformed the Kw-AllV and Kw-AllV approaches. The performance improvement attained by the Sc approaches is more noticeable on instances of high and medium dynamism.

Table 13 – Descriptive statistics: travel time and number of vehicles per data set

Data Set	Approach	Statistics														
			Travel time							No. of vehicles						
		Obs	Min	Max	1st Q.	Median	3rd Q.	Mean	St Dv.	Min	Max	1st Q.	Median	3rd Q.	Mean	St Dv.
RC1-L	Kw-AllV	80	1262.91	1768.17	1389.41	1565.57	1617.82	1518.41	146.10	11.00	17.00	12.00	13.00	16.00	13.81	1.71
	Kw-ARNR	80	1276.66	1738.83	1414.02	1564.07	1647.87	1534.31	133.47	12.00	17.00	13.00	13.00	15.25	13.79	1.53
	Sc-AllV	80	1098.50	1692.04	1310.18	1403.13	1599.64	1435.30	164.68	11.00	17.00	12.00	13.00	15.00	13.80	1.90
	Sc-ARNR	80	1098.50	1718.32	1322.52	1419.30	1605.43	1445.58	166.34	11.00	18.00	12.00	13.00	15.00	13.79	1.94
RC1-M	Kw-AllV	80	1583.37	2240.39	1799.45	1913.47	2039.92	1930.89	191.21	14.00	20.00	14.75	15.50	16.25	15.84	1.77
	Kw-ARNR	80	1603.89	2290.93	1824.91	1925.55	2052.87	1942.09	199.24	14.00	20.00	14.00	15.50	16.25	15.80	1.80
	Sc-AllV	80	1189.62	1857.36	1291.49	1449.04	1577.88	1467.60	180.18	11.00	19.00	12.00	14.00	16.00	14.21	2.16
	Sc-ARNR	80	1189.62	1858.98	1291.49	1453.42	1603.06	1474.82	182.26	11.00	19.00	12.00	14.00	16.00	14.22	2.18
RC1-H	Kw-AllV	80	1825.14	2444.30	1997.98	2110.48	2194.58	2116.86	175.82	14.00	20.00	16.00	17.50	18.00	17.16	1.59
	Kw-ARNR	80	2050.26	2909.15	2108.92	2150.81	2274.47	2263.18	267.26	15.00	20.00	16.75	19.00	19.00	18.07	1.55
	Sc-AllV	80	1167.45	1979.10	1301.57	1401.04	1552.65	1432.36	166.17	11.00	19.00	12.00	14.00	16.00	13.96	2.06
	Sc-ARNR	80	1184.21	1979.10	1310.39	1405.78	1558.35	1437.22	164.24	11.00	19.00	12.00	14.00	16.00	13.97	2.04
RC2-L	Kw-AllV	80	857.71	1300.88	994.60	1154.48	1197.57	1102.46	128.18	3.00	8.00	5.00	5.00	6.00	5.26	1.22
	Kw-ARNR	80	973.53	1328.41	997.25	1103.74	1226.54	1114.87	106.36	3.00	8.00	5.00	5.00	7.00	5.50	1.40
	Sc-AllV	80	768.61	1319.75	910.65	1067.97	1132.52	1039.67	157.62	3.00	7.00	4.00	5.00	5.00	4.85	0.93
	Sc-ARNR	80	768.61	1325.72	910.65	1070.03	1138.41	1041.65	158.56	3.00	8.00	4.00	5.00	6.00	5.01	1.13
RC2-M	Kw-AllV	80	1277.24	1468.06	1303.54	1338.17	1400.30	1356.66	71.00	3.00	6.00	4.75	5.00	5.00	4.75	0.83
	Kw-ARNR	80	1239.82	1574.53	1362.66	1403.34	1426.76	1396.68	95.90	4.00	8.00	5.00	6.00	6.25	5.88	1.17
	Sc-AllV	80	760.55	1297.44	928.16	1065.02	1161.32	1037.82	156.81	3.00	7.00	4.00	5.00	6.00	5.04	0.89
	Sc-ARNR	80	760.55	1297.44	928.16	1065.02	1172.00	1043.03	160.54	3.00	8.00	4.00	5.00	6.00	5.16	1.01
RC2-H	Kw-AllV	80	1461.02	1910.03	1519.97	1555.60	1782.80	1636.28	155.79	4.00	7.00	4.00	4.50	5.00	4.75	0.97
	Kw-ARNR	80	1461.02	2023.74	1556.12	1655.32	1768.64	1692.72	190.02	4.00	7.00	4.00	5.50	6.25	5.38	1.23
	Sc-AllV	80	760.63	1424.09	958.66	1066.09	1180.96	1057.01	175.31	3.00	7.00	4.00	5.00	6.00	4.92	0.91
	Sc-ARNR	80	760.63	1438.52	958.66	1066.93	1180.97	1063.05	178.39	3.00	7.00	4.00	5.00	6.00	5.03	0.95

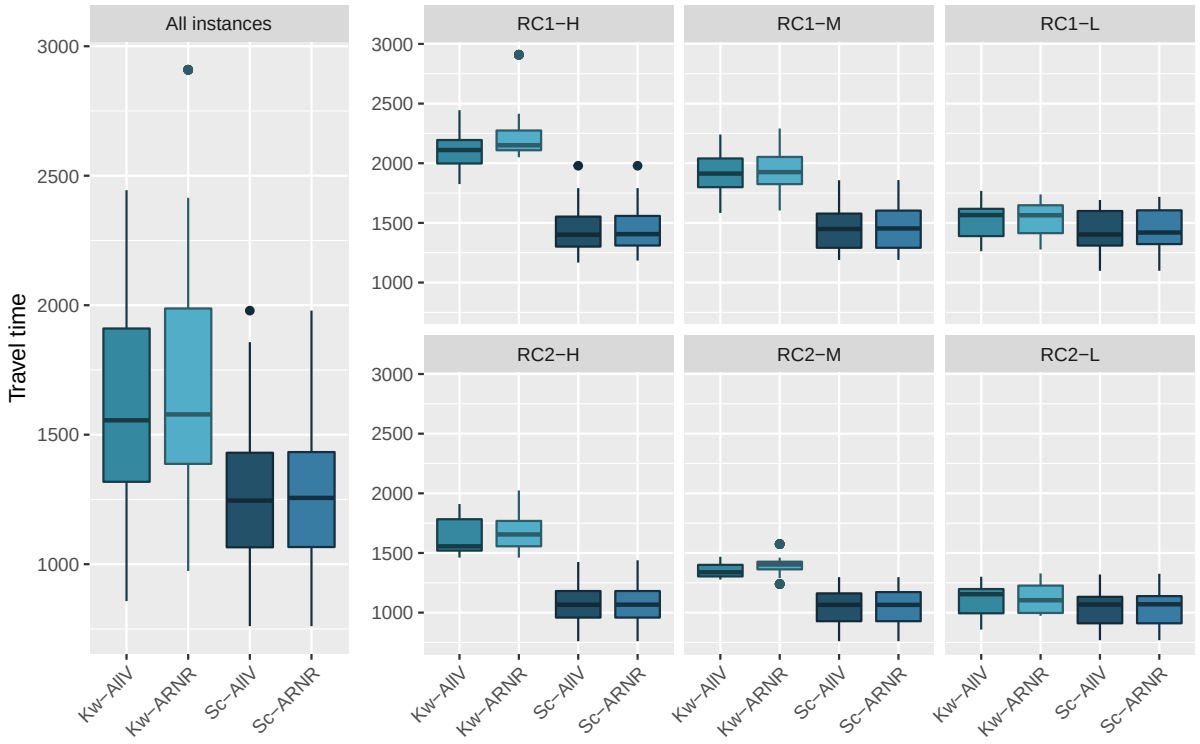


Figure 30 – Box plots: solutions in terms of travel time.

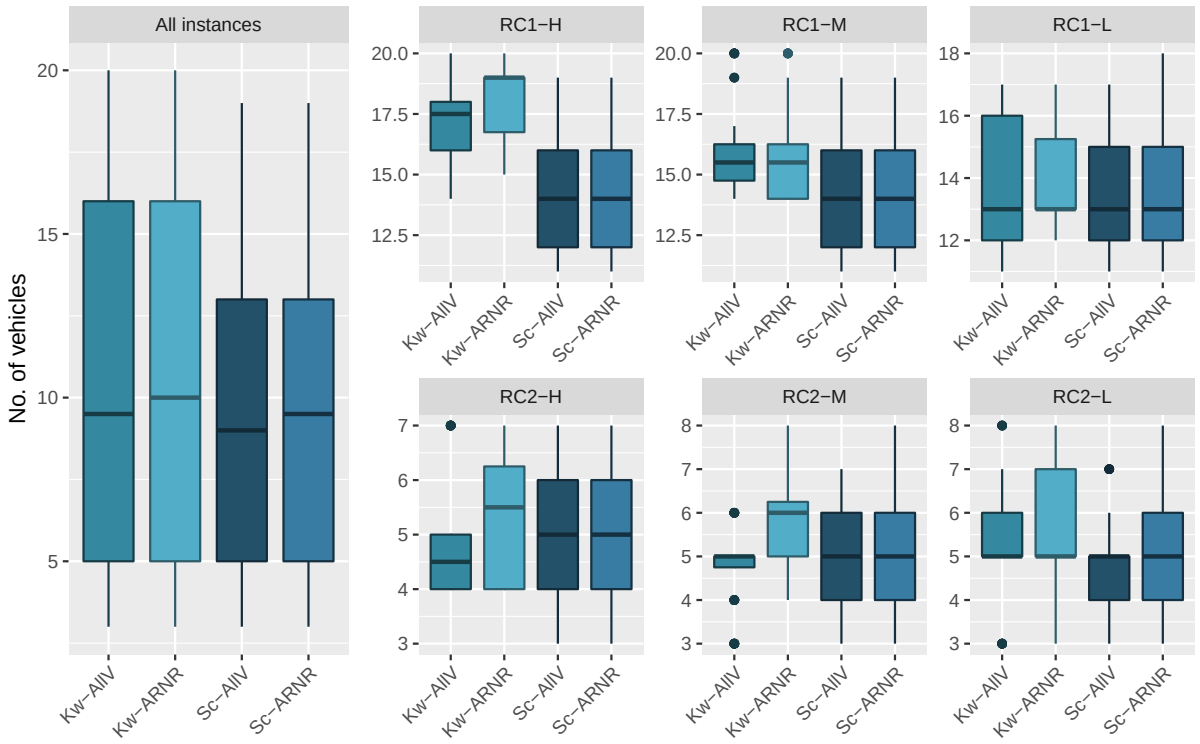


Figure 31 – Box plots: solutions in terms of number of vehicles.

Regarding the number of vehicles used, the Sc approaches outperformed the Kw approaches on instances of data sets RC1-H and RC1-M. For the other data sets, the performance differences between the four approaches were less notorious.

Table 14 – Best solutions found by each approach

Instance	Approach							
	Kw-AllV		Kw-ARNR		Sc-AllV		Sc-ARNR	
	TT	NV	TT	NV	TT	NV	TT	NV
RC101-L	1605.01	16	1646.44	16	1597.86	16	1597.86	16
RC102-L	1621.44	13	1566.52	13	1483.43	14	1487.98	14
RC103-L	1343.33	12	1372.56	12	1255.08	12	1259.36	12
RC104-L	1558.97	13	1558.97	13	1221.28	12	1226.92	12
RC105-L	1605.81	16	1647.35	16	1519.16	15	1531.93	15
RC106-L	1382.81	12	1401.08	13	1321.29	13	1337.53	13
RC107-L	1508.08	14	1560.91	14	1373.87	13	1403.93	13
RC108-L	1262.92	11	1276.66	12	1098.50	11	1098.50	11
RC101-M	2129.69	19	2179.09	19	1702.34	17	1702.34	17
RC102-M	1583.37	14	1603.89	14	1450.85	15	1450.85	15
RC103-M	1809.38	15	1852.36	16	1270.90	12	1272.22	12
RC104-M	1911.27	15	1855.08	15	1189.62	11	1189.62	11
RC105-M	1895.45	17	1987.43	17	1556.25	16	1556.25	16
RC106-M	1769.67	14	1742.57	14	1392.43	13	1406.69	13
RC107-M	1968.49	15	1912.49	14	1209.65	12	1209.65	12
RC108-M	2162.39	16	2163.32	16	1215.29	12	1215.29	12
RC101-H	2263.57	20	2227.27	19	1551.54	16	1555.71	16
RC102-H	2127.07	18	2150.81	19	1392.91	15	1392.91	15
RC103-H	2110.48	17	2414.68	19	1233.30	12	1237.39	12
RC104-H	1898.66	14	2079.75	15	1167.45	11	1184.21	11
RC105-H	1997.98	18	2050.26	19	1555.99	16	1566.28	16
RC106-H	1825.14	15	2122.86	17	1375.39	13	1382.27	13
RC107-H	2023.75	16	2108.92	16	1252.07	12	1253.41	12
RC108-H	2444.30	18	2909.15	20	1228.26	12	1228.26	12
RC201-L	1271.66	7	1220.78	8	1253.28	6	1253.28	6
RC202-L	1199.18	5	1226.54	5	1079.01	5	1079.01	5
RC203-L	990.14	5	995.00	5	889.37	4	889.37	4
RC204-L	857.71	5	973.54	5	787.72	4	787.72	4
RC205-L	1121.77	7	1103.74	7	1117.24	7	1124.41	8
RC206-L	1117.29	6	1126.63	7	1063.80	5	1066.44	5
RC207-L	1062.83	5	1069.47	5	996.23	5	996.23	5
RC208-L	996.09	3	996.09	3	768.61	4	768.61	4
RC201-M	1468.06	5	1415.23	7	1217.81	6	1217.81	6
RC202-M	1277.24	5	1387.40	6	1111.72	5	1111.72	5
RC203-M	1311.89	5	1402.29	6	911.35	4	911.35	4
RC204-M	1358.28	4	1461.36	5	765.81	4	765.81	4
RC205-M	1379.96	6	1404.39	8	1111.31	7	1133.63	7
RC206-M	1278.49	5	1288.43	6	1030.21	5	1038.00	5
RC207-M	1461.33	5	1574.53	5	979.13	5	979.13	5
RC208-M	1318.06	3	1239.82	4	760.56	5	760.56	5
RC201-H	1782.80	5	1732.19	7	1265.14	6	1266.83	7
RC202-H	1519.97	5	1578.44	4	1097.88	6	1097.88	6
RC203-H	1523.15	4	1515.86	6	950.30	5	950.30	5
RC204-H	1461.02	4	1461.02	4	777.84	4	777.84	4
RC205-H	1871.64	7	2023.74	7	1167.65	6	1167.65	6
RC206-H	1625.50	5	1737.60	5	1059.92	5	1059.92	5
RC207-H	1555.60	4	1569.54	6	961.45	6	961.45	6
RC208-H	1511.41	4	1861.77	4	760.63	4	760.63	4

The performance gaps between the Sc-ARNR approach and the other three approaches are presented in Table 15. Let $\bar{x}^{*}_{\text{cost}}$ be the average cost (measured in terms of travel time or number of vehicles used) attained by the Sc-ARNR approach on a given data set and let $\bar{x}_{\text{cost}}$ be the average cost attained by other approach on the same data set. The performance gap is given by $100 \cdot (\bar{x}^{*}_{\text{cost}} - \bar{x}_{\text{cost}})/\bar{x}_{\text{cost}}$.

On all data sets, the gaps in average travel times between the Sc-ARNR and Sc-AllV approaches were less than 0.75%. Overall, the gap in average travel time between both ap-

Table 15 – Performance gaps between Sc-ARNR and the other approaches

Data Set	Gap (%)					
	Avg. Travel time			Avg. No. of vehicles		
	Sc-AllV	Kw-ARNR	Kw-AllV	Sc-AllV	Kw-ARNR	Kw-AllV
RC1-L	0.72	-5.78	-4.80	-0.07	0.00	-0.14
RC1-M	0.49	-24.06	-23.62	0.07	-10.00	-10.23
RC1-H	0.34	-36.50	-32.11	0.07	-22.69	-18.59
RC2-L	0.19	-6.57	-5.52	3.30	-8.91	-4.75
RC2-M	0.50	-25.32	-23.12	2.38	-12.24	8.63
RC2-H	0.57	-37.20	-35.03	2.24	-6.51	5.89
All instances	0.48	-24.52	-22.32	0.74	-11.27	-7.12

proaches was 0.48%. On all data sets, the Sc-ARNR outperformed the Kw approaches. As previously commented, the performance improvements were more notorious on instances of medium and high dynamism, where the improvements reached 25.32% and 37.20%, respectively. Overall, the Sc-ARNR outperformed the Kw-ARNR and Kw-AllV approaches by 24.52% and 22.32%, respectively. Regarding the average number of vehicles used, the Sc-ARNR and Sc-AllV approaches performed similarly, the highest gap was 3.3% and it was recorded on the data set RC2-L. Overall, the performance gap between the Sc approaches remained small, 0.74%. The Sc-ARNR approach outperformed the Kw-ARNR approach on all data sets but one: RC1-L; whereas the Kw-AllV approach was outperformed on all data sets but two: RC2-M and RC2-H. Overall, the Sc-ARNR approach outperformed the Kw-ARNR and Kw-AllV approaches by 11.27% and 7.12%, respectively.

The obtained results suggest that the Sc-ARNR and the Sc-AllV approaches produce similar solutions in terms of travel time and number of vehicles used. The Sc-ARNR approach outperforms the Kw approaches on all data sets when assessing the solutions in terms of travel time, and the more noticeable performance gaps were obtained on instances of medium and high dynamism. On most of the data sets, the Sc-ARNR approach outperformed the Kw approaches in terms of number of vehicles used. Statistical tests were conducted to confirm these results. The Shapiro–Wilk test was applied to test the normality of the results obtained by each approach. In all cases, the collected data did not conform to normality, because all p-values were less than 0.05. Kruskal-Wallis tests with Dunn's post hoc tests were used to compare the results grouped by data set and the aggregate results. The p-values were corrected for multiple comparisons by an FDR correction procedure. At a 95% confidence level, the Kruskal–Wallis tests indicated that there is a significant difference in travel times between the obtained results (p-values < 0.05 for the results grouped by data set and the aggregate results). Regarding the number of vehicles used, the Kruskal–Wallis tests showed that there is a significant difference between the results obtained for all data sets but one (the tests performed on the aggregate results and the results for data sets RC2-L, RC1-M, RC2-M, RC1-H, and RC2-H yielded p-values < 0.05; the test performed on the results for data set RC1-L yielded a p-value of 0.98). The results of the Dunn's post hoc tests are presented in Table 16. Since the results related to number of vehicles used were not significantly

Table 16 – Dunn's corrected p-values: comparison of approaches to dealing with dynamism

Data Set		Travel time			No. of vehicles		
		Kw-AllV	Kw-ARNR	Sc-AllV	Kw-AllV	Kw-ARNR	Sc-AllV
RC1-L	Kw-ARNR	0.65	-	-	-	-	-
	Sc-AllV	< 0.05	< 0.05	-	-	-	-
	Sc-ARNR	< 0.05	< 0.05	0.77	-	-	-
RC1-M	Kw-ARNR	0.77	-	-	0.96	-	-
	Sc-AllV	< 0.05	< 0.05	-	< 0.05	< 0.05	-
	Sc-ARNR	< 0.05	< 0.05	0.91	< 0.05	< 0.05	0.98
RC1-H	Kw-ARNR	0.19	-	-	< 0.05	-	-
	Sc-AllV	< 0.05	< 0.05	-	< 0.05	< 0.05	-
	Sc-ARNR	< 0.05	< 0.05	0.91	< 0.05	< 0.05	0.98
RC2-L	Kw-ARNR	0.77	-	-	0.57	-	-
	Sc-AllV	< 0.05	< 0.05	-	< 0.05	< 0.05	-
	Sc-ARNR	< 0.05	< 0.05	0.91	0.08	< 0.05	0.66
RC2-M	Kw-ARNR	0.20	-	-	< 0.05	-	-
	Sc-AllV	< 0.05	< 0.05	-	0.20	< 0.05	-
	Sc-ARNR	< 0.05	< 0.05	0.91	0.05	< 0.05	0.69
RC2-H	Kw-ARNR	0.57	-	-	< 0.05	-	-
	Sc-AllV	< 0.05	< 0.05	-	0.23	0.07	-
	Sc-ARNR	< 0.05	< 0.05	0.91	0.08	0.22	0.73
All instances	Kw-ARNR	0.37	-	-	< 0.05	-	-
	Sc-AllV	< 0.05	< 0.05	-	< 0.05	< 0.05	-
	Sc-ARNR	< 0.05	< 0.05	0.84	0.05	< 0.05	0.87

different for the RC1-L data set, no test was performed in that case. Regarding the travel time of the solutions, the Dunn's tests indicate that there is no statistically significant difference between the solutions found by the Sc-ARNR and Sc-AllV approaches (p-values > 0.05 in all cases). The tests also show that there is statistically significant difference between the solutions found by the Sc and Kw approaches (p-values < 0.05 in all cases). Regarding the number of vehicles used, the tests indicate that there is no statistically significant difference between the Sc-ARNR and Sc-AllV approaches (p-values > 0.05 in all cases). Significant differences were found between the Sc-ARNR and Kw-AllV approaches in results for data sets RC1-M and RC1-H. Significant differences were also found between the Sc-ARNR and Kw-ARNR approaches in results for data sets RC1-M, RC1-H, RC2-L, RC2-M, and the aggregate results.

The analysis of the descriptive statistics and the results of the statistical tests offer evidence that the Sc-ARNR and Sc-AllV approaches perform similarly both in terms of travel time and number of vehicles used. The Sc-ARNR approach outperforms the Kw approaches on all data sets regarding travel time, and on some data sets regarding number of vehicles used.

4.4.3 Impact of the ARNR procedure in reducing the number of vehicles considered for serving unanticipated dynamic requests

The aim of the ARNR procedure is to reduce the number of vehicles considered for serving dynamic requests that were not anticipated in the initial routing plan, without compromising the final solution quality regarding total travel time. Since the Sc-ARNR and Sc-AllV approaches

produce solutions of similar quality, there is evidence that the use of the ARNR procedure does not compromise solution quality when compared with an exhaustive search-based approach. Now, it is necessary to assess the impact of the ARNR procedure in reducing the number of vehicles considered for serving unanticipated dynamic requests.

For each experiment conducted with the Sc-ARNR and Sc-AllV approaches, the average number of vehicles considered for serving unanticipated dynamic requests (*nvd*) was computed by dividing the total number of vehicles considered for serving all unanticipated dynamic requests (a vehicle could be counted more than once) between the total number of unanticipated dynamic requests. In each experiment performed on a problem instance, the same initial routing plan was used by the Sc-ARNR and Sc-AllV approaches. Thus, both approaches dealt with the same number of unanticipated dynamic requests.

The descriptive statistics for the *nvd* values are presented in Table 17 and Figure 32. The results show that both approaches considered fewer vehicles to serve unanticipated dynamic requests in RC2 instances. This is a predictable result, since in RC2 instances the vehicle capacity is larger and time windows are wider than in RC1 instances. Given that a single vehicle can serve more customers than in RC1 instances, fewer vehicles are assigned a route, and thus, fewer active vehicles end up being considered for assigning a request. In all cases, the Sc-ARNR approach noticeably outperformed the Sc-AllV approach. The gaps between the average *nvd* values are presented in Table 18. Let $\bar{x}^{*}_{nvd}$ be the average *nvd* value obtained by the Sc-ARNR approach on a given data set and let $\bar{x}_{nvd}$ be the average *nvd* value obtained by the Sc-AllV approach on the same data set. The gap is calculated as $100 \cdot (\bar{x}^{*}_{nvd} - \bar{x}_{nvd})/\bar{x}_{nvd}$. For all data sets, the Sc-ARNR approach consistently reduced the average *nvd* values by nearly 50% with respect to the Sc-AllV approach, i.e, the average number of vehicles considered for serving unanticipated dynamic requests decreased by nearly half when the ARNR procedure was used.

Statistical tests were conducted to confirm the previously analyzed results. First, the

Table 17 – Descriptive statistics: average number of vehicles considered for serving unanticipated dynamic requests

Data set	Approach	Statistics							
		Obs	Min	Max	1st Q.	Median	3rd Q.	Mean	St Dv.
RC1-L	Sc-AllV	80	9.78	16.50	11.47	12.55	14.74	13.02	1.92
	Sc-ARNR	80	2.00	9.75	4.59	5.76	6.68	5.61	1.91
RC1-M	Sc-AllV	80	10.00	17.11	11.86	12.73	15.29	13.40	1.97
	Sc-ARNR	80	2.50	12.50	4.85	6.31	7.75	6.48	2.19
RC1-H	Sc-AllV	80	10.09	17.43	11.96	13.00	15.21	13.35	1.92
	Sc-ARNR	80	2.00	14.00	4.31	5.55	6.60	5.74	2.27
RC2-L	Sc-AllV	80	3.00	7.00	4.00	4.90	5.00	4.67	0.95
	Sc-ARNR	80	2.00	3.25	2.17	2.33	2.51	2.39	0.31
RC2-M	Sc-AllV	80	2.89	6.67	4.00	5.00	5.14	4.76	0.84
	Sc-ARNR	80	2.00	3.00	2.14	2.31	2.50	2.37	0.30
RC2-H	Sc-AllV	80	3.00	6.17	4.00	4.88	5.00	4.75	0.78
	Sc-ARNR	80	2.00	3.50	2.12	2.26	2.50	2.34	0.32
All instances	Sc-AllV	480	2.89	17.43	5.00	8.39	12.81	8.99	4.52
	Sc-ARNR	480	2.00	14.00	2.29	2.95	5.78	4.16	2.36

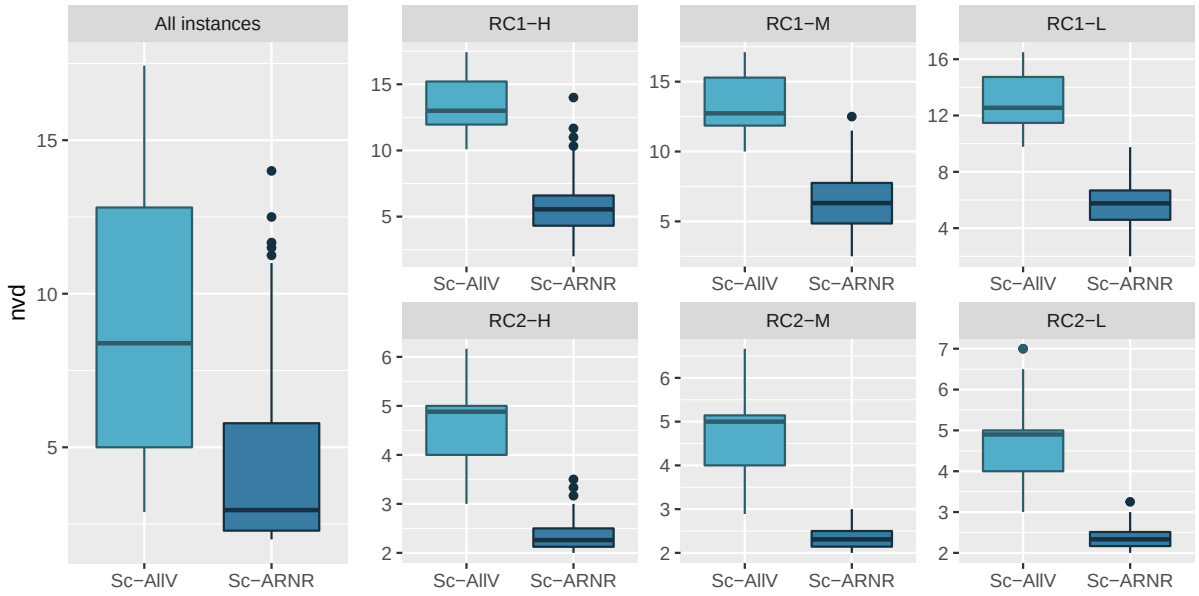


Figure 32 – Box plots: average number of vehicles considered for serving unanticipated dynamic requests.

Table 18 – Gaps between the average *nvd* values obtained by the Sc-ARNR and Sc-AllV approaches

Data set	RC1-L	RC1-M	RC1-H	RC2-L	RC2-M	RC2-H	All instances
Gap (%)	-56.91	-51.64	-57.00	-48.82	-50.21	-50.74	-53.73

Shapiro–Wilk test was applied to test the normality of the results grouped by data set and the aggregate results. In all cases, the data did not conform to normality, because all p-values were less than 0.05. The results were then compared using the Mann–Whitney U test, performing an FDR correction procedure on the p-values. At a 95% confidence level, the Mann–Whitney U tests indicated that there is statistically significant difference between the results produced by both approaches for each data set and for the aggregate results (all p-values were less than 0.05).

The presented results offer evidence that, when compared with an approach based on exhaustive search, the ARNR procedure significantly reduces the average number of vehicles considered for assigning unanticipated dynamic requests. In the conducted experiments, the reduction was close to 50% for all data sets.

4.4.4 Online vs offline solutions

The proposed approach builds a solution for a given DVRPTWSC instance in an online manner. The offline solution for the dynamic problem can be computed when all the information becomes disclosed. In this case, the offline solution for a DVRPTWSC instance can be obtained by solving a static VRPTW instance containing all the customers that end up making requests.

The value of information, proposed by Mitrović-Minić, Krishnamurti and Laporte (2004), measures the effectiveness of a method in solving a dynamic problem. The value of information represents the gain obtained from solving a dynamic problem when all information is known in

advance, and it is calculated as the cost gap between the online and offline solutions produced for the dynamic problem. Let S and S' be, respectively, the online and offline solutions for a dynamic problem. The value of information with respect to the objective function ψ is given by: $(\psi(S) - \psi(S'))/\psi(S)$.

The MA and the parameter settings presented in Chapter 3 were employed to find the offline solutions for the DVRPTWSC instances. Each instance was solved 10 times. For all data sets, the travel times associated with the online and offline solutions are reported in Table 19 and Figure 33. As expected, the offline solutions were better for all data sets. At a 95% confidence level, Mann–Whitney U tests with FDR correction indicated that the difference between the offline and online solutions is statistically significant (p-values < 0.05 for the results grouped by data set and the aggregate results).

Table 19 – Descriptive Statistics: solutions to offline and online instances (in terms of travel time)

Data set	Approach	Statistics							
		Obs	Min	Max	1st Q.	Median	3rd Q.	Mean	St Dv.
RC1-L	MA-offline	80	1039.85	1564.20	1117.87	1227.00	1410.04	1260.74	179.74
	Sc-ARNR	80	1098.50	1718.32	1322.52	1419.30	1605.43	1445.58	166.34
RC1-M	MA-offline	80	1043.18	1603.46	1101.05	1247.96	1367.18	1260.32	183.24
	Sc-ARNR	80	1189.62	1858.98	1291.49	1453.42	1603.06	1474.82	182.26
RC1-H	MA-offline	80	1035.25	1483.12	1052.93	1200.75	1305.64	1210.87	157.01
	Sc-ARNR	80	1184.21	1979.10	1310.39	1405.78	1558.35	1437.22	164.24
RC2-L	MA-offline	80	721.66	1191.50	813.87	966.23	1020.32	931.00	152.03
	Sc-ARNR	80	768.61	1325.72	910.65	1070.03	1138.41	1041.65	158.56
RC2-M	MA-offline	80	733.95	1192.31	855.43	941.78	1047.82	944.41	145.62
	Sc-ARNR	80	760.55	1297.44	928.16	1065.02	1172.00	1043.03	160.54
RC2-H	MA-offline	80	725.50	1234.32	859.54	978.90	1049.38	962.63	158.39
	Sc-ARNR	80	760.63	1438.52	958.66	1066.93	1180.97	1063.05	178.39
All	MA-offline	480	721.66	1603.46	966.67	1053.04	1230.39	1094.99	221.31
instances	Sc-ARNR	480	760.55	1979.10	1066.19	1255.80	1432.91	1250.89	262.80

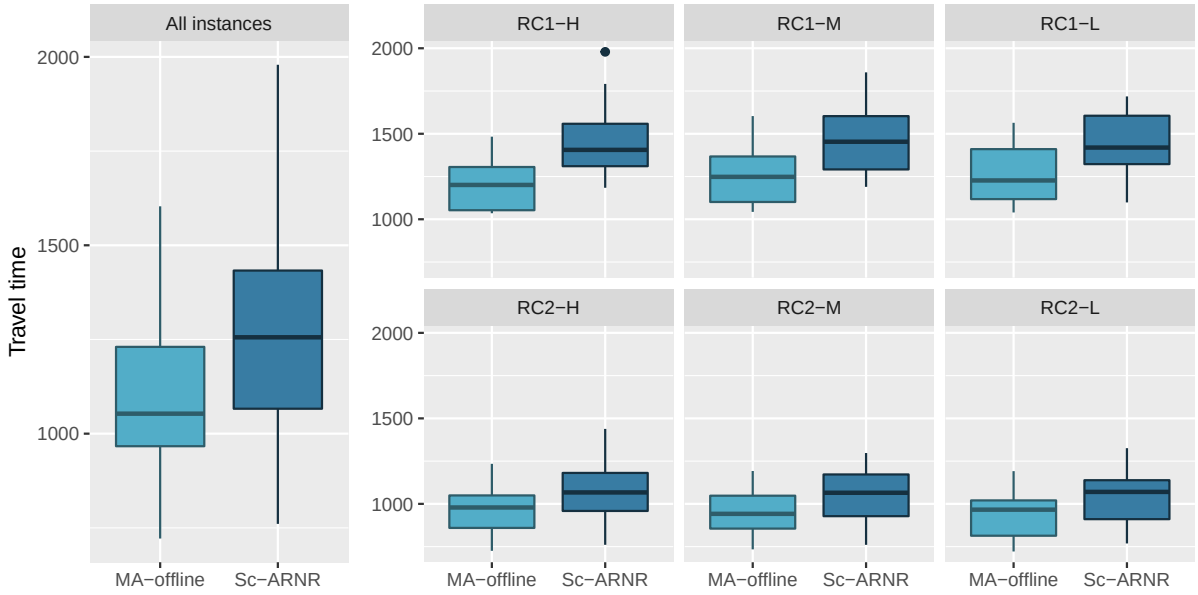


Figure 33 – Box plots: solutions to offline and online instances.

Table 20 – Values of information with respect to average travel times

Data set	RC1-L	RC1-M	RC1-H	RC2-L	RC2-M	RC2-H	All instances
Value of information	0.13	0.15	0.16	0.11	0.09	0.09	0.12

Table 20 presents the values of information with respect to the average travel times obtained for the instances of the benchmark set. Let $\bar{x}^*_{tt}$ be the average travel time obtained by the Sc-ARNR approach on a given data set and let $\bar{x}_{tt}$ be the average travel time of the offline solutions found by the MA for the same data set. The value of information with respect to the average travel time is given by $(\bar{x}^*_{tt} - \bar{x}_{tt})/\bar{x}^*_{tt}$. The values of information were similar for all data sets, although they were a bit smaller for RC2 instances. The level of dynamism of the instances appeared to have little influence on the values of information. Indeed, the values of information obtained for data sets RC2-M and RC2-H were a little bit smaller than that obtained for data set RC2-L. The results suggest that the proposed approach performs similarly under different levels of dynamism.

To further investigate the impact that the level of dynamism has on the performance of the proposed approach, the correlation between level of dynamism and value of information was assessed for all instances. The level of dynamism for each instance was computed using the measure proposed by Lund, Madsen and Rygaard (1996), which defines the level of dynamism δ as the ratio of the number of dynamic requests n_d to the total number of requests n_t: $\delta = n_d/n_t$. The value of information with respect to average travel time was computed for each instance considering the 10 online solutions produced by the Sc-ARNR and the 10 offline solutions

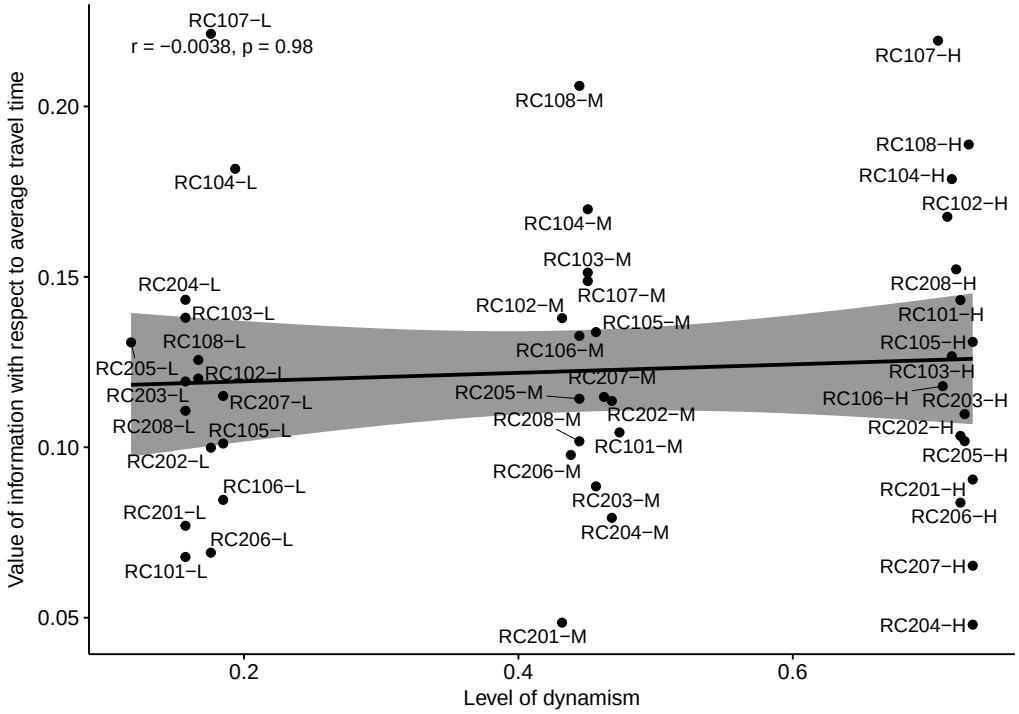


Figure 34 – Correlation between value of information and level of dynamism.

produced by the MA. The Spearman correlation test was employed to assess the association between the value of information and the level of dynamism for all instances. The result is presented in Figure 34. The scatter-plot shows that there is not a strong correlation between the value of information and the level of dynamism of the instances. For the three levels of dynamism, the values of information of the instances range between similar values. The correlation coefficient is $r = -0.0038$ and the p-value of the test is $0.98 > 0.05$. This indicates that the value of information and the level of dynamism of the instances are not significantly correlated. This result confirms that the proposed approach performs consistently under the three considered levels of dynamism.

4.5 Conclusions

This chapter studied the DVRPTWSC. In this problem, the customers are classified as static or dynamic depending on the time they make their requests. Static customers make their requests before the start of the planning horizon and dynamic customers may or may not make requests over the planning horizon. The probability that a dynamic customer makes a request is known. The objective is to define a set of actions that allows a dispatcher to assign requests to a fleet of vehicles based at a single depot in such a way that: the total travel time is minimized, the number of dynamic requests served is maximized, all static requests are served, customers are visited at most once, all vehicles start and end their routes at the depot, and time-window and vehicle-capacity constraints are satisfied.

A cartographic approach to the DVRPTWSC was described. The proposed approach was incorporated into a multiagent system, where a dispatcher agent receives service requests from several customer agents and plans the routes for vehicle agents, who update their routing information and follow the dispatcher's instructions. Geographic information from the customers is used by the dispatcher for creating the initial routing plan and for processing dynamic requests.

Prior to the start of the planning horizon, a cartographic processing is performed. Employing hierarchical clustering, the geographical region where customers are located is divided into a hierarchy of nested regions, and each customer is assigned a sequence of nested regions. The dispatcher agent creates an initial routing plan including known and likely requests. A slightly modified version of the MA proposed in Chapter 3 is employed to create the initial routing plan. The output of the hierarchical clustering procedure is used by the MA to perform the search. A pool containing several routing scenarios, which include possible realizations of the requests, is optimized during a fixed number of iterations. Using a similarity measure, one routing scenario is selected as initial routing plan.

Over the planning horizon, the dispatcher orders the vehicles to update their routes as new information becomes known. Some requests that were included in the initial routing plan and do not materialize must be removed. New requests that were not considered in the initial

routing plan must be assigned to a vehicle or must be rejected if they cannot be feasibly served. The dispatcher aims to assign the new requests to the vehicles capable of feasibly serve them at the lowest detour costs. Given that calculating the detour cost for several vehicles could be a computationally expensive operation, the dispatcher seeks to reduce as much as possible the number of vehicles considered for serving each new request. To this aim, the ARNR procedure was proposed. In a first stage, the ARNR procedure identifies a set of candidate vehicles that currently are or will be near the customer who requested service, and it tries to assign the request to the candidate vehicle that can feasibly serve the customer at the lowest detour cost. The sequence of nested regions associated with the customer is used to identify the vehicles candidates to serve the request. If the request cannot be assigned in the first stage, the vehicles that were not considered yet become candidates to serve the request. If the request still cannot be assigned, the procedure checks if there still are idle vehicles available and if the customer can be served without lateness. If both conditions are met, the request is assigned to an idle vehicle, otherwise, the request is rejected.

Computational experiments were conducted on DVRPTWSC instances derived from the Solomon's benchmark set. The results offer several insights into the performance of the proposed approach. The proposed reoptimization method significantly improves the quality of the routing scenarios created with the Solomon's NN heuristic. The exploitation of stochastic information related to possible future requests can significantly improve the quality of the produced solutions. Assessing the results in terms of travel time, solution approaches that included likely but unconfirmed requests in the initial routing plan significantly outperformed approaches that only considered known requests in the initial routing plan. The improvements were more noticeable on instances of medium and high dynamism. The ARNR procedure success at identifying the vehicles yielding the lowest detour costs for serving new requests. The solutions produced by the proposed approach were similar in terms of travel time and number of vehicles used to the solutions produced by an exhaustive search-based approach, which considers all vehicles for serving dynamic requests. Furthermore, the use of the ARNR procedure significantly reduces the number of vehicles considered for serving new requests in comparison with the exhaustive search-based approach. In the experiments conducted, the ARNR procedure reduced by nearly half the average number of vehicles considered for serving dynamic requests. Thus, the ARNR procedure achieves the objectives for which it was conceived: its use leads to solutions of similar quality to those produced by using an exhaustive search-based approach, and the number of vehicles considered for serving new requests is significantly reduced. The proposed solution approach performs consistently under the three considered levels of dynamism: low, medium, and high. In the experiments conducted, the value of information with respect to average travel time over all instances was equal to 0.12.

Directions for further research include the development of new techniques to divide the geographical region into subregions. If the proposed approach is adapted to being used in a real problem, an ad hoc geographical division could be performed with aid of a geographic

information system (GIS). In the proposed approach, the detour costs for assigning requests to the vehicles are calculated by employing an exhaustive search-based method. More sophisticated heuristics could be developed to perform this task. In the proposed multiagent system, the dispatcher plans the routes having access to global information about the problem and the vehicles just follow instructions and update their routing information. Future work could adapt the proposed solution methodology to a decentralized approach, where vehicles and customers interact directly. When a customer makes a request, vehicles calculate locally the cost of serving the request and then they bid their costs in an auction to select the vehicle that will serve the customer. In this case, vehicle agents should become "smarter", since they should be capable of optimizing their own routes and anticipating requests, i.e., the vehicles would need to implement a logic similar to that currently implemented in the MA. To optimize their routes, vehicles could employ heuristics to perform intra-route movements. Furthermore, they could autonomously interchange requests with other vehicles, emulating inter-route moves. This sort of decentralized improvement mechanisms has been previously used by authors like Kohout and Erol (1999). To anticipate potential requests, vehicles could use a sampling-based approach, like that used in the proposed solution methodology, or they could make predictions based on learning mechanisms.

CHAPTER

5

CONCLUSIONS AND OUTLOOK

This dissertation presented a cartographic approach to the DVRPTWSC. The objectives of the problem are to minimize the total travel time and maximize the number of dynamic requests served. The main characteristic of the proposed approach is the use of customers' spatial information in creating and updating the routing plan. Computational results provide proof of concept for the proposed solution methodology.

In Chapter 3, the VRPTW was addressed aiming to minimize the total travel time. A simple yet effective MA was proposed to solve this problem. The main search operators of the MA are the CB-BCRC and the CB-Inter-Route-NS. Both operators create new solutions by exchanging customers between routes passing close to each other, seeking to minimize the detour costs associated with the modifications. Clustering arrangements of customers are used to identify the routes undergoing modifications.

Computational results indicate that the proposed MA produces high-quality solutions to the VRPTW, rendering it competitive with state-of-the-art heuristics. The CB-BCRC operator can produce solutions identical to those produced by a similar crossover operator based on exhaustive search while requiring shorter execution times. The main factor responsible for the quality of the solutions produced by the MA is the CB-BCRC operator.

In Chapter 4, the cartographic approach to the DVRPTWSC was presented. The proposed approach is incorporated into a multiagent system composed of three kinds of agents: dispatcher, vehicle, and customer. The dispatcher receives service requests from the customers and plans the routes that the vehicles must follow. The dispatcher creates an initial routing plan containing all known requests and some likely requests and updates the routing plan in real time according to the information revealed over the planning horizon: some requests that do not materialize are removed and new requests that can be feasibly served are added to the routing plan.

The first action the dispatcher takes is to perform a cartographic processing of the customers' spatial data. This procedure clusters the customers based on their spatial distance

and associates each customer with a hierarchy of nested regions. This is the main step of the proposed approach because the outputs of the cartographic processing are used for creating the initial routing plan and processing new requests.

A slightly modified version of the MA presented in Chapter 3 is used to create the initial routing plan. In this case, a pool of routing scenarios (solutions to VRPTW instances containing all known requests and potential requests) is created using a construction heuristic and then it is reoptimized during a fixed number of iterations by the same search operators of the original MA. The clusters created in the cartographic processing are used by the search operators of the MA. Once the reoptimization process is completed, the routing scenario most similar to the other routing scenarios in the pool is selected as the initial routing plan.

Whenever a new request is received, the dispatcher processes it using the ARNR. This procedure scans the hierarchy of nested regions associated with the customer requiring service, looking for the vehicle capable of serving the request at the lowest detour cost. The nested regions are used to identify the vehicles that currently are or will be close to the customer; such vehicles are expected to serve the request at lower detour costs than vehicles located farther from the customer.

Computational results show that the reoptimization process significantly improves the quality of the routing scenarios created with the construction heuristic. The exploitation of stochastic information related to potential requests can significantly improve the quality of the produced solutions. When compared with an exhaustive search-based approach, the ARNR procedure shows remarkable results: it consistently identifies vehicles yielding low detour costs to serve new requests and it considers significantly fewer candidate vehicles. The proposed solution approach performs consistently under three levels of dynamism: low, medium, and high.

There are several avenues for further research. Regarding the MA, comparative studies can be carried out to assess the performance of the CB-BCRC and CB-Inter-Route-NS operators against other search operators of the literature, the MA can also be applied to other variants of VRP, and other clustering techniques can be implemented to guide the search. Regarding the cartographic approach, the cartographic processing can be performed using ad hoc approaches based on real geographical data, more sophisticated heuristics can be implemented to calculate the detour costs, and local search mechanisms can be implemented to improve the routes of the vehicles in operation. Yet the most interesting direction for further research is the conversion of the proposed solution approach into a decentralized multiagent system, letting vehicle agents take their own decisions. This would imply addressing several coordination issues, but it certainly would extend the scope of the proposed solution approach to transportation problems of a decentralized nature, like those arising in crowd-based logistics.

BIBLIOGRAPHY

ALVARENGA, G.; MATEUS, G.; TOMI, G. de. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. **Computers & Operations Research**, v. 34, n. 6, p. 1561–1584, jun 2007. ISSN 03050548. Citations on pages 19, 28, 29, 30, 55, 56, and 57.

BANKER, S. The UPS-Workhorse Group Deal Highlights Advances In Last-Mile Routing. **Forbes**, 2018. Available: <https://www.forbes.com/sites/stevebanker/2018/02/23/the-upsworkhorse-group-deal-highlights-advances-in-last-mile-routing/#6a56702555ed>. Accessed: 2018-05-16. Citation on page 24.

BARKAOUI, M. A co-evolutionary approach using information about future requests for dynamic vehicle routing problem with soft time windows. **Memetic Computing**, apr 2017. ISSN 1865-9292. Citations on pages 25 and 36.

BARKAOUI, M.; BERGER, J.; BOUKHTOUTA, A. Customer satisfaction in dynamic vehicle routing problem with time windows. **Applied Soft Computing**, v. 35, p. 423–432, 2015. ISSN 1568-4946. Citation on page 36.

BEKTAş, T.; REPOUSSIS, P. P.; TARANTILIS, C. D. Dynamic Vehicle Routing Problems. In: TOTH, P.; VIGO, D. (Ed.). **Vehicle Routing: Problems, Methods, and Applications**. 2nd. ed. Philadelphia, PA: SIAM - Society for Industrial and Applied Mathematics, 2014. chap. 11, p. 299–347. ISBN 978-1-61197-358-7. Citations on pages 23, 27, 31, and 32.

BELFIORE, P.; YOSHIZAKI, Y.; TSUGUNOBU, H. Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil. **European Journal of Operational Research**, v. 199, n. 3, p. 750–758, dec 2009. ISSN 03772217. Citation on page 28.

BENT, R.; Van Hentenryck, P. A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. **Transportation Science**, v. 38, n. 4, p. 515–530, 2004. Citation on page 35.

____. Regrets Only! Online Stochastic Optimization Under Time Constraints. In: **Proceedings of the 19th National Conference on Artifical Intelligence**. San Jose, CA: AAAI Press, 2004. (AAAI'04), p. 501–506. ISBN 0-262-51183-5. Citation on page 35.

____. Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers. **Operations Research**, v. 52, n. 6, p. 977–987, 2004. Citations on pages 25, 35, and 69.

____. Waiting and Relocation Strategies in Online Stochastic Vehicle Routing. In: **Proceedings of the 20th International Joint Conference on Artifical Intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. (IJCAI'07), p. 1816–1821. Citation on page 35.

BLOCHO, M.; CZECH, Z. J. A Parallel Memetic Algorithm for the Vehicle Routing Problem with Time Windows. In: **2013 Eighth International Conference on P2P, Parallel, Grid, Cloud**

and Internet Computing. Compiègne, France: IEEE Press, 2013. p. 144–151. Citations on pages 28 and 29.

BRÄYSY, O.; GENDREAU, M. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. **Transportation Science**, v. 39, n. 1, p. 104–118, 2005. Citation on page 27.

____. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. **Transportation Science**, v. 39, n. 1, p. 119–139, 2005. Citation on page 27.

BRÄYSY, O.; HASLE, G. Software Tools and Emerging Technologies for Vehicle Routing and Intermodal Transportation. In: ____. **Vehicle Routing: Problems, Methods, and Applications**. 2nd. ed. Philadelphia, PA: SIAM - Society for Industrial and Applied Mathematics, 2014. chap. 12, p. 351–380. ISBN 978-1-61197-358-7. Citations on pages 23 and 33.

Buldeo Rai, H.; VERLINDE, S.; MERCKX, J.; MACHARIS, C. Crowd logistics: an opportunity for more sustainable urban freight transport? **European Transport Research Review**, v. 9, n. 3, p. 39, jul 2017. ISSN 1866-8887. Citation on page 34.

CHIANG, W.-C.; RUSSELL, R. A. A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows. **INFORMS Journal on Computing**, v. 9, n. 4, p. 417–430, 1997. Citations on pages 19, 56, and 57.

____. Integrating purchasing and routing in a propane gas supply chain. **European Journal of Operational Research**, v. 154, n. 3, p. 710–729, 2004. ISSN 0377-2217. Citation on page 28.

COOK, T. M.; RUSSELL, R. A. A SIMULATION AND STATISTICAL ANALYSIS OF STOCHASTIC VEHICLE ROUTING WITH TIMING CONSTRAINTS. **Decision Sciences**, v. 9, n. 4, p. 673–687, 1978. Citation on page 27.

CORDEAU, J.-F.; LAPORTE, G.; MERCIER, A. A unified tabu search heuristic for vehicle routing problems with time windows. **Journal of the Operational Research Society**, v. 52, n. 8, p. 928–936, 2001. ISSN 1476-9360. Citations on pages 19, 57, and 58.

CORDEAU, J.-F.; LAPORTE, G.; SAVELSBERGH, M. W. P.; VIGO, D. Vehicle Routing. In: BARNHART, C.; LAPORTE, G. (Ed.). **Transportation**. Amsterdam, the Netherlands: Elsevier, 2007, (Handbooks in Operations Research and Management Science, v. 14). chap. 6, p. 367–428. Citation on page 33.

DAN, Z.; CAI, L.; ZHENG, L. Improved multi-agent system for the vehicle routing problem with time windows. **Tsinghua Science and Technology**, v. 14, n. 3, p. 407–412, 2009. Citation on page 84.

DANTZIG, G. B.; RAMSER, J. H. The Truck Dispatching Problem. **Management Science**, v. 6, p. 80–91, 1959. ISSN 0025-1909. Citation on page 23.

DESAULNIERS, G.; MADSEN, O. B. G.; ROPKE, S. The Vehicle Routing Problem with Time Windows. In: TOTH, P.; VIGO, D. (Ed.). **Vehicle Routing: Problems, Methods, and Applications**. 2nd. ed. Philadelphia, PA: SIAM - Society for Industrial and Applied Mathematics, 2014. chap. 5, p. 119–159. ISBN 978-1-61197-358-7. Citations on pages 27, 28, 40, and 46.

DICE, L. R. . Measures of the Amount of Ecologic Association Between Species. **Ecology**, v. 26, n. 3, p. 297–302, 1945. ISSN 00129658. Citation on page 71.

DOERNER, K. F.; HARTL, R. F. Health Care Logistics, Emergency Preparedness, and Disaster Relief: New Challenges for Routing Problems with a Focus on the Austrian Situation. In: ____. **The Vehicle Routing Problem: Latest Advances and New Challenges**. Boston, MA: Springer US, 2008. p. 527–550. ISBN 978-0-387-77778-8. Citation on page 33.

FISCHER, K.; MÜLLER, J.; PISCHEL, M. Cooperative transportation scheduling: An application domain for dai. **Applied Artificial Intelligence**, Taylor & Francis, v. 10, n. 1, p. 1–34, 1996. Citations on pages 33 and 34.

FUNKE, B.; GRÜNERT, T.; IRNICH, S. Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration. **Journal of Heuristics**, v. 11, n. 4, p. 267–306, jul 2005. ISSN 1572-9397. Citations on pages 46 and 49.

GARCIA-NAJERA, A.; BULLINARIA, J. A. An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. **Computers & Operations Research**, v. 38, n. 1, p. 287–300, jan 2011. ISSN 03050548. Citations on pages 19, 28, 55, 56, and 71.

GATH, M. **Optimizing transport logistics processes with multiagent planning and control**. Bremen, Germany: Springer Vieweg, 2016. ISBN 978-3-658-14003-8. Citations on pages 33 and 34.

GEHRING, H.; HOMBERGER, J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: CITESEER. **Proceedings of EUROGEN99**. Jyväskylä, Finland, 1999. v. 2, p. 57–64. Citations on pages 28 and 29.

GENDREAU, M.; GUERTIN, F.; POTVIN, J.-Y.; TAILLARD, É. Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching. **Transportation Science**, v. 33, n. 4, p. 381–390, 1999. Citation on page 73.

HVATTUM, L. M.; LØKKETANGEN, A.; LAPORTE, G. Solving a Dynamic and Stochastic Vehicle Routing Problem with a Sample Scenario Hedging Heuristic. **Transportation Science**, v. 40, n. 4, p. 421–438, 2006. Citations on pages 25, 35, and 71.

____. A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. **Networks**, Wiley Subscription Services, Inc., A Wiley Company, v. 49, n. 4, p. 330–340, 2007. ISSN 1097-0037. Citation on page 36.

ISAAC, M. One Surprise Standout for Uber: Food Delivery. **The New York Times**, 2017. Available: <https://www.nytimes.com/2017/09/23/technology/ubereats-food-delivery.html>. Accessed: 2018-05-16. Citation on page 24.

JUNG, S.; MOON, B.-R. A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. In: **Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. (GECCO'02), p. 1309–1316. ISBN 1-55860-878-8. Citations on pages 19, 28, 29, 55, 56, and 57.

KIM, B.-I.; KIM, S.; SAHOO, S. Waste collection vehicle routing problem with time windows. **Computers & Operations Research**, v. 33, n. 12, p. 3624–3642, dec 2006. ISSN 03050548. Citation on page 28.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science (New York, N.Y.)**, v. 220, n. 4598, p. 671–680, 1983. ISSN 0036-8075. Citation on page 49.

KNIGHT, K. W.; HOFER, J. P. Vehicle Scheduling with Timed and Connected Calls: A Case Study. **Journal of the Operational Research Society**, v. 19, n. 3, p. 299–310, sep 1968. ISSN 1476-9360. Citation on page 27.

KOHOUT, R.; EROL, K. In-time Agent-based Vehicle Routing with a Stochastic Improvement Heuristic. In: **Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence**. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1999. (AAAI '99/IAAI '99), p. 864–869. ISBN 0-262-51106-1. Citations on pages 33 and 102.

LUND, K.; MADSEN, O.; RYGAARD, J. **Vehicle routing problems with varying degrees of dynamism**. Lyngby, Denmark, 1996. Citation on page 99.

MÁHR, T.; SROUR, J.; WEERDT, M. de; ZUIDWIJK, R. Can agents measure up? A comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. **Transportation Research Part C: Emerging Technologies**, v. 18, n. 1, p. 99–119, 2010. ISSN 0968-090X. Citations on pages 33 and 34.

MECHTI, R.; POUJADE, S.; ROUCAIROL, C.; LEMARIÉ, B. Global and Local Moves in Tabu Search: A Real-Life Mail Collecting Application. In: ____. **Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization**. Boston, MA: Springer US, 1999. p. 155–174. ISBN 978-1-4615-5775-3. Citation on page 28.

MITROVIĆ-MINIĆ, S.; KRISHNAMURTI, R.; LAPORTE, G. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. **Transportation Research Part B: Methodological**, v. 38, n. 8, p. 669–685, 2004. ISSN 0191-2615. Citation on page 97.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, v. 24, n. 11, p. 1097–1100, nov 1997. ISSN 03050548. Citation on page 49.

MOSCATO, P. **On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms**. Pasadena, CA, 1989. Citation on page 28.

NAGATA, Y.; BRÄYSY, O.; DULLAERT, W. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. **Computers & Operations Research**, v. 37, n. 4, p. 724–737, apr 2010. ISSN 03050548. Citations on pages 28 and 29.

NALEPA, J.; BLOCHO, M. Co-operation in the Parallel Memetic Algorithm. **International Journal of Parallel Programming**, v. 43, n. 5, p. 812–839, 2015. ISSN 1573-7640. Citations on pages 28 and 29.

OLIVEIRA, H. C. de; VASCONCELOS, G. C. A hybrid search method for the vehicle routing problem with time windows. **Annals of Operations Research**, v. 180, n. 1, p. 125–144, nov 2010. ISSN 1572-9338. Citations on pages 20, 28, 29, 30, 55, 56, and 57.

OMBUKI, B.; ROSS, B. J.; HANSHAR, F. Multi-objective genetic algorithms for vehicle routing problem with time windows. **Applied Intelligence**, v. 24, n. 1, p. 17–30, 2006. Citations on pages 28 and 43.

OSVALD, A.; STIRN, L. Z. A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food. **Journal of Food Engineering**, v. 85, n. 2, p. 285–295, 2008. ISSN 0260-8774. Citation on page 28.

PECIN, D.; CONTARDO, C.; DESAULNIERS, G.; UCHOA, E. New Enhancements for the Exact Solution of the Vehicle Routing Problem with Time Windows. **INFORMS Journal on Computing**, v. 29, n. 3, p. 489–502, 2017. Citation on page 28.

PIERRE, D. M.; ZAKARIA, N. Stochastic partially optimized cyclic shift crossover for multi-objective genetic algorithms for the vehicle routing problem with time-windows. **Applied Soft Computing**, v. 52, p. 863–876, 2017. Citation on page 43.

PILLAC, V.; GENDREAU, M.; GUÉRET, C.; MEDAGLIA, A. L. A review of dynamic vehicle routing problems. **European Journal of Operational Research**, v. 225, n. 1, p. 1–11, 2013. ISSN 0377-2217. Citations on pages 27, 31, 32, 33, and 34.

PSARAFTIS, H. N. Dynamic vehicle routing problems. In: GOLDEN, B.; ASSAD, A. (Ed.). **Vehicle routing: methods and studies**. Amsterdam, the Netherlands: North-Holland, 1988. p. 223–248. Citation on page 31.

____. Dynamic vehicle routing: Status and prospects. **Annals of Operations Research**, v. 61, n. 1, p. 143–164, 1995. ISSN 1572-9338. Citation on page 33.

PSARAFTIS, H. N.; WEN, M.; KONTOVAS, C. A. Dynamic vehicle routing problems: Three decades and counting. **Networks**, v. 67, n. 1, p. 3–31, 2016. Citations on pages 23, 27, 31, and 33.

PULLEN, H. G. M.; WEBB, M. H. J. A computer application to a transport scheduling problem. **The Computer Journal**, v. 10, n. 1, p. 10–13, 1967. Citation on page 27.

RITZINGER, U.; PUCHINGER, J.; HARTL, R. F. A survey on dynamic and stochastic vehicle routing problems. **International Journal of Production Research**, Taylor & Francis, v. 54, n. 1, p. 215–231, 2016. Citation on page 27.

ROCHAT, Y.; TAILLARD, É. Probabilistic diversification and intensification in local search for vehicle routing. **Journal of Heuristics**, v. 1, n. 1, p. 147–167, 1995. Citations on pages 20, 56, and 57.

SAVELSBERGH, M.; WOENSEL, T. V. 50th Anniversary Invited Article—City Logistics: Challenges and Opportunities. **Transportation Science**, v. 50, n. 2, p. 579–590, 2016. Citations on pages 24 and 33.

SKUPIN, A. A Cartographic Approach to Visualizing Conference Abstracts. **IEEE Comput. Graph. Appl.**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 22, n. 1, p. 50–58, jan 2002. ISSN 0272-1716. Citations on pages 25 and 66.

SOLOMON, M. M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. **Operations Research (INFORMS)**, v. 35, n. 2, p. 254–265, 1987. Citations on pages 27, 28, 29, 35, 36, 37, 39, 51, 62, 68, 84, and 111.

TAN, K. C.; LEE, T. H.; OU, K.; LEE, L. H. A messy genetic algorithm for the vehicle routing problem with time window constraints. In: **Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)**. Seoul, Korea: IEEE Press, 2001. v. 1, p. 679–686 vol. 1. ISBN VO - 1. Citations on pages 20, 56, and 57.

URSANI, Z.; ESSAM, D.; CORNFORTH, D.; STOCKER, R. Localized genetic algorithm for vehicle routing problem with time windows. **Applied Soft Computing**, v. 11, n. 8, p. 5375–5390, dec 2011. ISSN 15684946. Citations on pages 20, 28, 29, 30, 55, and 56.

VIDAL, T.; CRAINIC, T. G.; GENDREAU, M.; PRINS, C. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. **Computers & Operations Research**, v. 40, n. 1, p. 475–489, 2013. ISSN 0305-0548. Citations on pages 28, 29, and 31.

____. Time-window relaxations in vehicle routing heuristics. **Journal of Heuristics**, v. 21, n. 3, p. 329–358, 2015. ISSN 1572-9397. Citations on pages 20, 27, 28, 29, 30, 55, 56, and 57.

WILSON, N.; COLVIN, N. J. **Computer control of the Rochester dial-a-ride system**. Cambridge, Massachusetts, 1977. Citation on page 27.

WOOLDRIDGE, M. **An introduction to multiagent systems**. 2nd. ed. Chichester, UK: John Wiley & Sons, 2009. ISBN 978-0470519462. Citation on page 33.

XU, R.; WUNSCH, D. Hierarchical Clustering. In: **Clustering**. Hoboken, NJ: John Wiley & Sons, Inc., 2008. chap. 3, p. 31–62. ISBN 978-0-470-27680-8. Citations on pages 42 and 66.

ZOU, H.; DESSOUKY, M. M. A look-ahead partial routing framework for the stochastic and dynamic vehicle routing problem. **Journal on Vehicle Routing Algorithms**, feb 2018. ISSN 2367-3605. Citation on page 36.

APPENDIX

A

DVRPTWSC BENCHMARK INSTANCES

The DVRPTWSC instances were generated from the RC instances of Solomon (1987). The RC instances were chosen because they present a combination of clustered and randomly located customers. The instances are classified into classes RC1 and RC2; each class contains eight instances. In instances of class RC1, the planning horizon lasts 240 time units, the customers have narrow time windows and the vehicle capacity is equal to 200. In instances of class RC2, the planning horizon lasts 960 time units, the customers have wide time windows and the vehicle capacity is equal to 1000. In all instances, there are 100 customers without counting the depot, and the fleet is composed of 25 vehicles.

A DVRPTWSC instance contains two sets of customers. The first set contains the static customers, who made their requests before the start of the planning horizon. The second set contains the dynamic customers, who may or may not make their requests over the planning horizon. For each RC instance, three different DVRPTWSC instances were generated to represent different degrees of dynamism: low, medium, and high. The generated benchmark set is divided into classes RC1-L, RC1-M, RC1-H, RC2-L, RC2-M, and RC2-H. Instances of classes RC1-L and RC2-L represent situations of low dynamism; instances of classes RC1-M and RC2-M represent situations of medium dynamism; and instances of classes RC1-H and RC2-H represent situations of high dynamism. Each class contains 8 instances, for a total of 48 instances.

For a given DVRPTWSC instance, let n_s and n_d denote, respectively, the number of requests made by static and dynamic customers; $n_t = n_s + n_d$ is the total number of requests. In instances of classes RC-L, $n_s = 75$; in instances of classes RC-M, $n_s = 50$; and in instances of classes RC-H, $n_s = 25$. The static customers were randomly selected from the set of customers of the original Solomon instance with a bias in favor of the customers whose time windows start early. With the remaining customers, a binomial experiment was performed to choose the dynamic customers that make requests. Letting p be the probability that a dynamic customer makes a request, $n_d \sim \text{Binomial}(100 - n_s, \text{p})$ is the number of customers who make a request in the experiment. For instances of classes RC-L, $\text{p} = \frac{3}{5}$; for instances of classes RC-M, $\text{p} = \frac{4}{5}$; and

for instances of classes RC-H, $p = \frac{13}{15}$. With these probabilities, the total expected number of requests is equal to 90 for all instances:

Instances of classes RC-L:
$$\begin{aligned} E[n_t] &= E[n_s] + E[n_d] \\ &= E[75] + 25 \cdot \tfrac{3}{5} \\ &= 75 + 15 \\ &= 90 \end{aligned}$$

Instances of classes RC-M:
$$\begin{aligned} E[n_t] &= E[n_s] + E[n_d] \\ &= E[50] + 50 \cdot \tfrac{4}{5} \\ &= 50 + 40 \\ &= 90 \end{aligned}$$

Instances of classes RC-H:
$$\begin{aligned} E[n_t] &= E[n_s] + E[n_d] \\ &= E[25] + 75 \cdot \tfrac{13}{15} \\ &= 25 + 65 \\ &= 90 \end{aligned}$$

□

The requests made by dynamic customers are associated with an arrival time. For a dynamic customer j who makes a request, the arrival time of the request is given by $R_j \sim U(e_0, e_j - \mathfrak{t})$, where $\mathfrak{t}$ is a very short amount of time. To avoid generating requests that possibly cannot be feasibly served, R_j was generated in such a way that $R_j + t_{0,j} \leq l_j$ (where $t_{0,j}$ is the direct travel time between the depot and customer j). For dynamic customers with $e_j = e_0$, the start of their time windows was slightly delayed: $e_j = e_0 + \mathfrak{t}$, and $R_j = e_0$.

The likely degree of dynamism $\bar{\delta}$ of an instance is given by the ratio of the expected number of dynamic requests to the total expected number of requests:

$$\bar{\delta} = \frac{E[n_d]}{E[n_t]} \tag{A.1}$$

For instances of classes RC-L, $\bar{\delta} = 0.16$; for instances of classes RC-M, $\bar{\delta} = 0.44$; and for instances of classes RC-H, $\bar{\delta} = 0.72$.


ICMC
USP
SÃO CARLOS