Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

EXACT AND HEURISTIC ALGORITHMS FOR THE TRAFFIC COUNTING LOCATION PROBLEM, THE PERIODIC ROUTING PROBLEM OF OFFSHORE SUPPLY VESSELS, AND THE VEHICLE ROUTING PROBLEM

Bruno Salezze Vieira

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Transportes, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Transportes.

Orientadores: Glaydston Mattos Ribeiro
Laura Silvia Bahiense da Silva Leite

Rio de Janeiro
Agosto de 2021

EXACT AND HEURISTIC ALGORITHMS FOR THE TRAFFIC COUNTING LOCATION PROBLEM, THE PERIODIC ROUTING PROBLEM OF OFFSHORE SUPPLY VESSELS, AND THE VEHICLE ROUTING PROBLEM

Bruno Salezze Vieira

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE TRANSPORTES.

Orientadores: Glaydston Mattos Ribeiro
Laura Silvia Bahiense da Silva Leite

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2021

Vieira, Bruno Salezze

Exact and heuristic algorithms for the traffic counting location problem, the periodic routing problem of offshore supply vessels, and the vehicle routing problem/Bruno Salezze Vieira. – Rio de Janeiro: UFRJ/COPPE, 2021.

XVIII, 104 p.: il.; $29,7$cm.

Orientadores: Glaydston Mattos Ribeiro

Laura Silvia Bahiense da Silva Leite

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Transportes, 2021.

Referências Bibliográficas: p. 87 – 98.

1. Vehicle routing problem. 2. Heterogeneous fleet. 3. Platform supply vessel routing. 4. Multiple Trips. 5. Traffic counting location. 6. Set covering. 7. Hybrid heuristic. 8. Adaptive large neighborhood search. 9. Unified hybrid genetic search. I. Ribeiro, Glaydston Mattos *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Transportes. III. Título.

Dedico este trabalho aos meus pais, Afonso e Auxiliadora, por todo o apoio, confiança e dedicação, fundamentais para tornar realidade mais essa conquista.

Agradecimentos

Inicio meus agradecimentos por Deus, já que Ele colocou pessoas tão especiais a meu lado, sem as quais certamente não teria dado conta.

A meus pais, Afonso e Auxiliadora, todo meu agradecimento. Sempre me apoiarem e me incentivarem a ter uma educação de qualidade, e sempre acreditando em minha capacidade. Obrigado.

Aos meus orientadores, professor Glaydston Mattos Ribeiro e professora Laura Silvia Bahiense da Silva Leite, sempre próximos criticando e dando sugestões com muita paciência de como prosseguir. Também ao professor Gilbert Laporte pela sua orientação durante ao meu período de pesquina no Canadá.

À equipe do Projeto PNCT, professor Rômulo Dante Orrico Filho, Saul Quadros, Cristiane da Penha Bernardo e Gerusa Ravache, por toda amizade, companheirismo e ensinamentos. Participar desse projeto me proporcionou enorme aprendizado, pessoal e profissional, que levarei por toda a vida.

Aos colegas de mestrado e doutorado, especialmente, Tulio Silveira, Marcus Vinícius Oliveira Câmara, Lygia Bronneberg, Vanessa Guimarães, Bruno Guida, Pedro Basilio, Igor Godeiro, Thayse Ferrari, Ewerton Torres, Victor Albino, Inaê Lucato, Breno Garcia, Pedro Geaquinto e Luiz Saldanha.

A Jane Correa de Souza e Maria Helena Santos Oliveira por todo apoio na secretaria do PET. Sua disposição em ajudar contribuíram para minha formação acadêmica.

Ao DNIT e ao COTER pelas informações e sugestões que contribuíram para o aperfeiçoamento da pesquisa.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, pelo auxílio concedido.

Aos professores Geraldo Regis Mauri, Thibault Vidal, Pedro González e Rômulo Orrico Filho por aceitarem participar da banca e por toda contribuição para aprimoramento da pesquisa.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ALGORITMOS EXATOS E HEURÍSTICOS PARA O PROBLEMA DE LOCALIZAÇÃO DE CONTAGEM DE TRÁFEGO, O PROBLEMA DE ROTEAMENTO PERIÓDICO DE NAVIOS DE ABASTECIMENTO OFFSHORE E O PROBLEMA DE ROTEAMENTO DE VEÍCULOS

Bruno Salezze Vieira

Agosto/2021

Orientadores: Glaydston Mattos Ribeiro
Laura Silvia Bahiense da Silva Leite

Programa: Engenharia de Transportes

Este trabalho aborda três problemas de transporte: o Problema de localização de contadores de tráfego (em inglês TCLP), o Problema de planejamento de navios de suprimentos periódico (em inglês PSVPP) e o Problema de roteamento de veículos dependente de local periódico com multiplas viagens e multiplos depósitos (em inglês HSDMDMTPVRP). No TCLP, propomos um algoritmo híbrido progressivo baseado em abordagens exatas, heurísticas e híbridas em uma estrutura de cobertura de conjuntos para resolvê-lo. Instâncias reais obtidas nos estados brasileiros são usadas e os resultados computacionais mostram que o TCLP pode ser resolvido de forma mais eficiente do que as abordagens anteriores. A variante PSVPP abordada neste trabalho modela um problema da vida real, para o qual apresentamos um algoritmo branch-and-cut e uma heurística *Adaptive Large Neighborhood Search* (ALNS). Esses algoritmos foram testados em instâncias com até 79 plataformas, obtendo resultados melhores do que os melhores disponíveis. No HSDMDMTPVRP, estudamos uma variante real do problema de roteamento de veículos e apresentamos duas metaheurísticas. A primeira é uma adaptação do *Unified Hybrid Genetic Search* (UHGS). A segunda é uma nova metaheurística chamada *Adaptive Variable Neighborhood Race* (AVNR) que combina busca de vizinhança variável e mecanismos adaptativos integrados com uma população decrescente gerenciada com um mecanismo de diversidade. Nossos experimentos computacionais usaram 398 instâncias da literatura e 16 novas. As metaheurísticas propostas encontraram 143 novas soluções e 205 melhores até então conhecidas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

EXACT AND HEURISTIC ALGORITHMS FOR THE TRAFFIC COUNTING LOCATION PROBLEM, THE PERIODIC ROUTING PROBLEM OF OFFSHORE SUPPLY VESSELS, AND THE VEHICLE ROUTING PROBLEM

Bruno Salezze Vieira

August/2021

Advisors: Glaydston Mattos Ribeiro
Laura Silvia Bahiense da Silva Leite

Department: Transportation Engineering

This work tackles three transportation problems: the Traffic Counting Location Problem (TCLP), the Periodic Supply Vessel Planning Problem (PSVPP) and the Heterogeneous Site-Dependent Multi-depot Multi-trip Periodic Vehicle Routing Problem (HSDMDMTPVRP). In the TCLP, we propose a progressive hybrid algorithm based on exact, heuristic and hybrid approaches embedded on a set covering framework to solve it. Real-world instances obtained from the Brazilian states are used in the computational experiments and the results show that the TCLP can be solved more efficiently than previous approaches. The PSVPP variant tackled in this work models a real-life problem faced by a Brazilian oil and gas exploration and production (E&P) operator. We present a branch-and-cut algorithm and an Adaptive Large Neighborhood Search (ALNS) heuristic to solve this PSVPP. These algorithms were tested on instances with up to 79 offshore units, providing better results than the best available. In the HSDMDMTPVRP, we study a real Vehicle Routing Problem (VRP) variant and present two metaheuristcs. The first one is an adaptation of the Unified Hybrid Genetic Search (UHGS). The second one is a new metaheuristic named Adaptive Variable Neighborhood Race (AVNR) which combines variable neighborhood search and adaptive mechanisms integrated with a shrinking population managed with a diversity mechanism. Both approaches are used to solve the HSDMDMTPVRP and its subproblems. Our computational experiments used 398 literature instances and presents 16 new ones. The proposed metaheuristics found 143 new best-known solutions and 205 of the best-known ones.

Contents

List of Figures

List of Tables

List of Symbols

A	Set of Edges, p. 10
$C(s)$	Penalized cost of solution s, p. 40
D	Set of depots, p. 54
D^{max}	Maximum distance between any two clients, p. 41
G	Graph, p. 10
K	Set of vehicle types, p. 54
K_b	Queue Capacity of berth b, p. 26
L_b^f	Vessel startup time of berth b, p. 26
L_b^q	Vessel loading rate of berth b, p. 26
L^-	List of removed clients, p. 41
M_k^{d2t}	Average distance to time coefficient of vehicle type k, p. 54
M_k^d	Maximum operation time of vehicle type k, p. 54
M_k^l	Maximum capacity of vehicle type k, p. 54
M_k^r	Maximum amount of vehicles per period of vehicle type k, p. 54
M_k^t	Maximum amount of trips per period of vehicle type k, p. 54
N	Set of Nodes, p. 10
N^B	Set of berths, p. 26
N^{OI}	Set of offshore units, p. 26
P	Set of all O-D paths, p. 10
P_i	Set of delivery patters of i, p. 26

P_{repair}	Probability of repair, p. 65
Q	Maximum demand between all clients, p. 41
S_v	Travel speed of vessel type v, p. 26
T	Set of periods, p. 26
U	Set of clients, p. 54
U_v	Capacity of vessel v, p. 26
V	Set of vessels, p. 26
W	Set of O-D pairs, p. 10
$\Delta(s_1, s_2)$	Distance between solutions s_1 and s_2, p. 60
$\Omega(s)$	Penalty cost of solution s, p. 56
$\Phi(u, s)$	Best insertion position for client u at solution a, p. 42
ϵ^{close}	Proportion of the closest solutions, p. 60
ϵ^{elite}	Proportion of elite solutions, p. 60
$\eta(s, t, i)$	Neighborhood of client i, at solution s and period t, p. 58
μ^{base}	Base population size, p. 60
c_v^f	Fixed Cost of v, p. 26
c_v^f	Variable Cost of v, p. 26
$e(s)$	Sorted multiset of edges from solution s, p. 54
f_i	Number of services required by i, p. 26
$gt(s_{tkd}^p)$	Giant tour of partial solution s_{tkd}^p, p. 54
n^-	Number of removed clients, p. 40
n^1	Lower rate of removed clients, p. 40
n^2	Higher rate of removed clients, p. 40, 41
n_{ts}	Total number of services, p. 54
$p(s, i)$	Delivery pattern of client p in solution s, p. 54

$pCost(s^p, d, k)$	Penalised cost of partial solution s^p, for depot d and vehicle type k, p. 56
q_i	Load Demand of i, p. 26
s	Solution, p. 30
s^p	Partial Solution, p. 54
s^{red}	Solution for the PSVPP-BA reduced model, p. 32
t_i	Service time required by i, p. 26

List of Abbreviations

ALNS	Adaptive Large Neighborhood Search, p. 5
AVNR	Adaptive Variable Neighborhood Race, p. 6
CVRP	Capacitated Vehicle Routing Problem, p. 53
GA	Genetic Algorithm, p. 9
HSDMDMTPVRP	Heterogeneous Site-Dependent Multi-depot Multi-trip Periodic Vehicle Routing Problem, p. 1
HVRP	Heterogeneous Vehicle Routing Problem, p. 53
LNS	Large Neighborhood Search, p. 25
MDVRP	Multi-depot Vehicle Routing Problem, p. 53
MTVRP	Multi-trip Vehicle Routing Problem, p. 53
PSVPP-BA	Periodic Supply Vessel Planning Problem with Berth Allocations, p. 4
PSVPP	Periodic Supply Vessel Planning Problem, p. 1
PVRP	Periodic Vehicle Routing Problem, p. 24, 53
SCF	Set Covering Framework, p. 12
SCP	Set Covering Problem, p. 12
SDMTPVRP	Site-dependent Multi-trip Periodic Vehicle Routing Problem, p. 53
SDVRP	Site-dependent Vehicle Routing Problem, p. 53
TCLP	Traffic Counting Location Problem, p. 1
UHGS	Unified Hybrid Genetic Search, p. 6

USCP	Unicost Set Covering Problem, p. 12
VRP	Vehicle Routing Problem, p. 5

Chapter 1

Introduction

Optimization problems appears in several fields. In the transport engineering area, we can mention, for example, traffic signalling management, bus scheduling, traffic counting location and vehicle routing. Many optimization techniques have been developed to solve these problems, but there is room to improve the results found so far given the difficulty of these problems.

This thesis solves three hard problems: the Traffic Counting Location Problem (TCLP), the Periodic Supply Vessel Planning Problem (PSVPP) and the Heterogeneous Site-Dependent Multi-depot Multi-trip Periodic Vehicle Routing Problem (HSDMDMTPVRP). Each one is presented in the next sections which also show our contributions.

1.1 Traffic counting location problem

The strong Brazilian economic growth registered in recent decades has promoted an increase in the demand for products and services all over the country. This has direct influence on the number of passengers and the quantity of cargo transported on national highways, since the road is the most used mode of transportation in Brazil.

An efficient road transportation system is essential for the development of a country, since it is necessary to ensure the circulation of people and goods throughout the territory safely and efficiently. In Brazil, investments on road infrastructure are even more relevant, since this mode is responsible for 96% of passenger and 62% of freight transportation [3].

According to GOMES [4], these investments are very important to provide security and reduction of travel times and operational costs. In this sense, studies regarding the traffic, flows, and their temporal variation in road networks are essential.

Several counting methods can be used to monitor traffic flows on highways, which tend to be costly mainly because they usually require a skilled workforce. Traffic sensors, that count the number of vehicles passing by a given point over a predefined period of time, tend to be more affordable options, making them often used in traffic monitoring [5]. These devices provide data to support traffic management applications, such as traffic control, semaphore, ramp control, incident detection and vehicle classification [6, 7].

The Traffic Counting Location Problem (TCLP) aims to locate optimally traffic sensors (stations) with a given objective and sets of constraints. As reported by GENTILI and MIRCHANDANI [8], these sensors are usually grouped into four categories - traffic counting, route, image, and vehicle identification, and the most commonly used is the first one. If a traffic sensor is located on a road, it measures the volume of vehicles on it, on the other hand, if it is located on a network node, it measures the flows of all incoming and outgoing roads connected to this node.

A good traffic monitoring system must receive information from the entire network of interest. In this context, TCLP arises, aiming to find the best number and location of counting stations to cover a road network in order to obtain its traffic flows. Finding the best location for the sensors is important to reduce costs of deployment, maintenance and operation of the traffic stations.

The TCLP considers a network which has a set of roads and a set of nodes. The set of nodes is divided into two sets: the origin and destination (O-D) nodes and the network intersections. When travelling between cities, for example, trips are performed from origins to destinations throughout the roads, passing by the network intersections. In this case, there are frequently different paths connecting an O-D pair which can be used by the drivers. So, the TCLP has to define the set of roads to receive counting stations to intercept (or cover) all paths between the O-D pairs.

We propose a progressive hybrid algorithm based on exact, heuristic and hybrid approaches embedded on a set covering framework to solve the TCLP. We have used 26 real-world instances obtained from the Brazilian georeferenced state road networks, containing information about all O-D pairs (Brazilian municipalities). The entire network was provided by the National Department of Transport Infrastructure (DNIT), that is responsible for implementing the National Traffic Counting Plan (NTCP). The NTCP aims to obtain the traffic flows for the actual network having more than 50,000 km and connecting approximately 5,600 municipalities. Our approach solved 84% of the instances optimally, and for three ones it found best-known solutions.

1.2 Fleet composition and periodic routing problem of offshore supply vessels with berth allocation decisions

The periodic supply vessel planning problem (PSVPP) arising in the upstream offshore petroleum logistics chain of oil and gas exploration and production. The PSVPP variant tackled in this work models a real-life problem faced by a Brazilian oil and gas exploration and production (E&P) operator.

As described in GRIBKOVSKAIA *et al.* [9] and in AAS *et al.* [10], the PSVPP is a key problem in this context since it ensures the regular replenishment of equipment and other supplies to offshore units, as well as the return of waste and depleted equipment back to the onshore base. This is usually a high-cost operation, essential to avoid interruptions in the continuous production of oil.

We consider (*i*) the simultaneous determination of the heterogeneous fleet composition, the vessel schedules and the berth allocations; (*ii*) continuous and flexible departures from the base; (*iii*) historical data to model the berth allocations and departures; and (*iv*) the solution for the largest real-world offshore instances ever presented, with up to 79 installations.

The periodic supply vessel planning problem with berth allocations is defined according to the description provided by a Brazilian oil and gas operator, and unifies the characteristics described in CRUZ *et al.* [2], KISIALIOU *et al.* [11], and HALVORSEN-WEARE and FAGERHOLT [12]:

- *Non-stop operations*: No opening hours are considered for either the onshore base or the offshore unit. The onshore base, the offshore units and the vessels operate continuously, seven days a week, without any interruption, and no time windows are considered at the offshore units.

- *Periodic service*: Each offshore unit has its frequency of service, commonly called "frequency of visits", depending mainly on its own demand, which is limited by the required deck area. The production units have a more stable demand, implying a lower frequency of service, typically once or twice a week. In the case of drilling rigs, the demand is more unpredictable, implying a higher frequency of service, typically three or more times per week.

- *Spread departures from the base*: The frequency of attendance to each maritime unit is linked to the vessel departures. In fact, the departures from the onshore base to each offshore unit should be spread as evenly as possible over the planning horizon in order to avoid delays in meeting the delivery requests from the installations. This characteristic and "Periodic service" are satisfied through the service frequency patterns (mentioned before), creating what we call *delivery patterns.*

- *Heterogeneous and common resource fleet*: Each type of vessel has its own deck capacity, a fixed cost over the planning horizon, and a variable cost per travelled distance. In addition, the vessels are not associated to routes, but are considered a common resource to be used whenever requested and with the objective of reducing costs.

- *Route duration*: The vessel capacity limits the maximum number of offshore units that can be visited on the same route, and longer routes are more subject to delays and adverse weather conditions. Hence, the maximum number of installations serviced in the same route cannot exceed eight. As a result, the time required for the routes vary between one and four days, approximately.

- *Two daily berth departures*: The loading times at the onshore base are calculated by considering a fixed setup time (vessel berthing time) added to the variable loading time, which is proportional to the voyage's total demand. The maximum number of departures from each berth on each day is limited to two, based on the actual loading rates and on the historical average amount of cargo transported. Therefore, there is one physical berth position that has to be shared for a maximum of two unloading/loading operations, and each departure from each berth's queue position is called a *departing position.*

- *Berth departures' tolerance*: In order to bring flexibility to the berth planning process, a vessel can begin the loading process on a day and finish it on the next day, given a tolerance of up to 12 hours to leave the onshore base. The inclusion of this flexibility in hours on the departure day in the model is important to deal with cases where the port is overloaded.

- *Static routing*: All parameters are known in advance, so no stochastic data are used and the weather impact on the voyage duration is not considered.

In short, the PSVPP tackled here consists of solving a periodic vehicle routing problem by simultaneously determining the optimal fleet size and mix of heterogeneous offshore supply vessels, their weekly routes and schedules for servicing the offshore oil and gas installations, as well as the berth allocations (BA) at the supply base. Hence we will call this problem the PSVPP-BA.

We present a branch-and-cut algorithm and an adaptive large neighborhood search (ALNS) heuristic with multiple starts and spaced local searches to solve this PSVPP. We also use a replicable one-week planning horizon both for the offshore units and the vessels, and solve the largest available benchmark instances without dividing them into clusters. These algorithms were tested on instances with up to 79 offshore units, providing better results than the best available.

1.3 Heterogeneous Site Dependent Multi depot Multi trip Periodic Vehicle Routing Problem

Vehicle routing has great importance in the literature and has been widely studied due to its relevance to industry and economy. Such study began with the Vehicle Routing Problem (VRP) proposed by DANTZIG and RAMSER [13] where vehicles with the same characteristics leave a distribution center (or warehouse) and deliver or pickup goods at specific locations (which may be customers or suppliers) at the lowest possible cost. The cost in this case is basically associated with the distance traveled.

Although this problem has been studied for almost six decades [14], real applications remain a challenge. They feature a variety of operational attributes that complicate the problem and may have a significant impact on the solution process. These additional considerations may, for example, affect vehicles costs and capacities, customers constraints and multiple depot operations.

In this work, we study a real VRP variant for a major automotive company in Brazil that makes around 300 weekly services. This version of the problem addresses the following considerations:

- *Clients:* They are geographically dispersed with predetermined positions to define the routes;
- *Periodic non-symmetric demands*: Each client can have individual demands, with individual frequencies as well;
- *Heterogeneous fleet*: There are different types of vehicles, with different capacities and costs that can be used to perform the services (pickups or deliveries). In this problem, larger vehicles lead to larger usage costs (KOÇ *et al.* [15]). The problem faced has distribution centers, sizing its fleets to service the demands;
- *Working day of the field teams*: The service must respect time windows for customers and distribution center, and there is a maximum number of hours for a driver work daily;
- *Docking constraints*: Each client can be served by certain types of vehicles;
- *Multiple depots*: Each vehicle route starts and ends at the same depot;

- *Multiple trips of each vehicle per day*: A vehicle may leave the distribution center, perform all pickups, and return to the starting point in a period of time shorter than its useful working time. Consequently this vehicle is able to perform a new trip; and
- *Periodic scheduling*: The scheduling of the routes should be done for a certain planning horizon which is about one week.

We adapted and implemented the current state-of-the-art algorithm for most of the VRP sub-problems presented in the HSDMDMTPVRP called Unified Hybrid Genetic Search (UHGS) [16] with an advanced diversity control, feasibility control and a restart mechanism.

In addition, we also present a new metaheuristic called Adaptive Variable Neighborhood Race (AVNR) which combines successful strategies in the literature: variable neighborhood search and adaptive mechanisms integrated with a shrinking population managed with a diversity mechanism. This metaheuristic was tested with well-known instances of the literature for all VRP sub-problems presented in the HSDMDMTPVRP and the results were compared against the best-know methods developed to solve each one.

Both approaches were tested in the same computer with the same execution time, with shared code and different set of parameters. The computational experiments considered 398 instances available in the literature. Our metaheuristic found 143 new best-known solutions and 205 of the best-known ones. For the remaining instances, it found results very close to best ones known.

1.4 Thesis structure

The following chapters are dedicated to each one of the transportation related optimization problems: TCLP (Chapter 2), PSVPP-BA (Chapter 3) and the HSDMDMTPVRP (Chapter 4).

More specifically, in Chapter 2, Section 2.1 surveys the related TCLP literature and Section 2.2 introduces the mathematical formulation used in this to represent the TCLP. The solution approach is presented in Section 2.3. Instances and computational results are presented in Section 2.4, which also shows a comparison with the literature. At last, Section 2.5 gives the final remarks of the chapter.

In Chapter 3, Section 3.1 surveys the related PSVPP-BA literature and Section 3.2 presents the mathematical modelling. Section 3.3 describes the branch-and-cut algorithm, Section 3.4 presents the ALNS heuristic and Section 3.5 details the computational experiments. Section 3.6 presents the final remarks of the chapter.

In Chapter 4, Section 4.1 overviews the related VRP literature and Section 4.2 presents the mathematical problem description for the HSDMDMTPVRP. The shared search mechanisms are described and than the proposed metaheuristics are detailed in Section 4.3, followed by computational experiments in Section 4.4. And Section 4.5 give the final remarks of the chapter.

Finally, Chapter 5 presents the conclusions, achievements and suggestions for future researches.

Chapter 2

A progressive hybrid algorithm set covering based for the traffic counting location problem

This chapter approaches the traffic counting location problem (TCLP). We propose a unicost set covering formulation based on the well-known formulations of YANG *et al.* [17]. A set covering based progressive hybrid algorithm is developed to test the performance of this model over a set of 26 real-world instances derived from the Brazilian roads. The obtained results are compared to the ones found by GONZÁLEZ *et al.* [1].

The proposed algorithm presented in this chapter was published in VIEIRA *et al.* [18].

2.1 Literature review

The Traffic Counting Location Problem (TCLP) is one of the Flow Capturing problems introduced by HODGSON [19], including some facility location problems, surveillance systems, and data network observation. There are three types of flow capturing problems over road networks: (*i*) those in which the facility to be located is desired by the driver, like gas stations or convenience stores, and the driver may take a longer path in order to achieve it, as can be seen in KIM and KUBY [20]; (*ii*) those in which the item to be located is avoided by the driver, like speed radars or toll booths, as recently investigated by ARSLAN *et al.* [21]; and (*iii*) those in which the driver does not care about the item to be located, like billboards or traffic counting sensors for traffic monitoring. The TCLP fits exactly in this last category of flow capturing problems, in which the driver does not care about the sensors to be located.

Traffic monitoring studies date back to FISK [22], and they are important for several transportation studies, for example, to find critical roads, determine average annual daily traffic, forecast future travel demand, and manage transport and control. It is also useful to estimate Origin-Destination (O-D) trip tables (or O-D trip matrices), which can condense significant information.

According to MOHAMAD *et al.* [23], traffic monitoring involves the collection of many types of data, such as vehicle speeds, vehicle weights, traffic volume, and traffic composition. As claimed by ZHONG *et al.* [24], highway agencies have developed traffic monitoring programs to obtain traffic data to be used in planning, design, control, operation, and management of traffic and highway facilities.

O-D matrices specify the number of trips between each pair of origin and destination nodes on a network. These matrices are important for various analyses in transportation planning and operations [8, 25]. The quality of estimated O-D matrices depends strongly on the quality of collected data from sensors, which is directly impacted by the number and the locations of traffic counting points [26].

The concept of Maximum Possible Relative Error (MPRE), based on a quadratic programming formulation, was introduced by YANG *et al.* [27] to investigate the reliability of estimated O-D trip matrices, resulting in a useful index for evaluating the estimation accuracy of these matrices.

A comprehensive research about traffic counting locations was conducted by YANG and ZHOU [28] to effectively estimate the O-D trip matrices from traffic counts. Based on the MPRE concept of YANG *et al.* [27], they derived four rules to locate traffic counting stations (O-D covering rule, maximal flow fraction rule, maximal flow-intercepting rule and link independence rule), and proposed integer linear programming models and heuristic algorithms to determine the counting links satisfying these rules, which were incorporated as constraints.

Screen-line based TCLP models were introduced by YANG *et al.* [29] to optimally select traffic counting stations in a road network. They were based on the O-D separation rule, without needing explicit references to existing O-D flows and turning proportions at each node, or behavioral assumptions of link/route choice proportions. They developed integer programming models and a genetic algorithm (GA) to solve two screen-line based TCLP: the location of a given number of counting stations to separate as many O-D pairs as possible, and the minimal number and locations of counting stations required to separate all O-D pairs.

A bi-objective model for locating traffic counting stations to estimate O-D matrices was proposed by CHOOTINAN *et al.* [30]. The two objectives explicitly consider the trade-off between the quality (maximization of the number of O-D being separated) and cost (minimization of the number of traffic counting stations used). They developed weighted-sum and distance-based genetic algorithms to generate an approximate set of non-dominated solutions. The distance-based GA performed better, and the numerical results indicated that, through this method, decision makers can examine the trade-off between the quality and the cost of the generated coverings, in order to make a proper selection without the need to repeatedly solve the maximal covering problem with different levels of resource.

In YANG *et al.* [17], the screen-line based TCLP models introduced by YANG *et al.* [29] were reformulated as integer linear programming problems. They developed a shortest path based column generation procedure to find optimal screen-line counting location solutions.

New strategies for selecting additional traffic counts, based on the screen-line TCLP models proposed by YANG *et al.* [17, 29], were introduced by CHEN *et al.* [31]. A GA embedded with a shortest path algorithm was developed in order to avoid the need to enumerate paths when solving the integer programs for large-scale network applications. They set up a real-world georeferenced experiment in order to visually observe the evolution of O-D estimation as the number of traffic counting locations increases, identifying various spatial properties of O-D trip tables estimated from traffic counts of different locations as results of their study.

Recently, GONZÁLEZ *et al.* [1] proposed a branch-and-cut algorithm and a clustering search (CS) heuristic to solve the TCLP. They also presented new real-world benchmark instances based on data from the Brazilian road network, which were used to compare the performance of their methods with the state-of-the-art GA algorithm developed by CHEN *et al.* [31]. By means of statistical significance analyses, they stated that their CS heuristic outperformed GA in all aspects.

The TCLP can be seen as a set covering problem where road segments with sensors cover a set of paths that connect O-D pairs. So, new exact, heuristic and hybrid techniques are proposed to solve the TCLP as presented in the next sections.

2.2 Mathematical formulation for the TCLP

Let $G = (N, A)$ be a road network, where A is the set of roads (bidirectional links) and $N = T \bigcup I$ is the set of nodes. The O-D nodes are represented by set T, while set I indicates all network intersections. The set $W = \{(i, j) | i \in T, j \in T, i \neq j\}$ represent the O-D pairs. For each $w = (i, j) \in W$, the set $P = \bigcup_{w \in W} P_w$ denotes all paths P_w between origin $i \in T$ and destination $j \in T$.

To better understand this notation, Figure 2.1 illustrates a small road network $G = (N, A)$ composed by $|A| = 13$ links, and $|N| = 9$ nodes, where $T = \{A, B, D\}$ represents O-D points, and the remaining $|I| = 6$ nodes denote intersection nodes.

The O-D pairs are represented in Figure 2.1 by $W = \{(A, B), (A, D), (B, D)\}$, and there are several paths connecting them. For example, paths $(1, 4, 6, 7)$ and

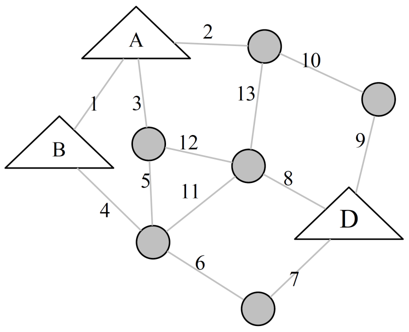


Figure 2.1: Hypothetical road network.

$(2, 10, 9)$ connect origin A to destination D, consequently they belong to set $P_{w=(A,D)}$.

The integer programming (IP) formulations introduced by YANG *et al.* [17] both decide the minimal number of traffic count stations and its locations on a road network G, so that they intercept all paths between O-D pairs. These are the most used formulations until nowadays, nonetheless they have a huge number of columns, and precisely for that reason, column generation approaches must be used. In order to avoid that, we decided to adopt a formulation based on the set covering problem.

Let $x_a \in \{0, 1\}$ be a binary variable indicating whether a traffic station should be located on road a ($x_a = 1$), or not ($x_a = 0$), and let $\delta^w_{pa} \in \{0, 1\}, w \in W, a \in A, p \in P_w$ be a binary parameter specifying if link a is on path p connecting O-D pair w ($\delta^w_{pa} = 1$), or not ($\delta^w_{pa} = 0$). Consequently, the mathematical formulation used in this work is presented below [17].

$$\text{Minimize } z = \sum_{a \in A} x_a \tag{2.1}$$

Subject to:

$$\sum_{a \in A} \delta^w_{pa} x_a \geq 1 \qquad \forall w \in W, p \in P_w \tag{2.2}$$

$$x_a \in \{0, 1\} \qquad \forall a \in A \tag{2.3}$$

The Objective Function (2.1) aims to minimize the number of traffic counting stations required to separate all O-D pairs in the network. Constraints (2.2) ensure that every O-D pair is separated by at least one counting station (selected link). Finally, Constraints (2.3) define the domain of the decision variables.

The mathematical formulation (2.1)-(2.3) can also represent the Unicost Set Covering Problem (USCP), since the covering cost is the same for each set. In this problem, the sets are composed by the paths between O-D pairs $w \in W$, and each link $a \in A$ is a node to be selected. As stated by YELBAY *et al.* [32], unicost problems are generally assumed to be more challenging in regard to their non-unicost counterparts. In fact, due to their intrinsic symmetry, the size of their restricted IP is used to be identical to the size of the original problem in dual approaches.

One notable characteristic of the TCLP formulation (2.1)-(2.3) is that it has a huge number of rows (paths), but a much smaller number of columns (links). This drove our approach to solve the problem, since the giant number of paths turns prohibitive to enumerate all of them efficiently in reasonable computational time [17, 29].

2.3 Set covering solution approach

This section details our approach to solve the TCLP. Differently from GONZÁLEZ *et al.* [1], that apply a branch-and-cut algorithm over Constraints (2.2) to separate the O-D pairs, we propose a framework, described in Algorithm 1, to solve the TCLP by solving an USCP multiple times.

Algorithm 1 successively applies two phases for solving the TCLP: a *Covering* phase (Line 4), to solve the USCP; and a *Cut* phase (Line 6), which tries to find paths between O-D pairs bypassing traffic count stations. The algorithm ends when these paths can no longer be found. After the *Covering* phase, a *Cleaning* procedure is applied to remove possible redundant links.

Algorithm 1: Set Covering Framework - *SCF*

```
Data: Network $G$, set of O-D pairs $W$, initial set of paths $P_1$, Covering
      method, Cleaning method, spread criterion SpreadCrit, time limit
      per cover $TL_1$ and time limit total $TL_2$
Result: A set $C$ of links to receive traffic counting stations (sensors).
1 $P_2 \leftarrow \emptyset$;
2 repeat
3     $P_1 \leftarrow P_1 \cup P_2$;
4     $C \leftarrow Covering(P_1, min(TL_1, TL_2 - Time))$;
5     $C \leftarrow Cleaning(C, P_1)$;
6     $P_2 \leftarrow Cut(G, C, SpreadCrit)$;
7 until $|P_2| = \emptyset$ or $Time \geq TL_2$;
8 if $|P_2| \neq \emptyset$ then $C \leftarrow A$;
```

It has as input: a network G, its set W of O-D pairs, the initial set of paths P_1, a set covering solution method $Covering$, a set covering cleaning method $Cleaning$, a spread criterion $SpreadCrit$ to be used in the *Cut* phase, a time limit per covering TL_1, and a total time limit TL_2. At last, P_2 represents the current set of new paths obtained by the *Cut* phase, and C is the current covering. At the end of the algorithm, either a feasible covering C is returned, or an empty set, meaning no cover was found. In Algorithm 1, *Time* is the CPU Time already used by the approach.

To illustrate how Algorithm 1 works, consider Figure 2.2 which is based on the hypothetical road network shown in Figure 2.1. Suppose that, in some iteration, we find the set P_1 of paths connecting the three O-D nodes A, B and D. The associated USCP is represented on the right-side of Figure 2.2. Note that when a path contains a link, there is an edge connecting them. Now, to solve this USCP, we have to find the minimum set of links to cover all paths. After solving this USCP, new paths can be added to P_1 to prevent sensors located in *Covering* phase from being bypassed by the uncovered paths, which is done by the *Cut* phase and explained latter at Section 2.3.2.

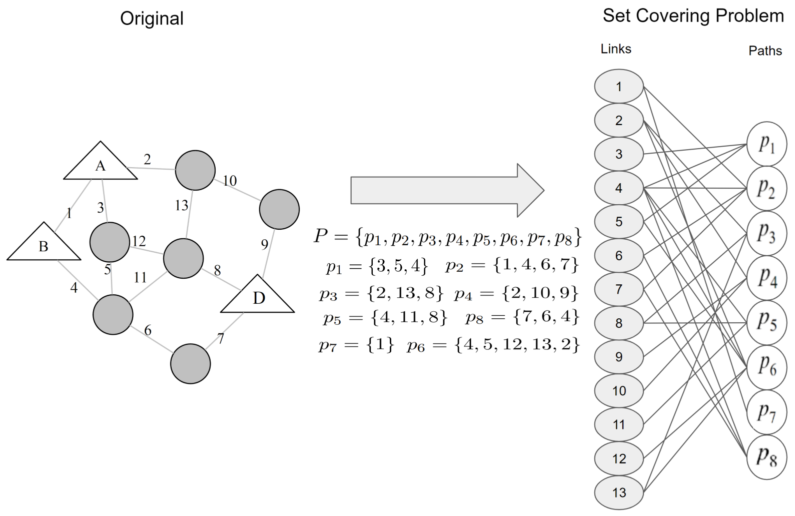

Original
Set Covering Problem
Links
Paths
$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$
$p_1 = \{3,5,4\}$ $p_2 = \{1,4,6,7\}$
$p_3 = \{2,13,8\}$ $p_4 = \{2,10,9\}$
$p_5 = \{4,11,8\}$ $p_8 = \{7,6,4\}$
$p_7 = \{1\}$ $p_6 = \{4,5,12,13,2\}$
A
B
D

Figure 2.2: Example of application of Algorithm 1.

The next sections detail the set covering solution approach that started to be presented by Algorithm 1 and that will end with Algorithm 5, which describes the progressive covering approach proposed in this chapter.

2.3.1 Covering phase

The *Covering* phase of Algorithm 1 can be solved exactly, heuristically or in a hybrid way, and each of them has its pros and cons. Since P_1 is a subset of all paths P, the covering algorithms to find P_1 have to be carefully chosen, since every time P_1 grows, a new USCP is generated and solved again.

Next we propose three approaches to solve the USCP: a greedy, an exact and a hybrid one.

Greedy covering heuristic

The greedy approach is the most widely used heuristic to solve the USCP [33], due to its basic assumption that the best elements (links for the TCLP) are the ones that cover the largest number of sets (paths) at once. In this sense, our greedy heuristic is presented in Algorithm 2. It has as input the set of current paths P_1, and in each loop, a link $l \in A$ connected to the largest number of paths ($l \leftarrow \text{argmax}_{a \in A} |P^*_a|$) is selected to receive a traffic counting station (lines 3 and 4), and then those paths connected to link l are removed from set P^* (line 5). The process is repeated until set P^* is empty and the greedy covering C is obtained.

Algorithm 2: Greedy Covering Heuristic

```
  Data: Set of Current Paths $P_1$
  Result: Cover $C$ for $P_1$
1 $C \leftarrow \emptyset$; $P^* \leftarrow P_1$ ;
2 repeat
3     $l \leftarrow \text{argmax}_{a \in A} |P^*_a|$;
4     $C \leftarrow C \cup \{l\}$; ;
5     $P^* \leftarrow P^* \setminus P^*_l$ ;
6 until $P^* = \emptyset$;
```

Figure 2.3 illustrates how this heuristic works for the current set of paths P_1 presented in Figure 2.2. As it can be seen, link 4 is the one covering the largest number of paths (five for this example), so it is selected, added to the solution, and the paths covered by it are removed from P^* in the first iteration. The second iteration selects link 2, and the third one selects link 1, in such a way that $C = \{1, 2, 4\}$ covers all paths. Therefore, for set P_1 shown in Figure 2.2, the Greedy covering heuristic places three traffic counting stations on links 1, 2 and 4. However, it can be seen in Figure 2.1 that there are still uncovered paths. For example, path $\{3, 12, 8\}$ connects the O-D pair (A, G) bypassing the sensors placed in C. So, it is important to add new paths to set P_1 in the *Cut* phase of Algorithm 1.

Cleaning procedure: Solutions returned by heuristic procedures usually contain redundant links, i.e., more links were selected than needed (CAPRARA *et al.* [33]). This happens because some links selected in the current iteration can make some previously selected links redundant. That is why we proposed a *Cleaning* procedure in Line 5 of Algorithm 1. The process of removing redundant links usually leads to solve another USCP. So, any USCP solution algorithm can be applied in order to find a tighter cover. Given a current cover C and the set of paths P_1 covered by C, the set of redundant links is defined as $R_C := \{i \in C \,|\, C \setminus \{i\} \text{ covers } P_1\}$. Thus, we propose the following cleaning algorithms:

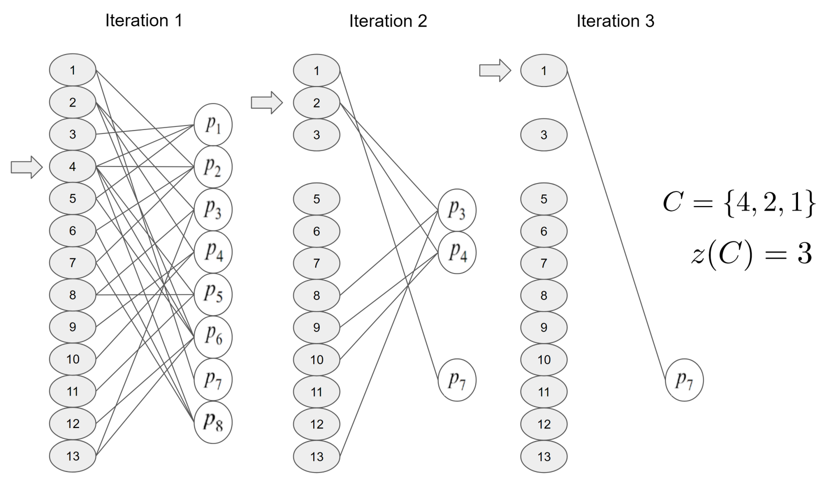


Figure 2.3: Example of application of the Greedy covering heuristic.

- Random (RC): Select randomly one link in R_C, remove it from C, and redefine R_C. Repeat this procedure until C no longer contains redundant links;
- Greedy (GC): Remove all links in R_C from C, and apply Algorithm 2 to the remaining uncovered paths by C; and
- Exact (EC): Remove all links in R_C from C, and apply the exact approach (described below) to the remaining uncovered paths by C.

Exact covering method

This method solves the integer programming model (2.1)-(2.3) for the current set of paths P_1, respecting a pre-established time limit TL. As mentioned in Section 2.2, problem (2.1)-(2.3) is NP-Hard, consequently T_L must be carefully defined.

The *Covering* phase of Algorithm 1 receives as input the current set of paths $P_1 \subseteq P$. Therefore, the exact solution found by solving model (2.1)-(2.3) is a lower bound for the original TCLP problem, i.e., it is a partial covering.

Despite each iteration in Algorithm 1 does not always yield a complete cover for the original problem, the lower bounds found at each iteration are valid for the TCLP, and help to prove the optimality with the support of upper bound for the TCLP.

Hybrid covering heuristic

The hybrid covering heuristic is based on two premises: (i) the number of paths is much greater than the number of links, so it is expected that the first links selected by the greedy heuristic of Algorithm 2 are good choices, covering most paths; and (ii) the exact method is well suited for the USCP with a small set of paths, i.e., the computational time is small when the USCP to be solved is also small.

Algorithm 3 details the Hybrid Covering Algorithm. Let $\alpha \in [0,1]$ be a scalar. Thus, our hybrid covering heuristic (i) applies the greedy covering heuristic presented in Section 2.3.1 (Lines 1−5), selecting links l until at least $(1-\alpha) \cdot |P_1|$ paths have been covered generating the cover C_1; and (ii) solves the remaining USCP composed by P^* by the exact method (Line 6) described in Section 6, generating the cover C_2. The final cover C is the union of C_1 and C_2. It is worth to highlight that if $\alpha = 1$, the greedy covering heuristic (Lines 1−5) is skipped.

Algorithm 3: Hybrid Covering Heuristic

```
Data: Current Set of Paths $P_1$, Scalar $\alpha$, Time limit $TL$
Result: Cover $C$ for $P_1$
1 $C_1 \leftarrow \emptyset$; $P^* \leftarrow P_1$ ;
2 while $|P^*| > \alpha\,|P_1|$ do
3     $l \leftarrow \text{argmax}_{a \in A}\, |P^*_a|$;
4     $C_1 \leftarrow C_1 \cup \{l\}$; ;
5     $P^* \leftarrow P^* \setminus P^*_l$ ;
6 $C_2 \leftarrow Exact(P^*, T_L)$ ;
7 $C \leftarrow C_1 \bigcup C_2$ ;
```

For example, when we apply this hybrid heuristic to the example illustrated in Figure 2.2 with $\alpha = 0.5$, $l = 4$ is the first and only link selected by the heuristic, since it covers 5 paths, and $\alpha \cdot |P_1| = 0.5 \cdot 8 = 4$. Link 4 covers paths p_1, p_2, p_5, p_6 and p_8, so the set of uncovered paths $\{p_3, p_4, p_7\}$ is solved by the exact method.

2.3.2 Cut phase

As mentioned before, after solving the *Covering* phase of Algorithm 1, new paths can be added to P_1 to prevent sensors located in that phase from being bypassed by the uncovered paths. That is precisely the purpose of the *Cut* phase, which searches for paths violating the last *Covering* solution, i.e., paths uncovered by the set of links selected to receive traffic counting stations. In this context, as already mentioned, it is important to manage the growth of the set P_1, during the *Cut* phase, in order to reduce the computational time required to solve the SCP at each loop of Algorithm 1.

That said, we propose a one-to-all Dijkstra [34] variation, to search for uncovered paths between pairs of O-D nodes. Given $G = (N, A)$ and a starting node $i \in T$, the search algorithm manages a set of visited nodes V (initially $V = \{i\}$) and a set of candidate notes $F := \{j \,|\, \exists\, (i,j) \in A', i \in V, j \in N \setminus V\}$. It iteratively moves one node $j \in F$ to V, passing only through links in $A' = A \setminus C$ in order to find paths uncovered by C. The spread criteria *SpreadCrit* is the one responsible to make the implementation of this Dijkstra variation, deciding which node $j \in F$ will be added into V.

The separation procedure used in the *Cut* phase is described by Algorithm 4, using this Dijkstra algorithm. For every path p_{ij} found, if $j \in T$, p_{ij} is added to P_2. Since each link in the network is bidirectional and we intend to avoid redundant paths in the model, we only add in P_2 paths from $i \in T$ to $j \in T$ such that $i < j$.

Algorithm 4: Cut generation

```
Data: Network $G = (N, A)$, Cover $C$ and the Spread Criterion
      ($SpreadCrit$)
Result: Set of New Paths $P_2$
1 $G' \leftarrow (N, A \setminus C)$;
2 $P_2 \leftarrow \emptyset$;
3 for $i \in T$ do
4     $P_2 \leftarrow P_2 \cup \{\, p_{ij} \,|\, j \in T,\ i < j,\ p_{ij} \in Dijkstra(G', i, SpreadCrit)\}$;
```

Every uncovered path between O-D pairs is a valid Constraint (2.2), so we can (i) select random paths; (ii) select minimum paths in relation to distance [35]; or (iii) use the integer programming theory which suggests that constraints with few variables make the model closer to the convex hull of its integer feasible solutions [36–38], to select minimum paths in relation to the number of links per path. From these ideas, we generated three spread criteria *SpreadCrit* to represent each type of cut generation (or separation) procedure:

- Non Discriminated (ND): Select next node randomly;
- Minimum Distance (MD): Select next node minimizing the total distance from initial node. If there are equal Minimum paths, select one randomly; and
- Minimum Link Count (MLC): Select next node minimizing the number of links from the initial node, in such a way that the Dijkstra algorithm becomes a breadth first search (BFS). If there are equal minimum paths, one is selected randomly.

We are now able to present our proposed algorithm to solve the TCLP.

2.3.3 Progressive Covering Algorithm (PCA)

After many experiments, Algorithm 5 showed to be very robust and flexible to solve the TCLP. It initially receives the Network G; its set W of O-D pairs; initial hybrid scalar $\alpha_0 < 1$, controlling the initial "hybridness" of the method; time limit for each iteration TL_1, which is the CPU time limit imposed to the exact covering method in each iteration; and a total time limit TL_2. The CPU time spent by other functions was too small to be limited. Algorithm 5 consists of progressively solving the TCLP with a Hybrid approach and every time the problem is solved for a given α, α is increased, progressively finding better solutions in the process.

More specifically, in the main loop between Lines 3-8: a cover C_1 is generated in Line 4; if C_1 is better than the current cover C, C_1 becomes the current cover C (Line 5); and, independently, α is doubled in Line 7. When $\alpha > 1$, Algorithm 5 usually ends finding a feasible cover C within the total given time limit TL_2; when this is not the case, in Lines 9-10 we force one to be found by applying the Greedy covering heuristic.

Algorithm 5: Progressive Covering Algorithm (PCA)

```
Data: Network $G$, set $W$, initial hybrid scalar $\alpha_0 < 1$, Time limit iteration
      $TL_1$, Time limit total $TL_2$
Result: A solution set $C$ of links to receive traffic counting stations
        (sensors).
1 $P_1 \leftarrow \emptyset$; $\alpha \leftarrow \alpha_0$; $C \leftarrow A$;
2 repeat
3     $C_1 \leftarrow SCF(P_1, Hybrid(\alpha), EC, MLC, TL_1, TL_2 - Time)$;
4     if $Cost(C_1) < Cost(C)$ then $C \leftarrow C_1$;
5     $\alpha \leftarrow 2 * \alpha$;
6 until $\alpha > 1$;
7 if $Cost(C) = |A|$ then $C \leftarrow SCF(P_1, Greedy, EC, MLC, \infty, \infty)$ ;
```

2.4 Computational experiments

All the algorithms presented in Section 2.3 were coded in C, compiled with GCC 10.0, and Gurobi 9.01 was used in the Exact covering method. All experiments were performed on a computer with an AMD R9 processor 3900X @4.15 GHz and 32 GB DDR4 of RAM memory, under Ubuntu 18.04 x64 operating system. The reported CPU times are related to single threaded processing times.

The set of instances is composed of 26 real-world cases (see Table 2.1), all derived from the Brazilian states. Each node of origin (or destination) represents a municipality, and each link represents a segment of state or federal highway. These data were extracted from a georeferenced database proposed by GONZÁLEZ *et al.* [1]. In Table 2.1, column Instance represents the state, $|N|$ is the number of nodes, $|A|$ is the number of links, $|T|$ is the number of O-D nodes and $|W|$ is the number of O-D pairs.

Section 2.4.1 presents the internal testing process for *Cleaning* and *Cut* procedures presented in Section 2.3. In these tests, Algorithm 1 was run 30 times for each instance.

Table 2.1: Instances' attributes.

Instance	$\lvert N\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert W\rvert$	Instance	$\lvert N\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert W\rvert$
AC	61	84	20	190	**PB**	384	480	213	22578
AL	169	219	97	4656	**PE**	362	472	172	14706
AM	74	77	37	666	**PI**	405	550	212	22366
AP	52	77	13	78	**PR**	780	1083	381	72390
BA	812	1113	395	77815	**RJ**	502	721	86	3655
CE	401	612	177	15576	**RN**	333	433	160	12720
ES	283	394	75	2775	**RO**	185	258	50	1225
GOeDF	799	1165	241	28920	**RR**	75	97	13	78
MA	257	355	163	13203	**RS**	675	859	391	76245
MG	1474	1917	803	322003	**SC**	481	604	266	35245
MS	343	497	76	2850	**SE**	183	248	74	2701
MT	711	1069	140	9730	**SP**	1280	1683	606	183315
PA	289	370	122	7381	**TO**	372	524	134	8911

Finally, our main results are compared with the other results in literature in Section 2.4.2. In these tests, Algorithm 5 with $\alpha_0 = 1/16$ and $TL_1 = 400s$ was run 15 times and Algorithm 5 with $\alpha_0 = 1$ and $TL_1 = 1200s$ was run once for each instance.

2.4.1 Calibration of Cleaning and Cut procedures

Table 2.2 depicts the statistics that summarize the behavior of Algorithm 1 with $Cover$="Greedy covering heuristic" without time limits varying the Cleaning procedures for all 26 instances. The first column indicates the type of Cleaning method used, except for the first line, which indicates that no cleaning method was used. The second and third columns show the deviations (in percentage) of the best result and the results' average related to the Best-Known Solution (BKS) for each method. The fourth column depicts the average CPU times (in seconds) spent for each method. The results indicate that, when the cleaning method is not applied, the greedy covering heuristic is the fastest, but it provides the worst quality solutions, presenting the greatest deviations in relation to BKS. In contrast, when the exact cleaning method is used, we found the best results with a small increase in the CPU time. The information about all results found for the calibration of cleaning and cut procedures are reported in the Appendix.

Table 2.2: Average results of the Greedy covering heuristic, when varying the Cleaning methods.

Cleaning Type	Best (%)	Average (%)	CPU(s)
No Cleaning	1.69	3.10	0.92
Random (RC)	1.40	2.87	1.31
Greedy (GC)	1.50	2.79	1.06
Exact (EC)	1.18	2.16	1.09

Based on the results of Table 2.2, we decided to use the Exact method as the standard Cleaning procedure for the Greedy covering heuristic in all remaining tests.

Figure 2.4 illustrates the behavior of the Algorithm 1 with *Cover*="Exact covering method", in terms of normalized minimum, average and maximum CPU times, for 10 of the 26 instances that are usually solved in less than 10 minutes. The abscissa shows each spread criterion (*SpreadCrit*=ND, MD, MLC) of the Cut generation procedure; and the ordinate depicts the normalized average CPU times.

Figure 2.4 shows that the Minimal Link Count (MLC) provided objectively the best results minus less than 5% of the cases where ND or MD can provide better exact CPU times, corroborating the polyhedral theory which suggests that constraints with few variables make the model closer to the convex hull of its integer feasible solutions, as mentioned in Section 2.3.2.

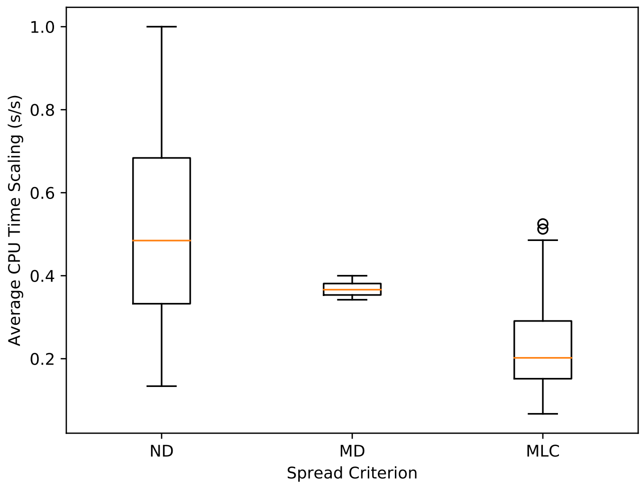


Figure 2.4: Behavior of the Exact covering method, in terms of normalized solution times, when varying the Cut generation procedures.

Based on the results illustrated in Figure 2.4, we decided to use the Minimal Link Count (MLC) as the standard Cut generation procedure in Algorithm 5.

2.4.2 Literature Comparison

Table 2.3 provides a comparison between the results found in this work with TL_2 fixed as $7200s$, by Algorithm 5 with $\alpha_0 = 1/16$, $TL_1 = 500s$, focused in finding better feasible solutions (Columns $5-7$) and Algorithm 5 with $\alpha_0 = 1$, $TL_1 = 1200s$, focused in proving optimality (Columns $11-14$, where LB is the lower bound found) and the results found by GONZÁLEZ *et al.* [1], using a Clustering Search (CS) heuristic (Columns $2-4$) and a Branch-and-Cut algorithm (B&C) (Columns $8-10$). The best solution costs, the average solution costs, and the CPU times (in seconds) are presented for the heuristic approaches; and the solution costs, the residual optimality gaps and the CPU times (in seconds) are shown for the exact approaches. The values in italic mean that they are optimal for that algorithm, the bold ones indicate the BKS, and the new BKSs are highlighted in underlined form, always in relation to each algorithm. The last two lines show the deviations (in percentage) of the best result and the results' average related to the BKS for each method.

Analyzing the heuristic results shown in Table 2.3, we can conclude that our PCA (with $\alpha_0 = 1/16$ and $TL_1 = 500$) found results equal or better than CS [1] in all instances, providing optimal solutions for 20 of them with lower CPU times. We provided better averages for RJ and GO-DF instances, and we obtained new BKSs for the hardest instances MT, MG and SP.

Regarding the exact methods, the B&C of GONZÁLEZ *et al.* [1] solved only four instances to optimality, whereas our PCA with $\alpha_0 = 1$ and $TL_1 = 1200s$ found 20 optimal solutions with much lower CPU times. At last, when combining the results of both PCAs related to MG and MS, we can ensure that the solutions of 1119 and 150, respectively, are optimal values since the upper bounds of PCA with $\alpha_0 = 1/16$, $TL_1 = 500s$ match the lower bounds of PCA with $\alpha_0 = 1$, $TL_1 = 1200s$. So, in the end, we were able to find optimal solutions for 22 of the 26 tested instances. All results found for each approach presented in Table 2.3 are reported in the Appendix.

Table 2.3: Comparison between our heuristic and exact approaches and the CS and B&C solutions of GONZÁLEZ *et al.* [1]

	Clustering Search (CS)			PCA, $\alpha_0 = 1/16$, $TL_1 = 500$			Branch-and-Cut (B&C)			PCA, $\alpha_0 = 1$, $TL_1 = 1200$			
Instance	**Best**	**Avg**	**CPU(s)**	**Best**	**Avg**	**CPU(s)**	**Sol**	**GAP(%)**	**CPU(s)**	**Sol**	**GAP(%)**	**LB**	**CPU(s)**
AC	**30**	**30.00**	0.1	***30***	**30.00**	0.1	*30*	0.00	0.6	*30*	0.00	30	0.2
AL	**137**	**137.00**	0.4	***137***	**137.00**	0.3	**137**	12.50	7200.0	*137*	0.00	137	0.4
AM	**39**	**39.00**	0.0	***39***	**39.00**	0.0	*39*	0.00	6.5	*39*	0.00	39	0.0
AP	**22**	**22.00**	0.1	***22***	**22.00**	0.1	*22*	0.00	0.1	*22*	0.00	22	0.1
BA	**630**	**630.00**	750.4	***630***	**630.00**	11.3	**630**	19.31	7200.0	*630*	0.00	630	5.8
CE	**329**	**329.00**	143.8	***329***	**329.00**	104.4	331	20.06	7200.0	*329*	0.00	329	55.0
ES	**144**	**144.00**	18.0	***144***	**144.00**	10.6	**144**	21.18	7200.0	*144*	0.00	144	18.6
GO-DF	**464**	465.50	1045.8	**464**	464.27	7200.6	483	31.90	7200.0	477	3.92	459	7200.8
MA	**250**	**250.00**	0.3	*250*	**250.00**	0.1	**250**	8.60	7200.0	*250*	0.00	250	0.1
MG	1122	1124.70	6191.8	**1119**	**1119.00**	6152.3	1131	22.81	7200.0	1130	0.98	1119	7201.9
MS	**150**	**150.00**	832.5	**150**	**150.00**	7200.5	153	22.88	7200.0	155	3.33	150	7200.5
MT	310	311.80	4327.4	**299**	302.80	7200.5	316	38.45	7200.0	314	14.59	274	7200.5
PA	**174**	**174.00**	117.4	*174*	**174.00**	2.2	**174**	17.53	7200.0	*174*	0.00	174	1.9
PB	**296**	**296.00**	2.63	*296*	**296.00**	6.9	**296**	14.53	7200.0	*296*	0.00	296	8.9
PE	**252**	**252.00**	94.2	*252*	**252.00**	20.7	253	22.02	7200.0	*252*	0.00	252	26.6
PI	**316**	**316.00**	50.5	*316*	**316.00**	1.7	**316**	14.67	7200.0	*316*	0.00	316	0.9
PR	**599**	599.80	955.6	*599*	**599.00**	259.1	603	21.56	7200.0	*599*	0.00	599	228.3
RJ	**166**	167.70	1485.2	**166**	166.60	7200.6	169	32.74	7200.0	173	9.49	158	7200.5
RN	**233**	**233.00**	2.7	*233*	**233.00**	2.3	**233**	17.60	7200.0	*233*	0.00	233	1.3
RO	**88**	**88.00**	5.57	*88*	**88.00**	10.9	**88**	15.34	7200.0	*88*	0.00	88	7.9
RR	**19**	**19.00**	9.0	*19*	**19.00**	0.1	*19*	0.00	0.1	*19*	0.00	19	0.1
RS	**544**	**544.00**	250.4	*544*	**544.00**	39.1	550	18.27	7200.0	*544*	0.00	544	24.7
SC	**371**	**371.00**	149.9	*371*	**371.00**	2.9	**371**	14.96	7200.0	*371*	0.00	371	2.9
SE	**112**	**112.00**	9.6	*112*	**112.00**	1.0	**112**	9.37	7200.0	*112*	0.00	112	1.2
SP	874	877.30	1469.8	**872**	872.73	7200.6	891	28.90	7200.0	890	3.85	857	7201.1
TO	**231**	**231.00**	35.4	*231*	**231.00**	11.9	232	22.65	7200.0	*231*	0.00	231	12.8
Dev (%)	0.16	0.26		0.00	0.07		0.77			0.71			
Time (s)			683.12			1640.03			6092.59				1677.04

2.5 Final remarks of the chapter

This chapter proposed a progressive hybrid algorithm based on set covering to solve the TCLP. Greedy, exact and hybrid methods, based on a simple and innovative concept which has not yet been explored in the literature, were developed embedded in a set covering framework to solve 26 real-world instances obtained from the Brazilian states.

Our algorithm found better results than the clustering search of [1], providing best-known solutions for the hardest instances MG, MT and SP. Regarding the exact methods, the branch-and-cut algorithm of [1] solved only four instances to optimality, whereas we were able to find 20 optimal solutions with lower CPU times. Besides, two new optimal solutions were found after evaluation of the results.

In view of these results, our work brings a good contribution to the location of traffic counters on highways. In particular, such results are now supporting the decision makers of the National Traffic Counting Plan in Brazil.

Regarding future works, we suggest to solve the case that uses partial coverage to maximize the total number of O-D pairs covered, when considering a pre-established budget for the sensors. New heuristics and metaheuristics could be proposed to generate Pareto curves to analyze the trade-off between maximizing the coverage of the traffic network and minimizing the cost of installing the sensors.

Chapter 3

Exact and heuristic algorithms for the fleet composition and periodic routing problem of offshore supply vessels with berth allocation decisions

This chapter considers the periodic supply vessel planning problem with berth allocations (PSVPP-BA). Although the problem solved is the one considered by CRUZ *et al.* [2], our approach is new in two important aspects: (*i*) *non-clustered clients*, expanding the search space, even knowing that the oil company suggest these clusters; this approach allows to handle a wider range of instances; and (*ii*) *circular planning horizon*, constraining the search space; such an approach makes the generated solutions easier to replicate indefinitely due to the problem's "non-stop operations" characteristic. Both features make the problem more general and harder to solve.

In order to quickly and efficiently solve the largest available benchmark instances, we have developed a branch-and-cut algorithm and an adaptive large neighborhood search (ALNS) heuristic with multiple starts and spaced local searches. The proposed algorithms were published in VIEIRA *et al.* [39].

3.1 Literature review

The PSVPP belongs to the family of periodic vehicle routing problems (PVRP), an important branch of the classical vehicle routing problem since many practical applications impose periodic visits to the customers during the planning horizon, as shown in BAPTISTA *et al.* [40].

The best available exact and metaheuristic algorithms for the PVRP can be found in BALDACCI *et al.* [41] and in VIDAL *et al.* [16]. The PSVPP is significantly more complicated than the classical PVRP since the supply vessels make multiple-day voyages, some of the offshore unit have time windows for service, and the vessels' departures from the onshore base should be spread as evenly as possible. The literature devoted to the supply vessels planning problems is rather limited. The seminal paper of FAGERHOLT and LINDSTAD [42] dealt with a simplified version of the PSVPP that did not consider the assignment of voyages to the days of the week, and did not directly address the required spread of departures from the base. The candidate routes were generated by considering the night closures of some offshore units. This paper was followed by the works of AAS *et al.* [43] and GRIBKOVSKAIA *et al.* [9], which tackled the pickup and delivery problem involving one single vessel under limited storage capacity at the offshore units and on the vessels. GRIBKOVSKAIA *et al.* [9] developed a tabu search metaheuristic to solve large instances.

HALVORSEN-WEARE *et al.* [44] incorporated periodicity in route planning, including the departing day in the route selection decision variable. These authors developed a two-phase algorithm which first generates feasible candidate voyages, and then uses these voyages as an input for a set covering model. SHYSHOU *et al.* [45] pursued this research by introducing a large neighborhood search heuristic (LNS) to solve large instances of the PSVPP, which was quite efficient compared with the two-phase approach. HALVORSEN-WEARE and FAGERHOLT [12] introduced an alternative arc-flow model which did not perform well as the voyage-based model.

FERNÁNDEZ CUESTA *et al.* [46] also studied a pickup and delivery problem considering a single and multiple vessels. They added the possibility of not servicing all offshore units by introducing a penalty associated with the losses due to the unattended demand. They also considered the transportation of all cargo to be compulsory, which led to the need of expanding the vessel fleet. BORTHEN *et al.* [47] adapted the genetic search heuristic of VIDAL *et al.* [16] to a simplified version of the PSVPP with a single and fixed departure time, homogeneous fleet of vessels and no time windows. This was followed by BORTHEN *et al.* [48] who introduced a bi-objective approach, this time trying to minimize changes from the current solution whenever offshore units are added or removed from the current plan. KISIALIOU *et al.* [11] extended the work of HALVORSEN-WEARE *et al.* [44] by allowing flexible departure times from the onshore base within discrete departure slots and considering coupled vessels. They proposed an ALNS heuristic to solve this problem and compared its results with a voyage-based formulation solved by CPLEX.

BIERWIRTH and MEISEL [49] presented a comprehensive overview of berth allocation problems. An ALNS implementation was made in MAURI *et al.* [50] in order to assign ships to berthing positions along a quay in a port. This was followed by ÇAĞATAY IRIS *et al.* [51] who presented improved formulations. CRUZ *et al.* [2] were the first to simultaneously integrate berth allocation decisions with the determination of the heterogeneous fleet composition and the vessels schedules in the PSVPP, considering continuous and flexible departures from the base and historical data to model the berth allocations and departures. They proposed a multi-step model to solve exactly real-world large instances comprising up to 79 offshore units. KISIALIOU *et al.* [52] developed an ALNS heuristic to generate robust supply vessel schedules under stochastic weather conditions, calculating the expected level of service for different sets of vessels.

3.2 Mathematical modelling

This section presents our mathematical models for the PSVPP-BA which, differently from CRUZ *et al.* [2], couples individual vessels to voyages (routes). Let $G = (N, E)$ be an undirected graph in which the set of nodes $N = N^B \cup N^{OI}$ is made up of the set of berths N^B and the set of offshore units N^{OI}, and the set $E = \{(i, j)|i, j \in N, i < j\}$ contains the edges between the nodes in N. Every offshore unit $i \in N^{OI}$ is periodically served by a set V of vessels of types in V', and each vessel departs from a berth $b \in N^B$ in a time horizon consisting of a set T of periods that can be infinitely replicated.

Each offshore unit $i \in N^{OI}$ has a demand q_i expressed in square meters (m^2), a service time t_i expressed in periods, and must be served f_i times within the time horizon. Thus, a set of delivery patterns P_i is associated to each offshore unit $i \in N^{OI}$, corresponding to f_i services. As mentioned in Section 1.2, the production units are serviced typically once or twice a week, while drilling rigs are commonly serviced at least three times a week. Each vessel type $v \in V'$ has a capacity U_v, a travel speed S_v, a fixed cost c^f_v, and a variable cost c^v_v per distance unit traveled in the time horizon. Since T usually consists of the seven days of the week, each vessel type $v \in V'$ can perform two or three voyages per week. Each berth $b \in N^B$ has a fixed maneuvering and startup time per vessel L^f_b, a variable loading rate L^q_b (in terms of period/m^2) and a queue capacity K_b with two departing positions $k \in K_b = \{1, 2\}$, due to problem attribute "Two daily berth departures". As mentioned in the Section 1.2, under "Berth departures' tolerance", a vessel may start loading in one period $t \in T$ and keep loading during period $t + 1$ with a departure time never exceeding half of that period. Such a tolerance, for each period and berth, is controlled in the model by the constant $TOL = 1.5$.

3.2.1 Complete model

The first proposed mathematical formulation is a voyage-based model in which every voyage r belongs to the set of all non-dominated feasible voyages R, composed of a set of sequentially served offshore units $r = \{i, i \in N^{OI}\}$, with total load $Q_r = \sum_{i \in r} q_i$, total distance D_r, and total service time $T_r = \sum_{i \in r} t_i$. The voyages are generated by enumerating all combinations of offshore units with feasible load for the largest vessel and following the "Route duration" problem attribute (Section 1.2), and then a Traveling Salesman Problem [53] is solved and the generated route is added to R. For each $r \in R$ there is a vessel type $v \in V'$ such that $Q_r \leq U_v$. Each vessel type in $v \in V'$ is associated to a subset $R_v = \{v \in V' \mid Q_r \leq U_v\}$ containing feasible voyages for $v \in V'$.

The constant M is a big positive number and $|T|$ is the total number of periods. The following binary matrices are used in the mathematical formulation: M^1_{ir} indicates whether offshore unit $i \in N^{OI}$ belongs or not to voyage $r \in R$; M^2_{pt} indicates whether delivery pattern $p \in P_i$ allows or not offshore unit $i \in N^{OI}$ to be serviced for some voyage starting in period $t \in T$, and M^3_{ltvr} indicates whether voyage $r \in R_v$ started in period $t \in T$ with vessel type $v \in V'$ is active or not in period $l \in T$. Due to the periodic propriety, $M^3_{ltvr} = M^3_{(l+|T|)tvr}$.

The binary variables are the following: $x_{rtvnbk} = 1$ indicates that route $r \in R_v$ starts at period $t \in T$, with vessel $n \in V$ of type $v \in V'$, on berth $b \in N^B$ and position $k \in K_b$, otherwise, $x_{rtvnbk} = 0$; $y_{vn} = 1$ indicates that vessel $n \in V$ of type $v \in V'$ is used, otherwise, $y_{vn} = 0$; $z_{ip} = 1$ determines that delivery pattern $p \in P_i$ is selected for offshore unit $i \in N^{OI}$, otherwise $z_{ip} = 0$.

The integer variable t_{vnt} represents the number of periods occupied by vessel $n \in V$ of type $v \in V'$ starting at period $t \in T$. The continuous variables are as follows: t^{WB}_{bt} is the waiting time for berth $b \in N^B$ at period $t \in T$ imposed by the previous period; t^{LB}_{btk} represents the loading time for berth $b \in N^B$ in position $k \in K_b$ and period $t \in T$; and finally t^{WV}_{vnt}, t^{LV}_{vnt} and t^{RV}_{vnt} respectively define, the wait, the load and the route time for vessel $n \in V$ of type $v \in V'$ starting at period $t \in T$.

The complete mathematical formulation is as follows:

$$\text{Minimize} \quad \sum_{v \in V'} \sum_{n \in V} (c^f_v y_{vn} + \sum_{r \in R_v} \sum_{t \in T} \sum_{b \in N^B} \sum_{k \in K_b} c^v_v D_r x_{rtvnbk}) \tag{3.1}$$

subject to

$$\sum_{p\in P_i} z_{ip} = 1 \qquad i \in N^{OI} \tag{3.2}$$

$$\sum_{v\in V'}\sum_{n\in V}\sum_{r\in R_v}\sum_{b\in N^B}\sum_{k\in K_b} M^1_{ir}x_{rtvnbk} = \sum_{p\in P_i} M^2_{pt}z_{ip} \qquad i \in N, t \in T \tag{3.3}$$

$$\sum_{l\in T}\sum_{r\in R_v}\sum_{b\in N^B}\sum_{k\in K_b} M^3_{ltvr}x_{rtvnbk} \leq y_{vn} \qquad v \in V', n \in V, t \in T \tag{3.4}$$

$$\sum_{v\in V'}\sum_{n\in V}\sum_{r\in R_v} x_{rtvnbk} \leq 1 \qquad b \in N^B, k \in K_b, t \in T \tag{3.5}$$

$$t^{LB}_{btk} = \sum_{v\in V'}\sum_{n\in V}\sum_{r\in R_v} (L^f_b + Q_r L^q_b)x_{rtvnbk} \qquad b \in N^B, k \in K_b, t \in T \tag{3.6}$$

$$t^{WB}_{bt} + \sum_{k\in K_b} t^{LB}_{btk} \leq TOL \qquad b \in N^B, t \in T \tag{3.7}$$

$$t^{WB}_{b(t+1)} \geq t^{WB}_{bt} + \sum_{k\in K_b} t^{L}_{btk} - 1 \qquad b \in N^B, t \in T \tag{3.8}$$

$$t^{WB}_{b(|T|+X)} = t^{WB}_{bX} \qquad X \in \mathbb{Z}, b \in N^B, t \in T \tag{3.9}$$

$$t^{WV}_{vnt} \geq t^{WB}_{bt} + \sum_{\alpha<k} t^{LB}_{bt\alpha} + (\sum_{r\in R_v} x_{rtvnbk} - 1)M \qquad v \in V', n \in V, t \in T, b \in N^B, k \in K_b \tag{3.10}$$

$$t^{LV}_{vnt} = \sum_{r\in R_v}\sum_{b\in N^B}\sum_{k\in K_b} (L^f_b + Q_r L^q_b)x_{rtvnbk} \qquad v \in V', n \in V, t \in T \tag{3.11}$$

$$t^{RV}_{vnt} = \sum_{r\in R_v}\sum_{b\in N^B}\sum_{k\in K_b} (T_r + \frac{D_r}{S_v})x_{rtvnbk} \qquad v \in V', n \in V, t \in T \tag{3.12}$$

$$t_{vnt} \geq t^{WV}_{vnt} + t^{LV}_{vnt} + t^{RV}_{vnt} \qquad v \in V', n \in V, t \in T \tag{3.13}$$

$$\sum_{t\in T} t_{vnt} \leq |T| \qquad v \in V', n \in V \tag{3.14}$$

$$y_{vn} \in \{0,1\} \qquad v \in V', n \in V \tag{3.15}$$

$$t^{WB}_{bt} \in [0, TOL-1] \qquad b \in N^B, t \in T \tag{3.16}$$

$$t^{LB}_{btk} \geq 0 \qquad b \in N^B, k \in K_b, t \in T \tag{3.17}$$

$$t^{WV}_{vnt} \geq 0, t^{LV}_{vnt} \geq 0, t^{RV}_{vnt} \geq 0 \qquad v \in V', n \in V, t \in T \tag{3.18}$$

$$t_{vnt} \in \mathbb{Z}^+ \qquad v \in V', n \in V, t \in T \tag{3.19}$$

$$z_{ip} \in \{0,1\} \qquad i \in N^{OI}, p \in P_i \tag{3.20}$$

$$x_{rtvnbk} \in \{0,1\} \qquad r \in R_v, t \in T, v \in V', n \in V, b \in N^B, k \in K_b. \tag{3.21}$$

The objective function (3.1) minimizes the sum of fixed and variable sailing costs. Constraints (3.2) and (3.3) ensures that any selected set of routes fulfills the periodic service for each offshore unit, servicing them with spread departures enforcing the

"Periodic service" and "Spread departures from the base" problem attributes (Section 1.2). Constraints (3.4) guarantee that the active routes in each period $t \in T$ for each vessel $n \in V$ of type $v \in V'$ do not overlap, managing the heterogeneous fleet. Constraints (3.5) ensure that at most one route $r \in R_v$ with vessel $n \in V$ of type $v \in V'$ using berth $b \in N^B$ and position $k \in K_b$ in period $t \in T$ can exist. Constraints (3.6) manage the berth loading times. Constraints (3.7) guarantee that the berth loading and waiting times do not exceed the operational tolerance TOL in any period. Since $TOL = 1.5$, these constraints enforce that the current period cannot take more than half (12h) of the next period, according to the "Berth departures' tolerance" problem attribute (Section 1.2). Constraints (3.8)−(3.9) state that if the loading time of some berth exceeds one period, then the exceeding amount of time is carried over the next period as waiting time. Constraints (3.10)−(3.14) manage the loading, waiting and total routing times for the vessels, restricting them to the planning horizon. Lastly, Constraints (3.15)−(3.21) define the domains of the variables.

In order to illustrate some aspects of this model, Figure 3.1 shows a case with routes $R = \{A, B, C, D\}$, one berth $N^B = \{1\}$, using two vessels of type 1, so $V = \{1, 2\}$. Route A loads and departures at period 2, Route B loads at period 6 but departures at period 7, route C loads at period 7 after route B and departures at period 1 and route D loads and departures at period 4. Since no pair of routes starts at the same period, all routes use only the first berth position. Thus, the set of variables equal to one is $X = \{x_{A,2,1,1,1,1}, x_{B,6,1,1,1,1}, x_{C,7,1,2,1,1}, x_{D,4,1,2,1,1}\}$. Let the total loading time to routes A, B, C and D be, respectively, 0.6, 1.3, 0.9 and 0.5 periods, so $t^{LB}_{1,2,1} = 0.6$, $t^{LB}_{1,4,1} = 0.5$, $t^{LB}_{1,6,1} = 1.3$ and $t^{LB}_{1,7,1} = 0.9$, following Constraints (3.7). Considering Constraints (3.8) and (3.7), for period 6, $t^{WB}_{1,7} \geq 1.3 - 1$ and $1.3 \leq 1.5$. For period 7, the exceeding loading time from period 6 (0.3) is carried as the waiting berth time $t^{WB}_{1,7}$, so following Constraints (3.8) and (3.7), for period 6, $t^{WB}_{1,8} \geq 0.3 + 0.8 - 1$ and $0.3 + 0.9 \leq 1.5$ and they are all satisfied, with Constraints (3.9) forcing the exceeding loading time from period 7 (0.2) to be carried to period 1 with $t^{WB}_{1,7+1} = t^{WB}_{1,1}$.

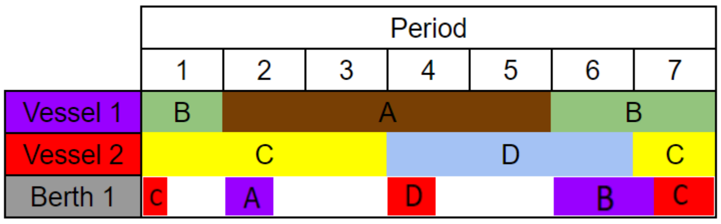


Figure 3.1: Example of a vessel and berth cyclical scheduling.

Finally, Constraints (3.4) verify in each time period $t \in T$, for each vessel $n \in V$

of type $v \in V'$, all the routes that have started in any previous period and are still active in period t. Therefore, they are necessary restrictions to avoid collisions when starting routes for each vessel, but they are not sufficient since they do not consider the berth dependant loading times. Let s^{comp} be an integer solution of model (3.1)−(3.21). Let S_x^{comp} be the set composed by the "x_{rtvnbk} part" of s^{comp} such that $x_{rtvnbk} = 1$. Then, S_x^{comp} is a set of vectors representing only the selected routes. In order to illustrate the non-sufficiency of Constraints (3.4), suppose that $S_x^{comp} = \{x_{37,2,3,1,1,2},\ x_{24,2,3,2,1,1},\ x_{82,5,3,1,1,1},\ x_{56,6,3,2,1,1}\}$ and, in period 2, vessel 1 of type 3 has a waiting time equal to 0.25 day, and a joint loading and routing time equal to 2.80 days. Then, we have one berth with two positions, and two vessels of type 3 performing two routes each. Vessel 1 departs at $t = 2$ in route 37 and at $t = 4$ in route 82, and vessel 2 departures at $t = 2$ in route 24 and at $t = 6$ in route 56. This solution respects Constraints (3.4), but is infeasible in practice. In fact, the total route time for vessel 1 of type 3 is equal to $\lceil 2.80 + 0.25 \rceil$ $= 4$, thus this vessel will not be able to perform route 82 at $t = 4$. For this example, constraint $x_{37,2,3,1,1,2} + x_{24,2,3,2,1,1} + x_{82,5,3,1,1,1} + x_{56,6,3,2,1,1} \leq 3$ could be included in (3.1)−(3.21) to cut s^{comp}. However, only the set $\{x_{37,2,3,1,1,2},\ x_{24,2,3,2,1,1},\ x_{82,5,3,1,1,1}\}$ of variables is required to avoid the overlap, then it should be sufficient to use constraint $x_{37,2,3,1,1,2} + x_{24,2,3,2,1,1} + x_{82,5,3,1,1,1} \leq 2$.

Constraint (3.22), derived from s^{comp}, cuts this solution if it contains any overlapping route not considered by Constraints (3.4) in which we consider only the variables responsible for each overlap.

$$\sum_{(r,t,v,n,b,k)\in S_x^{comp}} x_{rtvnbk} \quad \leq \quad |S_x^{comp}| - 1. \tag{3.22}$$

Consequently, in a branch-and-bound tree, whenever a new integer solution s^{comp} of model (3.1)−(3.21) is found, a new Constraint (3.22), derived from S_x^{comp}, is added to (3.1)−(3.21) within a branch-and-cut framework.

3.2.2 Reduced model

The complete model (3.1)−(3.21) suffers from the curse of symmetry: it is possible to obtain different solutions, of the same objective value, by changing vessels of the same type or swapping berth schedules. The solutions obtained are different, but the routes, berth scheduling, fleets and related costs are the same. Another weakness of this model is the presence of the "bigM" coefficient in Constraints (3.10). These characteristics make the model (3.1)−(3.21) hard to solve.

In order to overcome these difficulties, we propose a branch-and-cut framework based on the better-conditioned reduced formulation (3.23)−(3.31). This formulation considers the binary variables x_{rtv} and z_{ip}. If $x_{rtv} = 1$, route $r \in R_v$ starts at period $t \in T$, with vessel type $v \in V'$; otherwise, $x_{rtv} = 0$. If $z_{ip} = 1$, delivery pattern $p \in P_i$ is selected for offshore unit $i \in N^{OI}$, as in (3.1)−(3.21); otherwise, $z_{ip} = 0$. It also considers the integer variables y_v which indicates how many vessels of type $v \in V'$ are used. Finally, if $c_{vtl} = 1$ indicates that some route with vessel type $v \in V'$, starting at period $t \in T$ and active for $l \in T$ periods is used; otherwise, $c_{vtl} = 0$. The reduced mathematical formulation is as follows:

$$\text{Minimize} \quad \sum_{v \in V'} c_v^f y_v + \sum_{t \in T} \sum_{v \in V'} \sum_{r \in R_v} c_v^v D_r x_{rtv} \tag{3.23}$$

subject to

$$\sum_{p \in P_i} z_{ip} = 1 \qquad i \in N^{OI} \tag{3.24}$$

$$\sum_{v \in V'} \sum_{r \in R_v} M^1_{ir} x_{rtv} = \sum_{p \in P_i} M^2_{pt} z_{ip} \qquad i \in N, t \in T \tag{3.25}$$

$$\sum_{l \in T} \sum_{r \in R_v} M^3_{ltvr} x_{rlv} \leq y_v \qquad t \in T, v \in V' \tag{3.26}$$

$$\sum_{v \in V'} \sum_{r \in R_v} x_{rtv} \leq D \qquad t \in T \tag{3.27}$$

$$y_v \in \mathbb{Z}^+ \qquad v \in V' \tag{3.28}$$

$$z_{ip} \in \{0,1\} \qquad i \in N^{OI}, p \in P_i \tag{3.29}$$

$$c_{vtl} \in \{0,1\} \qquad v \in V', t \in T, l \in T \tag{3.30}$$

$$x_{rtv} \in \{0,1\} \qquad r \in R_v, t \in T, v \in V'. \tag{3.31}$$

The objective function (3.23) minimizes the sum of fixed and variable sailing costs. Constraints (3.24) and (3.25) ensure the periodic service for each offshore unit. Constraints (3.26) allocate the fleet according to the maximum number of active routes per period and per vessel type. Constraints (3.27) ensure that no period exceeds the maximum number of departures D, which is twice the number of available berths. Constraints (3.28)−(3.31) define the domain of the variables. The sets of indexes and parameters are the same as defined in (3.1)−(3.21).

Although being similar to the models of HALVORSEN-WEARE and FAGERHOLT [12] and KISIALIOU *et al.* [11], the reduced formulation (3.23)−(3.31) is a voyage (route) based model that performs the route scheduling and handles the fleet sizing while having a smaller number of variables and no symmetry or "BigM" coefficients.

Analogously to Constraints (3.4), Constraints (3.26) are also necessary but not sufficient to avoid collisions when starting routes for each vessel in model (3.23) −(3.31). Let s^{red} be an integer solution of (3.23)−(3.31), obtained in some node of its branch-and-bound tree. Let S_x^{red} be the "x_{rtv} part" of s^{red} such that $x_{rtv} = 1$.

Constraints (3.32), derived from s^{red}, are generated through a separation procedure for each new incumbent solution in order to bring the solution space from (3.23) $-$(3.32) to (3.1) $-$(3.22) as detailed below.

$$\sum_{(r,t,v)\in S_x^{red}} x_{rtv} \leq |S_x^{red}| - 1. \tag{3.32}$$

Finally, as Constraints (3.26) have a larger search space than Constraints (3.4), the difference between them is fixed by Constraints (3.33)-(3.34), dynamically generated through a separation procedure as well, with S_{xv}^{red} being the variables from S_x^{red} with vessel of type v, F_v is the correct fleet size and l_{rv} is the amount of periods route $r \in R$ remains active with vessel type $v \in V'$, i.e. $l_{rv} := \lceil \mathit{LoadTime}(rv) + \mathit{RouteTime}(rv) \rceil$. Constraints (3.34) can all be statically generated, however in our experiments, dynamically generating them showed better results for non-clustered instances. The details of how the search spaces differ are explained in Section 3.3.2.

$$y_v - \sum_{(r,t,v)\in S_{xv}^{red}} c_{vtl_{rv}} \geq F_v - |S_{xv}^{red}| \tag{3.33}$$

$$c_{vtl_{rv}} \geq x_{rtv} \qquad (r,t,v) \in S_{xv}^{red}. \tag{3.34}$$

3.3 Branch-and-cut algorithm

The complete model (3.1)$-$(3.21) contains a berth scheduling subproblem which is in fact a feasibility problem, so it does not interfere on the objective function, but adds two indices, various constraints and variables for managing these constraints. This was one of the motivations for proposing the reduced model (3.23)$-$(3.31).

So, whenever an incumbent solution is generated for the reduced model, we can try to generate a feasible berth schedule from it. If such a solution is found, it is accepted; otherwise, it can be used to generate a suitable cut to reinforce the reduced model in order to approximate its solution space to that associated with the complete formulation. As mentioned in CODATO and FISCHETTI [54], when cuts based on infeasible sets are minimal and generated within low computational times, this approach generally produces positive results.

Therefore, this process of transforming an incumbent reduced solution into a feasible solution for the complete formulation consists in solving a feasibility subproblem that will be called *SubProblem*, and will be detailed in Section 3.3.1. However, this process is not simple, and requires a study of the reasons for infeasibility, detailed in Section 3.3.2, in order to guarantee the generation of valid cuts for the separation method, which will be detailed in Section 3.3.3.

3.3.1 Feasibility subproblem

The feasibility subproblem $SubProblem$ receives an incumbent solution s^{red} of the reduced problem (3.24)−(3.32) comprising a heterogeneous fleet and a set of selected routes with fixed vehicle types and periods. To transform this incumbent solution into a feasible solution for the complete problem, it is necessary to determine, for each route a particular vessel $n \in V$, a berth $b \in N^B$, and a berth position $k \in K_b$, while verifying constraints (3.2)−(3.22).

In this way, $SubProblem$ can be formally described as [(3.4)*−(3.21)*, (3.35)] where * means that r, t and v are fixed in the values obtained from the "x_{rtv} part" of s^{red} in these constraints.

$$\sum_{n \in V} \sum_{b \in N^B} \sum_{k \in K_b} x_{rtvnbk} = 1 \qquad (r,t,v) \in S_x^{red}, \qquad (3.35)$$

If $SubProblem$ returns a feasible solution, it will be feasible solution for the complete problem with a heterogeneous fleet, and a set of routes so that each route has a departure period, type and vessel identification, berth and berth position, while respecting all berth scheduling constraints; otherwise, the causes of infeasibility must be analyzed to generate an appropriated cut, as detailed in Section 3.3.2.

3.3.2 Causes of infeasibility

There are two cases in which an incumbent solution s^{red} for the reduced model cannot be feasible for the complete model: the impossibility of allocating routes to vessels, and the impossibility of allocating vessels to berths.

In the reduced formulation, Constraints (3.26) control the size and mix of the fleet, according to the number of active routes per period, and the solution s^{red} satisfies these constraints. However, this fleet may not be sufficient to satisfy Constraints (3.4) of the complete problem, leading to the first cause of infeasibility - the impossibility of allocating routes to vessels.

To illustrate this situation, Figure 3.2(a) shows a possible solution of the reduced formulation that comprises a set of routes with defined periods for a given vessel type. Constraints (3.26) define a fleet of $y_v = 3$ vessels, since in each period there is a maximum of three active routes. Therefore, in order to transform the s^{red} solution of Figure 3.2(a) into a feasible solution to the complete formulation, these routes must be served by three vessels and simultaneously verify Constraints (3.4).

However, Figure 3.2(b) shows a case that, at best, there will always be a route that will not be allocated to a vessel.

There is a pair of well-known combinatorial problems that explains what happens in this first case. Let $G = (V, E)$ be a graph, representing Figure 3.2, in which every vertex $i \in V$ is a route with fixed departure time and every edge $(i, j) \in E$ means that routes i and j are active at some period. So, finding the right-hand side y_v of Constraints (3.26) is equivalent to determining the maximum clique size of G, known as $\omega(G)$. On the other hand, transforming y_v into y_{vn} of Constraints (3.4) is equivalent to solving the minimum vertex colouring (MVC) problem over G. There is a well-known relation in graph theory stating that $\omega(G) \leq MVC(G)$ for any graph G [55]. Therefore, as shown in Figure 3.2, and from the previous graph relation, we may need more vessels when transforming a feasible solution s^{red} into a feasible solution s^{comp}.

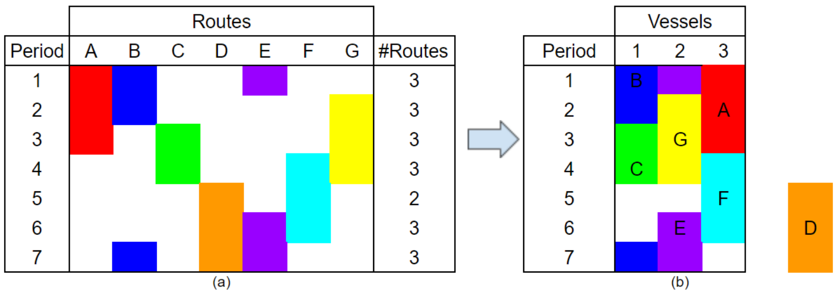
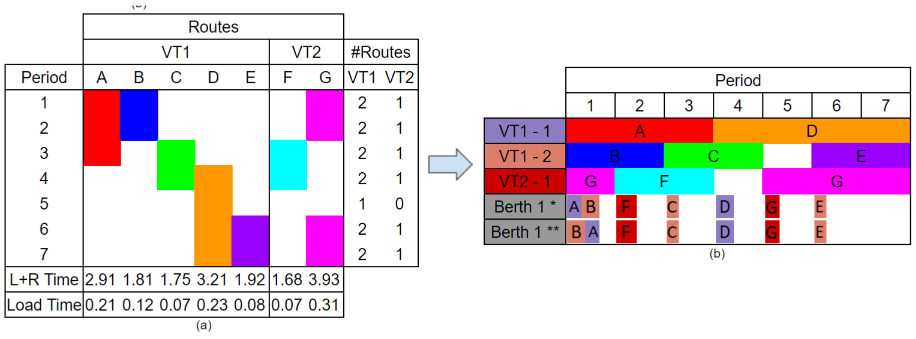


Figure 3.2: First cause of infeasibility: the impossibility of allocating routes to vessels.

The inconvenience is that solving the minimum vertex colouring problem over a generic graph G is NP-hard, since its decision version is NP-complete [56]. In KISIALIOU *et al.* [11], a voyage-based model based on flexible departures from the base and the possibility of coupling vessels by swapping their schedules was proposed. Their model was solved exactly on small- and medium-size instances. Generalizing these concepts, CRUZ *et al.* [2] introduced the integration of berth allocation decisions to the fleet composition and periodic routing problem, solving the largest available instances. Both considered a weekly planning horizon for the installations and a two-week planning horizon for the vessels.

Conversely, we decided to use a weekly cyclical horizon for both vessels and installations, and to solve the reduced problem within a branch-and-cut framework in such a way that whenever an incumbent solution s^{red} is infeasible only due to fleet allocation, as illustrated in Figure 3.2, a cut of type (3.33) based on s^{red}, for each vessel type v with problem allocating routes to vessel, with F_v being the correct fleet size (minimal colouration) for the vessel type v, is added to the reduced formulation aiming to approximate its solution space to that associated with the complete formulation. This approach works in our context since the colouring problems that we will have to solve are not very large.

The second cause of infeasibility occurs when an incumbent solution s^{red} for the reduced model cannot be transformed into a feasible solution for the complete model due to the impossibility of allocating vessels to berths. To illustrate this situation, Figure 3.3(a) shows a possible solution for the reduced formulation that is composed of one berth, five routes allocated to a vessel of type 1, and two routes allocated to a vessel of type 2. We can see that for the vessel of type 1, there are at most two simultaneously active routes per period, while for the vessel of type 2, this number is reduced to at most one active route per period. Therefore, Constraints (3.26) define a fleet of two vessels of type 1 and one vessel of type 2. In Figure 3.3, "VT1" and "VT2" denote vessel types; "L + R Time" represents the loading time plus the route time; and "Load time" indicates only the loading time. Figure 3.3(b) shows the only feasible vessel allocation to routes shown in Figure 3.3(a), represented by VT1 - 1, VT1 - 2 and VT2 - 1, and the two possibles berth allocations, indicated by * and **.



	Routes								
	VT1					VT2		#Routes	
Period	A	B	C	D	E	F	G	VT1	VT2
1	🟥	🟦					🟪	2	1
2	🟥	🟦					🟪	2	1
3	🟥		🟩			🟦		2	1
4			🟩	🟧		🟦		2	1
5				🟧				1	0
6				🟧	🟪		🟪	2	1
7				🟧	🟪		🟪	2	1
L+R Time	2.91	1.81	1.75	3.21	1.92	1.68	3.93		
Load Time	0.21	0.12	0.07	0.23	0.08	0.07	0.31		

(a)

Figure 3.3: Second cause of infeasibility: the impossibility of allocating vessels to berths.

As previously mentioned, when analyzing only the routes and vessel types, the fleet composed of two vessels of type 1 and one vessel of type 2 is sufficient to handle the five routes of Figure 3.3(a). For example, Routes A and D can be allocated to the first vessel of type 1, Routes B, C and E to the second vessel of type 1, and finally Routes F and G can be allocated to the vessel of type 2 as illustrated by Figure 3.3(b), VT1 - 1, VT1 - 2 and VT2 - 1.

However, since there is a single berth, an infeasibility is generated when considering the loading and route times of each route. For instance, if route A is the first to be executed, illustrated by Figure 3.3(b), Berth 1 *, the waiting time for loading route B will be 0.21, and this route will have a total duration of $0.21 + 1.81 = 2.02$. As a consequence, route B will be active in period 3, overlapping route C. This infeasibility also occurs if route B is the first to be executed illustrated by Figure 3.3(b), Berth 1 **. In this case, the waiting time for loading route A will be

0.12, and this route will have a total duration of $0.12 + 2.91 = 3.03$. Subsequently, route A will be active in period 4, overlapping route D. Since there is no other way to allocate the five routes to the two types of vessels and there is only one berth, it is impossible to transform the incumbent solution s^{red} of the reduced model into a feasible solution s^{comp} for the complete formulation. Nonetheless, note that if route C did not exist, it would be possible to allocate Routes A and E to a vessel of type 1 and routes B and D to another vessel of type 1, without generating overlaps.

3.3.3 Separation procedure

The separation procedure presented in Algorithm 6 generates Constraints (3.32) dynamically, in order to approximate the search space of the reduced model to that of the complete model. This procedure receives an incumbent solution s^{red} of the reduced model and then solves a minimum colouring problem for each type of vessel, as shown in Section 3.3.2; as a result, we obtain the necessary fleet to serve the routes of s^{red} (Line 1). If the fleet determined by the colouring problem is the same one used in s^{red}, the feasibility *SubProblem*, is solved (Line 2). If this problem is infeasible (Line 3), as in the example shown in Figure 3.3, then the solution s^{red} is reduced, according to Algorithm 7 (Line 4), and a cut (3.33) is added to the reduced problem (Line 5). Conversely, if the fleet determined by the colouring problem is different from that indicated by s^{red} (Line 6), as in the example shown in Figure 3.2, the s^{red} fleet is replaced with the one generated by the colouring problem (Line 7). If the feasibility *SubProblem* is infeasible (Line 8), as in the example shown in Figure 3.3, then the solution s^{red} is reduced, according to Algorithm 7 (Line 9), and a cut (3.32) is added to the reduced problem (Line 10). If the feasibility *SubProblem* is feasible, a cut (3.33) is added to the reduced problem (Line 12). Algorithm 7 is used in Lines 4 and 9 of the Algorithm 6. It tries to reduce the number of variables x_{rtv} from s^{red} (Line 2) which allows a stronger cut (3.32) for the reduced problem.

Algorithm 6: Separation procedure

```
Input: Incumbent $s^{red}$
1  $Fleet \leftarrow MinimalColouring(s^{red})$;
2  if  $Fleet = Fleet(s^{red})$ then
3      if $SubProblem(s^{red})$ is infeasible then
4          $s^{red} \leftarrow Reduce(s^{red})$;
5          Add cut (3.32) to (3.23)−(3.31) using $s^{red}$;
6  else
7      $Fleet(s^{red}) \leftarrow Fleet$;
8      if $SubProblem(s^{red})$ is infeasible then
9          $s^{red} \leftarrow Reduce(s^{red})$;
10         Add cut (3.32) to (3.23)−(3.31) using $s^{red}$;
11     else
12         Add cut (3.33) to (3.23)−(3.31) for each $v$ with conflicting minimal colouring,
             using $s^{red}$;
```

Algorithm 7: Reduce

```
Input: Solution $s^{red}$
Output: Minimal Solution $s^{red}$
1  forall $(r, t, v) \in S_x^{red}$ do
2      Remove $x_{rtv}$ from $s^{red}$;
3      if $SubProblem(s^{red})$ is feasible then
4          Insert $x_{rtv}$ into $s^{red}$;
5  return $s^{red}$;
```

3.4 Adaptive large neighborhood search heuristic

Introduced by ROPKE and PISINGER [57], ALNS extends the metaheuristic large neighborhood search (LNS) proposed by SHAW [58], which is based on the principle of destruction and reconstruction. At each iteration, ALNS applies an operator to *destroy* a solution s and *reconstruct* it in a different way, thus generating a new solution s'. This new solution is accepted according to the simulated annealing (SA) acceptance criterion [59]: if s' is better than s, the search continues from s', otherwise the search continues from s' with a given probability. What differentiates ALNS from LNS is that in LNS the destruction (removal) and construction (insertion) operators are chosen with the same probabilities, whereas in ALNS this selection is made according to an adaptive mechanism by which, at each iteration, the probability of selecting a method depends on its past performance by the adaptive layer.

The ALNS effectiveness was tested on various vehicle routing problems due to its ease of adaptation to many problems and yielded good results. This work couples the ALNS well tested routing decisions with multiple starts in order to explore a larger set of fleet distributions and spaced local searches to speed up the search.

3.4.1 Metaheuristic overview

Already implemented in some works (see [60–62]), the multi-start scheme works with three parameters which are the number of starts α, the number of iterations per start β, and the total number of iterations γ. Algorithm 8 presents the main procedure. It receives the general problem data, then generates α initial solutions and applies an ALNS (Algorithm 9) limited to β iterations. The best generated solution s^* over the α starts is then given to one more ALNS as an initial solution, with a limit of $\gamma - \alpha\beta$ iterations.

Algorithm 8: Multi-start metaheuristic

```
Input: Graph $G$, set of periods $T$, set of vessels $V$ and set of global parameters
Output: Best feasible solution found $s^*$
1 $S \leftarrow \emptyset$;
2 for $k = 1$ to $\alpha$ do
3     Build an initial solution $s$;
4     $s \leftarrow ALNS(s, \beta)$;
5     $S \leftarrow S \cup \{s\}$;
6 $s^* \leftarrow s_l \in S | \forall s \in S \rightarrow C(s_l) \leq C(s)$;
7 $s^* \leftarrow ALNS(s^*, \gamma - \alpha\beta)$;
8 return $s^*$.
```

Algorithm 9 presents the general framework of the ALNS implemented in this work. It receives an initial solution as the best known solution s_{best} and the maximum number of iterations N. It then initializes the weights for the insertion and removal operators, the current solution s_{curr}, the temperature θ, the best feasible solution found $s_{feasible}$, which is stored, and the most promising solution s_{prom} identified in the last δ iterations, the one to which the local searches are applied.

Lines 3−26 constitute the main loop. An auxiliary solution s_{aux} is first defined, and removal and insertion operators are applied. From Lines 7−11, the most promising solution s_{prom} is updated. If the current iteration is multiple of δ, a local search is applied to s_{prom}, the solution found is kept as s_{aux}, and s_{prom} is reset. Lines 12−13 update the best known feasible solution found $s_{feasible}$. Lines 14−26 describe the usual SA mechanism and the ALNS acceptance flow which includes updating the scores and the weights used in the selection of the insertion and removal operators when the current iteration is multiple of ρ. We use a penalized objective function with adjustable weights which are updated in Line 24. The related update process is described in the next section. At the end of the main loop, the best feasible solution found $s_{feasible}$ is returned.

Algorithm 9: ALNS(s_{best}, N)

```
Input: Initial Solution $s_{best}$ and maximum number of iterations $N$
Output: Best feasible solution found $s_{feasible}$
1  Initializes the weights for the insertion and removal operators;
2  $s_{curr} \leftarrow s_{best}$; $\theta \leftarrow C(s_{best})\theta'_0$; $C(s_{feasible}) \leftarrow \infty$; $C(s_{prom}) \leftarrow \infty$;
3  for $iteration = 1$ to $N$ do
4      $s_{aux} \leftarrow s_{curr}$;
5      Select a removal operator $o^-$ and apply it to $s_{aux}$;
6      Select a insertion operator $o^+$ and apply it to $s_{aux}$;
7      if $C(s_{aux}) \leq C(s_{prom})$ then
8          $s_{prom} \leftarrow s_{aux}$;
9      if iteration is multiple of $\delta$ then
10         $s_{aux} \leftarrow LocalSearch(s_{prom})$;
11         $C(s_{prom}) \leftarrow \infty$;
12     if $s_{aux}$ is feasible and $C(s_{aux}) \leq C(s_{feasible})$ then
13         $s_{feasible} \leftarrow s_{aux}$;
14     if $C(s_{aux}) \leq C(s_{best})$ then
15         $s_{best} \leftarrow s_{aux}$; $s_{curr} \leftarrow s_{aux}$;
16     else
17         if $C(s_{aux}) \leq C(s_{curr})$ then
18             $s_{curr} \leftarrow s_{aux}$;
19         else
20             if $Accept_{SA}(C(s_{aux}), C(s_{curr}), \theta)$ then
21                 $s_{curr} \leftarrow s_{aux}$;
22     Update scores $\pi_{o^-}$ and $\pi_{o^+}$;
23     if iteration is multiple of $\rho$ then
24         Update penalties' weights;
25         Update insertion and removal operators' weights;
26     $\theta \leftarrow \mu\theta$;
27 return $s_{feasible}$.
```

3.4.2 Search space and feasibility

The search space is computed as [(3.2)−(3.21)] and by two types of violation: (i) load violations associated with vessels exceeding their capacities; and (ii) time violations associated with vessels performing voyages concurrently in the same period (Constraints (3.4)), or berths occupying too much loading time from one period to the next (Constraints (3.8)) or exceeding the total loading time in the planning horizon (Constraints (3.7)).

Let O_v^t and E_v^q be the total overlapping time and exceeding load, respectively, for vessel v; let E_{ct}^t be the total loading time exceeded (greater than TOL) for berth $b \in N^B$ in period $t \in T$; and let E_b^{tt} be the total loading time exceeded in the planning horizon for berth $b \in N^B$. Each type of violation has an associated weight: $\omega^q > 0$ and $\omega^t > 0$ are the penalty weights for load and time violations, respectively. Hence, a cost for violations is computed as:

$$CV(s,\omega^q,\omega^t) = \omega^q \sum_{v \in V} E^q_v(s) + \omega^t(\sum_{v \in V} O^t_v(s) + \sum_{b \in N^B} (E^{tt}_b(s) + \sum_{t \in T} E^t_{ct}(s))). \tag{3.36}$$

Considering that the terms of Equation (3.36) are non-negative, solution s is feasible if and only if $CV(s,\omega^q,\omega^t) = 0$. Finally, the penalized objective function cost for a solution s is given by $C(s) = C'(s) + CV(s,\omega^q,\omega^t)$ where $C'(s)$ is given by objective function (3.1). The penalty weights have equal initial values ω_0 which are adjusted for each segment of ρ iterations. If the best solution for the ALNS (s_{best}) is infeasible for the load constraints, $\omega^q \leftarrow \tau^+\omega^q$ with $\tau^+ > 1$, otherwise $\omega^q \leftarrow \tau^-\omega^q$ with $\tau^- \leq 1$. This procedure is also used for the weight ω^t when s_{best} violates time constraints.

3.4.3 Initial solution

The general strategy is to diversify the initial fleet with poor routing sequences, so that the ALNS heuristic can iteratively reduce the fleet size and improve the routing. Every initial solution is generated in two phases called Fix and Split. The Fix phase assigns to each offshore unit $u \in N^{OI}$ a randomly selected delivery pattern $p \in P_u$. Thus, each period $t \in T$ has a list L_t of offshore units to be split into voyages made by the available vessels. In the Split phase, for each period $t \in T$ a vessel $v \in V$ is randomly selected and installations $u \in L_t$ are randomly chosen to be serviced by vessel v respecting its capacity. When the capacity of vessel v is reached, a berth is assigned to it. While list L_t is not empty, a new vessel is randomly selected and the process is reiterated. When all periods have been evaluated, the initial solution is obtained.

3.4.4 Removal and insertion operators

At each iteration of the ALNS heuristic, $n^- \in [n^-_1|N^{OI}|, n^-_2|N^{OI}|]$ offshore units are removed from s_{aux} by a removal operator o^- and reinserted by an insertion operator o^+. The parameters n^-_1 and n^-_2 define the range of the search around the current solution s_{curr} [57].

Our ALNS uses five removal and three insertion operators based on algorithms proposed by ROPKE and PISINGER [63], MATTOS RIBEIRO and LAPORTE [64] and KOÇ *et al.* [62]. We also tested the removal operators 'average cost per unit' from PARASKEVOPOULOS *et al.* [65], 'historical node-pair' and 'cluster removal' from PISINGER and ROPKE [66], as well as other SHAW [58] variants, but after extensive computational experiments, we realized that the best results were obtained with the five removal and the three insertion operators described as follows.

For the removal operators, we use three Shaw variants (RO1, RO2 and RO3), which are based on SHAW [58], a random removal (RO4) and a worst-removal procedure (RO5). The Shaw variants try to remove similar offshore units. The similarity between offshore units $u, v \in N^{OI}$ is given by Equation (3.37), where d_{uv} is the distance between the installations, D^{max} is the maximum distance between any two offshore units, q_u and q_v are demands of the offshore units, Q is the maximum demand found for all offshore units, f_u and f_v are the frequencies of the offshore units, and $|T|$ is the number of periods; ϕ_1, ϕ_2 and ϕ_3 are positive weights:

$$Shaw(u, v) = \phi_1 \frac{d_{uv}}{D} + \phi_2 \frac{|q_u - q_v|}{Q} + \phi_3 \frac{|f_u - f_v|}{|T|}. \tag{3.37}$$

Let L^- be the list of removed offshore units. Initially, the Shaw variants start by randomly selecting one offshore unit $u \in N^{OI}$ to be removed, thus $L^- \leftarrow \{u\}$. The remaining offshore units $v \in N^{OI} \setminus L^-$ are sorted in non-decreasing order according to Equation (3.37). Then, the operator selects a random number $a \in (0, 1]$ and the offshore unit $v \in N^{OI} \setminus L^-$ with index $\lceil a^p |N^{OI} \setminus L^-| \rceil$ in $N^{OI} \setminus L^-$, is removed. The parameter $p \geq 1$ must be tuned. Higher values of p increases the probabilities to select offshore units of $N^{OI} \setminus L^-$ with lower Shaw distances. The list of removed offshore units is updated: $L^- \leftarrow L^- + \{v\}$. This process is reiterated until the number of removed installations is reached (n^-). At each iteration, the initial offshore unit u is selected from L^-.

On this basis, with the ϕ values indicated by ROPKE and PISINGER [63] and KOÇ *et al.* [62], the following Shaw variants are:

- *Shaw removal (RO1)* uses $\phi_1 = 0.4$, $\phi_2 = 0.3$ and $\phi_3 = 0.3$;
- *Shaw neighbours removal (RO2)* uses $\phi_1 = 1$, $\phi_2 = 0$ and $\phi_3 = 0$; and
- *Shaw frequency removal (RO3)* uses $\phi_1 = 0.2$, $\phi_2 = 0$ and $\phi_3 = 0.8$.

The *Random removal (RO4)* operator removes randomly offshore units until reaches n^-. This operator generates a poor set of removed offshore units, but this helps diversify the search.

The *Worst removal (RO5)* operator also uses the removal strategy used in the Shaw operator. In this case, each non-removed offshore units v is assigned a cost which is the difference of the solution with and without it. All non-removed offshore units are sorted in non-decreasing order based on this cost and the removal operator used (that which uses $a \in (0, 1]$ and $p \geq 1$) and the Shaw operator is applied. This process is reiterated until n^- offshore units have been reached.

Conversely, every insertion operator o^+ starts with the set of removed offshore units L^- and inserts them back in solution s. The insertion operator $\Phi(u, s)$ inserts the offshore unit u into the best position of solution s. That is, $\Phi(u, s)$ finds the best position to insert u taking into account the incremental cost for solution s. All delivery patterns for u are analysed to define this best position. Thus, let $C(\Phi(u, s))$ be the incremental cost for solution s when u is inserted into the best position.

The first insertion operator, called *Deep greedy insertion (IO1)*, finds $u \in L^- | \forall v \in L^- \rightarrow C(\Phi(u, s)) \leq C(\Phi(v, s))$ and inserts it into the best position $\Phi(u, s)$. This process is reiterated until all removed offshore units have been inserted back. The second insertion operator, called *Greed insertion (IO2)*, randomly selects an offshore unit $u \in L^-$ and inserts it into the best position $\Phi(u, s)$. This process ends when all removed offshore units have been inserted back. Finally, the last insertion operator is the *k-regret insertion (IO3)* which is based on a k-regret criterion, as in ROPKE and PISINGER [63] and MATTOS RIBEIRO and LAPORTE [64]. Given a set of removed offshore units L^-, for each offshore units $u \in L^-$, it computes a regret value which is based on the delivery patterns available for u. For each offshore unit $u \in L^-$, the k-best insertion positions per period are obtained. Each position has a cost increment, and therefore a regret cost can be found for each period. Thus, for each delivery pattern available for u, we compute a regret cost for it which is obtained by adding the regrets generated for each period defined for the corresponding delivery pattern. We then choose the offshore unit $u \in L^-$ with the largest value of regret per delivery pattern to be inserted into solution s.

3.4.5 Local search

The local search mechanism was inspired by VIDAL *et al.* [16] and is described in Algorithm 10 which contains two main steps: voyage improvement (VI) and delivery pattern improvement (PI). These steps are sequentially applied on the current solution s, with the first one selected randomly. The algorithm ends when no improvement can be found for solution s. Step VI tries to improve the total distance travelled by all vessels per period by applying swap moves, and Step PI tries to find a better delivery pattern distribution for the offshore units.

Step VI first randomly selects a period $t \in T$. Two offshore units u and v, both serviced in period t, are then randomly chosen. Let $k_u, k_v \in \{0, 1, 2, 3\}$ be indicators of size. For each combination of k_u and k_v, the k_u installations serviced after u are swapped with the k_v installations serviced after v. If this move reduces the cost, Step VI ends and returns true. Otherwise, the swap is undone and a new combination is tested. When all combinations are evaluated, two new installations not yet selected are chosen and the combinations are tested again. If all installations have been

Algorithm 10: Local Search

```
   Input: Solution $s$
   Output: Improved solution $s$
1  $k_0 \leftarrow$ TRUE; $k_1 \leftarrow$ TRUE;
2  $b \leftarrow Rand(TRUE, FALSE)$;
3  repeat
4      if $b$ then
5          $k_0 \leftarrow PI(s)$;
6      else
7          $k_1 \leftarrow VI(s)$;
8      $b \leftarrow \neg b$;
9  until $k_0 \vee k_1$;
10 return $s$.
```

tested, a new period t not yet selected is randomly chosen and the entire process is repeated. Finally, if an improvement move is not found, Step VI returns false.

For Step PI, an installation u is randomly selected from N^{OI}. This installation is removed from s and reinserted according to position $\Phi(u, s)$, generating solution s'. If the cost of solution s' is better than that of solution s, Step PI returns true and ends. Otherwise, a new installation not yet evaluated is randomly selected from N^{OI}. If all installations have been evaluated and no improvement has been found, Step PI ends, returning false.

It is very important to consider berth allocation decisions in the periodic routing of offshore supply vessels since an infeasible allocation of a berth yields a high cost. The incorporation of berth allocation decisions in the periodic planning of offshore supply vessels makes the problem more difficult to solve by exact approaches, but this was not the case for our ALNS heuristic. Here, the berth allocation decisions were integrated to the insertion and removal processes. We tested some specific heuristics for swapping routes or reallocating berths, as in BORTHEN *et al.* [47] and KISIALIOU *et al.* [11], but they only increased the complexity of the heuristic with only a negligible improvement in solution quality.

3.4.6 Adaptive layer

Every iteration, the ALNS chooses a pair of insertion and removal operators through a roulette wheel mechanism. Initially, all operators have the same weight. The search is divided into segments of ρ iterations each. When a segment ends, the weights are updated based on the score obtained for the last segment, taking into account the number of times each operator has been used during that segment.

The score of an operator is increased by a parameter equal to σ_1, σ_2 or σ_3 when it identifies a new solution. If a pair of removal-insertion operator finds a new best solution, their scores are increased by σ_1, if it finds a solution better than the current one, their scores are increased by σ_2, and if it finds a non-improving solution which is accepted, their scores are increased by σ_3. Thus, the weight ϕ_i of operator i is updated by Equation (3.38), where π_i is the resulting score and ξ_i is the number of times operator i has been used in the last segment, where the parameter τ_A is called *reaction factor*:

$$\phi_i = (1 - \tau_A)\phi_i + \tau_A \frac{\pi_i}{\xi_i}. \tag{3.38}$$

3.4.7 Simulated annealing

We use the SA acceptance criterion, i.e., given a current solution s, a neighbor solution s' is accepted if it provides a better cost, and it is accepted with probability $\exp[(C(s') - C(s))/\theta]$ otherwise, where $\theta \geq 0$ is the current temperature and $C(s)$ is the penalized solution cost defined in Section 3.4.2. The temperature starts at θ_0 and is multiplied by a cooling rate μ at every iteration.

It is expected that θ will have values proportional to $C(s)$ throughout the search. The choice of the initial temperature θ_0 and the final temperature θ_F are therefore related to the cost of the initial solution $s_{initial}$ for a consistent acceptance criterion throughout the search process. Thus, the values of θ_0 and θ_F are chosen according to parameters θ'_0 and θ'_F, explicitly defined by Equations (3.39) and (3.40). This leads to the cooling rate μ which is dependent of the maximum number of iterations N, as expressed by Equation (3.41):

$$\theta_0 = C(s_{initial})\theta'_0 \tag{3.39}$$

$$\theta_F = C(s_{initial})\theta'_F \tag{3.40}$$

$$\mu = \sqrt[N-1]{\frac{\theta'_F}{\theta'_0}}. \tag{3.41}$$

The values of θ'_0 e θ'_F must be tuned. The results for the PSVPP-BA are shown in Section 3.5.2.

3.5 Computational experiments

This section presents the results of our computational experiments. From the E&P operators' perspective, or even from the perspective of offshore logistics providers, cost savings can be achieved through a better planning of the PSV fleet. This planning must take into consideration the berth capacity constraints in a 24/7 continuous operation, and the berth time dependents on the amount of cargo loaded rather than being a fixed time. These operational characteristics make the problem hard to solve for large instances.

The ALNS based heuristic and exact the methods was implemented in the C programming language, using the gcc 10.2 compiler with -O3 option. The computer used in all experiments was an AMD Threadripper 9 3960x 24c/48t with static clocks @ 4.0Ghz processor, 128GB DDR4 of RAM and 130GB of SSD reserved expanded memory, running Ubuntu 20.04 x64 operating system. The commercial solver used for the TSP solving, complete model, reduced model and subproblem was Gurobi 9.0.3. All the reported heuristic and TSP solving results are from single thread processing, and the complete and reduced models results are the total 24 thread processing time, and all computational times are expressed in seconds. Before presenting the results, we introduce the set of instances, and the details of the parameters tuning processing.

3.5.1 Set of instances

The fleet-sizing and periodic routing problem with berth allocation decisions tackled in this chapter is difficult to solve, due to its combinatorial nature and the size of the considered instances. These are the largest available benchmark instances for this problem. In order to achieve a good-quality solution or even an optimal solution, CRUZ *et al.* [2] solved the problem in steps, with a Multi Step Model Approach (MSMA), aiming to reduce the processing time. The instances solved in this chapter are real-based cases obtained from a Brazilian oil company, from its operations at Campos Basin. Four cases are presented: C10, C15, C41 and C79, having 10, 15, 41 and 79 offshore units, respectively. In cases C10 and C15, routes are generated considering all possible combinations for all units. In cases C41 and C79, CRUZ *et al.* [2] generated routes considering all possible combinations of the units belonging to their clusters (i.e., groups of offshore units), while we solved these instances without clustering, which makes them even harder to solve by mathematical programming algorithms.

The largest available real-world instance, C79 (CRUZ *et al.* [2]), is made up to 79 offshore units operating in the Campos Basin, while C10, C15 and C41 are cut-down versions of C79. Four berths are available at the onshore base and the maximum number of departures from each berth in each day is limited to two. The heterogeneous fleet is composed of three types of vessels: PSV4500 with deck capacity of 900 m^2, PSV3000 deck capacity of 600 m^2 and PSV1500 with deck capacity of 300 m^2. All vessels share approximately the same travel speed of 10 knots ($\approx$ 18.52 km/h).

CRUZ *et al.* [2] also defined segregated instances with an “S” suffix. Their motivation to segregate the instances came from the fact that the production installations usually remain fixed in the same position for many years, while the drilling rigs and maintenance platforms are constantly moved from one oilfield to another. In this work, there is no need to consider these instances, since they would be the same as those without the suffix. The authors also tested C41 and C79 with different numbers of berths. Therefore, in order to standardize the tests, each instance is named with the number of offshore units, weekly visits and available berths: for example C79-112-5 instance has 79 offshore units to be served, 112 weekly visits and five available berths.

For more extensive testing, we created instances C21-39-1, C30-48-2, C50-70-3, C60-89-3 and C69-99-4 with the following procedure: starting with instance C79-112-4, randomly select and remove offshore units until the targeted number is reached, than each offshore unit has its coordinates randomized within the boundary box defined by the offshore units presented in instance C79-112-4. In order to evaluate the isolated impact of clustering, for each new instance and for instance C15-1-1, we created a manual cluster arrangement based on location of the offshore units. The instances with 41 and 79 offshore units already had cluster arrangements from CRUZ *et al.* [2] and they were kept. Finally, two extra instances, C66-94-4 and C40-62-2, were created just for the heuristic parameter tuning, applying the same method used to create the new instances.

Table 3.1 shows, for each test instance and cluster configuration, the total number of feasible routes and the total CPU time to solve all respective Traveling Salesman Problems. The CPU time ranges from 0.05 seconds and approximately 3.7 hours. It is important to highlight that these times are included in the computational experiments performed for the exact methods.

3.5.2 Parameters tuning

All parameters, except for α, β and γ, were calibrated through single executions of the ALNS heuristic. The initial parameters values were estimated based on related articles, such as KISIALIOU *et al.* [52] and KOÇ *et al.* [62], or simply by trial and error.

As mentioned in Section 1.2, the fixed costs are much higher than the variable costs for the PSVPP-BA. Therefore, in order to attenuate the influence of the variance of the initial random solutions on the calibration process, a random seed and an initial solution were fixed in each run (within a total of 30 fixed initial solutions), so that the difference between the rounds was only due the parameters to be calibrated.

The tuning of the multi-start parameters α, β and γ was based on the same principle with a sequence of 15 random seeds and a sequence of 100 initial solutions

Table 3.1: Number of offshore units, cluster arrangements and feasible non dominated routes.

#Offshore Units	#Clusters	#Routes	CPU(s)
10	1	912	0.22
15	1	10,021	2.79
15	2	370	0.05
21	1	28,929	8.79
21	2	1,103	0.18
30	1	91,689	20.69
30	4	581	0.08
41	1	782,289	209.55
41	5	1,373	0.22
41	7	438	0.06
50	1	1,298,168	373.71
50	7	591	0.07
60	1	4,071,416	1529.14
60	8	1,178	0.15
69	1	14,930,022	6402.72
69	8	1,169	0.16
79	1	29,013,740	13317.54
79	9	2,168	0.32
79	12	1,107	0.11

were fixed for each tuning run, having a total of $100 \times 15 = 1,500$ initial solutions, per instance. In this way, for the same values of α and β, a larger γ implies the same best solution after the α starts, but a better final solution is not guaranteed since the temperature values per iteration in the final ALNS execution are different.

The tuning of the ALNS' parameters occurred in three different ways: in isolation (ρ, p, δ, and τ_A), and in triplets (($\sigma_1, \sigma_2, \sigma_3$), and ($\alpha, \beta, \gamma$)), according to the types of correlations between them. The results were normalized based on the initial parameter values, and the average CPU times were omitted when the variation of the parameters did not change the CPU times by more than 5%.

As a result of the tuning process, Table 3.2 presents the initial values, the tested values and the best achieved values for each parameter of the ALNS based heuristic developed in this work. The details about the tunning process are reported in the Appendix.

3.5.3 Computational results and comparative analysis

Considering the parameters presented in Table 3.2, the heuristic was executed 15 times, with random seeds and the exact methods were executed once per instance. Table 3.3 presents, for each instance, the results of the branch-and-cut algorithm described in Section 3.3 to the reduced model and the results of the complete formulation, including the value of the best incumbent solution found (UB), the lower bound (LB) and CPU time (CPU, in seconds). It also shows the results of the ALNS comprising the best solution cost (Best), the average solution cost (Average), the coefficient of variation (CV) in percentage based on average, and the average

Table 3.2: Parameters settings.

Parameter	Description	Initial value	Tested values	Best value
τ^-	Infeasible update weight	1.15	1.05, 1.10, 1.15, 1.25	1.15
τ^+	Feasible update weight	0.9	0.6, 0.75, 0.9, 1.00	0.9
n_1^-	Lower removal rate	0.1	0.05, 0.10, 0.15, 0.20	0.1
n_2^-	Upper removal rate	0.3	0.20, 0.25, 0.30, 0.40	0.25
θ_0'	Initial relative temperature	1/3	1, 1/3, 1/10, 1/50	1/50
θ_F'	Final relative temperature	1/300	1/50, 1/100, 1/300, 1/500	1/500
ρ	Segment size	50	25, 50, 100, 200, 300	50
p	Removal's Level of determinism	4	1, 2, 4, 6, 10	4
δ	Local search interval	50	10, 25, 50, 200, 500	50
τ_A	Adeptness rate	0.1	0.01, 0.05, 0.1, 0.2, 0.3	0.05
σ_1	Best score	10	0, 5, 10	5
σ_2	Improvement score	5	0, 5, 10	0
σ_3	Worst score	0	0, 5, 10	10
γ	Total number of iterations	50,000	(15, 50, 150, 300)x10^3	300,000
α	Number of starts	-	1, 15, 35, 50, 75, 100	50
β	Iterations per start	-	100, 300, 500, 1000	100

CPU time (CPU, in seconds). A "-" value means that no result was found. Table 3.4 similarly presents the results for the exact methods with the different cluster arrangements for each instance in order to further analyse the performance of both exact methods and compare the results with the non-clustered instances, evaluating the impact of clustering.

Analyzing the results for the exact methods for Tables 3.3 and 3.4, it is important to highlight that either an optimal solution was found or we stopped due to lack of memory or we stopped due to a time limit of 86400 seconds. The branch-and-cut (B&C) method applied to the reduced model showed a clear advantage over the complete formulation, providing better bounds, with significantly reduced CPU times.

The ALNS, when compared to the exact methods, presented the best performance, both in terms of solution quality and computational time. The percentile coefficient of variation values demonstrate the robustness of the ALNS heuristic, remaining below 1% for the smaller instances, and between 0.5% and up to 2% for the larger instances, which correspond to the real-world cases derived from the Brazilian oil industry. The CPU times scaled well between the instances, with a maximum time of 583.3 seconds for the largest instance, which is quite reasonable for a difficult problem such as the PSVPP-BA. Finally, Table 3.3 shows that the increase in the number of berths per group of instances does not have a measurable impact on the CPU times.

Table 3.5 shows the comparison between the MSMA [2] and the ALNS results for the C41 group of instances. Columns 2−4 and 5−6 show the results for the two-berth instances and the three-berth instances, respectively. For the two-berth scenario, MSMA found a solution of total cost 261.99, and the ALNS was able to find a better total cost solution of 256.23, with both lower fleet and routing costs. In the three-berth scenario, MSMA found a solution of total cost 256.59, and the

Table 3.3: Heuristic and exact results for all instances.

	Complete Model			Branch and Cut			ALNS			
Instance	UB	LB	CPU(s)	UB	LB	CPU(s)	Best	Average	CV(%)	CPU(s)
C10-15-1	53.77	53.77	85.6	53.77	53.77	1.2	53.77	53.77	0.00	13.4
C15-24-1	80.37	77.11	86400.0	80.37	80.37	82.5	80.37	80.37	0.00	21.6
C21-39-1	202.55	147.9	86400.0	176.33	151.08	86400.0	160.75	160.95	0.08	48.4
C30-48-2	-	205.56	18873.5	221.84	211.68	86400.0	221.84	225.05	0.54	99.2
C41-59-2	-	233.59	5100.4	259.91	234.54	86400.0	256.23	257.22	0.91	150.8
C41-59-3	-	233.59	5042.4	252.64	233.95	86400.0	252.01	254.88	0.29	149.4
C50-70-3	-	-	623.4	385.64	343.68	86400.0	356.97	359.47	0.98	220.6
C60-89-3	-	-	1730.7	482.40	464.00	57981.2	464.19	470.38	1.27	291.6
C69-99-4	-	-	6612.6	-	455.94	26540.7	480.84	495.84	1.01	351.7
C79-112-4	-	-	13417.7	-	-	14562.3	559.59	578.45	1.99	583.3
C79-112-5	-	-	13356.9	-	-	14562.4	554.75	562.76	0.75	549.0
C79-112-6	-	-	13303.4	-	-	14562.9	554.89	557.21	0.45	547.7

Table 3.4: Results with clustered instances and exact methods.

		Complete Model			Branch and Cut		
Instance	#Clusters	UB	LB	CPU(s)	UB	LB	CPU(s)
C15-24-1	2	90.18	90.18	2785.4	90.18	90.18	10.2
C21-39-1	2	177.16	165.15	86400.0	177.14	170.17	86400.0
C30-48-2	4	246.68	222.14	86400.0	226.28	226.28	185.4
C41-59-2	5	-	252.10	86400.0	259.56	259.56	77400.1
C41-59-3	5	302.02	246.79	86400.0	259.52	259.52	1750.8
C41-59-2	7	262.09	261.84	86400.0	262.09	262.09	33500.7
C41-59-3	7	262.09	261.88	86400.0	262.09	262.09	1065.4
C50-70-3	7	392.82	366.44	86400.0	375.07	375.07	57488.6
C60-89-3	8	-	448.90	86400.0	483.03	470.14	86400.0
C69-99-4	8	-	460.58	86400.0	492.43	492.43	65781.6
C79-112-4	9	-	536.16	86400.0	583.78	559.42	86400.0
C79-112-5	9	-	536.16	86400.0	566.85	557.56	86400.0
C79-112-6	9	-	536.16	86400.0	566.85	557.56	86400.0
C79-112-4	12	-	547.42	86400.0	577.22	574.78	86400.0
C79-112-5	12	-	548.10	86400.0	577.22	572.74	86400.0
C79-112-6	12	-	548.10	86400.0	577.22	572.74	86400.0

ALNS was able to find a better total cost solution of 252.01, with lower fleet cost and higher routing costs.

Table 3.6 provides a comparison between MSMA and the ALNS results for the C79 group of instances. Columns 2−4, 5−6 and 7−8 show the results for the four-berth instances, the five-berth instances, and the six-berth instances, respectively. The four-berth scenario is the most restrictive and closest to the real case faced by the Brazilian oil industry. The MSMA found a solution of total cost 570.37 for the C79S(4B) instance, and the ALNS was able to find a better total cost solution of 559.59, with both lower fleet and routing costs, although concentrating the fleet of vessels only on the PSV4500 type, with deck capacity of 900 m^2. For the five-berth scenario, MSMA was not able to find an optimal solution, but found an incumbent solution of total cost 591.55 and 5.58% of optimality gap for the C79(5B) instance, the same solution found for the C79(4B) four-berth instance. The ALNS was able to find a better total cost solution of cost 554.75 for the C79-5 instance, with both lower fleet and higher routing costs, although again the fleet concentrates mainly on the

Table 3.5: Comparison between MSMA [2] and ALNS for the C41 group of instances.

C41	MSMA	MSMA	ALNS	MSMA	ALNS
Instance	C41S(2B)	C41(2B)	C41-2	C41(3B)	C41-3
Clusters	7	5	1	5	1
Total Cost	261.99	270.20	256.23	256.59	252.01
Fleet Cost	248	256	243	243	238
Routing Cost	13.99	14.20	13.23	13.59	14.01
PSV4500	5	5	4	4	3
PSV3000	1	2	2	2	3
PSV1500	1	0	1	1	1
Routes	17	19	17	19	19
CPU(s)	322	99,000	150.8	487	149.4

PSV4500 vessel. Finally, in the six-berth scenario, MSMA found a solution of total cost 558.55, and the ALNS was able to find a better total cost solution of 554.89, with lower routing costs, and the same fleet cost, although mainly concentrated on the PSV4500 vessel. Once again, it is important to emphasize that we are not considering clustered installations.

Table 3.6: Comparison between MSMA [2] and ALNS for the C79 group of instances.

C79	MSMA	MSMA	ALNS	MSMA	ALNS	MSMA	ALNS
Name	C79S(4B)	C79(4B)	C79-4	C79(5B)	C79-5	C79(6B)	C79-6
Clusters	12	9	1	9	1	9	1
Total Cost	570.37	591.55	559.59	591.55	554.75	558.55	554.89
Fleet Cost	540	560	532	560	527	527	527
Routing Cost	30.37	31.55	27.59	27.75	28.49	31.55	27.89
PSV4500	9	8	14	8	13	8	13
PSV3000	6	7	0	7	1	6	1
PSV1500	0	1	0	1	0	1	0
Routes	34	38	28	38	29	38	29
CPU(s)	259,180	144,299	583.3	144,299	549.0	109,585	547.7

To conclude, Tables 3.5 and 3.6 show that the proposed ALNS-based heuristic was able to find better solutions than those of CRUZ *et al.* [2] with both a smaller fleet and smaller routing costs, with well balanced fleets for the C41 instances and fleets concentrated mainly on the PSV4500 vessel for the C79 instances. Lastly, the good heuristic results were uphold by the good exact results of the branch-and-cut method applied to the reduced model.

3.6 Final remarks of the chapter

We have presented an exact branch-and-cut method and an adaptive large neighborhood search (ALNS) heuristic with multiple starts and spaced local searches to solve the periodic supply vessel planning problem (PSVPP) arising in the upstream offshore petroleum logistics chain. The PSVPP tackled in this work consists of a periodic vehicle routing problem while simultaneously determining the optimal fleet size and a mix of heterogeneous offshore supply vessels, their one week routes and schedules for servicing the offshore oil and gas installations, besides the berth allocations at the supply base.

We have extended the previous works of HALVORSEN-WEARE and FAGERHOLT [12], KISIALIOU *et al.* [11] and CRUZ *et al.* [2] since we used a replicable one-week planning horizon both for the offshore units and the vessels. We solved the largest available real-world instances, without dividing them into clusters, and we achieved good solutions relatively fast, performing significantly better than the branch-and-cut algorithm.

Chapter 4

Metaheuristics with variable diversity control and neighborhood search for the Heterogeneous Site-Dependent Multi-depot Multi-trip Periodic Vehicle Routing Problem

This chapter approaches the heterogeneous site-dependent multi-depot multi-trip periodic vehicle routing problem (HSDMDMTPVRP). Given a real world demand, we propose a metaheuristic named Adaptive Variable Neighborhood Race (AVNR) which combines variable neighborhood search and adaptive mechanisms integrated with a shrinking population managed with a diversity mechanism. We also adapted and implemented the metaheuristic Unified Hybrid Genetic Search (UHGS) [16] for the computational experiments. Our results show that AVNR finds good results for several instances.

A scientific paper containing these results is under production for submission to a high quality journal.

4.1 Literature Review

The VRP addressed in this chapter can be seen as a combination of the following problems: Periodic Vehicle Routing Problem (PVRP); Site-dependent Vehicle Routing Problem (SDVRP); Multi-depot Vehicle Routing Problem (MDVRP); Fleet Sizing Heterogeneous Vehicle Routing Problem (HVRP); and Multi-trip Vehicle Routing Problem (MTVRP).

The literature about VRP started with DANTZIG and RAMSER [13]. Based on a real case study, the problem consists of sizing a homogeneous fleet, constrained by the vehicles' capacity, and minimizing the total traveled distance. This VRP class is usually called Capacitated Vehicle Routing Problem (CVRP).

The periodic vehicle routing problem (PVRP) is an VRP extension in which vehicle routes are spread over several periods, and each client has a service frequency and delivery pattern constraints. Introduced in BELTRAMI and BODIN [67], with many real world applications, the problem was studied in many scientific works as surveyed in CAMPBELL and WILSON [68].

The site-dependent vehicle routing problem (SDVRP) can be seen as an particular case of the PVRP (except for the heterogeneous fleet), since both problems can be characterized as multilevel routing problems: at the first level, an allowable vehicle type (or pattern of delivery days) is selected for each customer and at the second level, a capacity and time constrained VRP is solved for each type of vehicle (or day of the period). Taking advantage of this characteristic, CORDEAU and LAPORTE [69] solved the SDVRP applying the same algorithm used to solve the PVRP [70]. Thus, the MDVRP can be seen as a particular case of the PVRP as well [16], differentiated by each level having a different starting depot.

The heterogeneous vehicle routing problem (HVRP) was introduced at GOLDEN *et al.* [71] and its most generic version consists of defining routes for a fleet of vehicles with different capacities, fixed and variable costs to serve a set of clients. KOÇ *et al.* [15] present a good review of this problem.

The vehicle routing problem with multiple use of vehicles (VRPM) [72] or, more commonly called, the multi-trip vehicle routing problem (MTVRP) [73], is characterized by the fact that vehicles are driven by drivers who have working hours established by law and, as such, can carry out more than one trip per day. In practice, multiple travel schedules are important because they reduce costs as it reduces the number of vehicles used and best employs the time of the drivers.

When analyzing the literature, to the best of our knowledge, there are no works that address all these VRPs together. The two closest works found in the literature are COELHO *et al.* [74], in which the VRP with heterogeneous fleet and multiple trips is studied but with no periodicity, and ALONSO *et al.* [75], in which the problem is called Site-Dependent Multi-Trip Periodic Vehicle Routing Problem (SDMTPVRP) which is periodic, site dependent, fixed heterogeneous fleet and multi trips but no sizing and no multi-depot. As in the work of of ALONSO *et al.* [75], in this chapter we tested the AVNR against the best algorithms in the literature for each VRP variant.

4.2 Mathematical Description

Let $G = \{N, E\}$ be a graph, where N is the set nodes and E is the set of edges. The nodes are partitioned in the set of depots, represented by D, and the set of clients, represented by $U = N \backslash D$. To each client $i \in U$ is associated a demand q_i, a service time ϕ_i, a set of delivery patterns P_i, and it has to be served f_i times (frequency). So, all clients have a total number of services $n_{ts} = \sum_{i \in U} f_i$.

The travel paths between nodes are represented by the set of edges $E = \{(i, j) | \forall i, j \in N, i \neq j\}$. An edge $(i, j) \in E$, between nodes i and j has a travel distance d_{ij} and a travel time t_{ij}. For any edge $e = (i, j) \in E$, considers that $e_o = i$ and $e_d = j$. When comparing edges, given edges a and b, $a = b$ means $a_o = b_o \wedge a_o = b_o$, $a < b$ means $a_o < b_o \vee (a_o = b_o \wedge a_d < b_d)$, otherwise $b < a$.

The set of available vehicle types is represented by K. Each vehicle type $k \in K$ has the fixed cost per use c_k^f, a cost per traveled distance c_k^v, maximum capacity M_k^l, maximum operation time M_k^d, maximum amount of trips per period M_k^t, maximum amount of vehicles available per period M_k^r and a time coefficient M_k^{d2t} (inverse of the speed) which is used to compute the time of the routes. Also, $K_i \subseteq K$ is the set of vehicle types that can visit node $i \in N$ and $N_k \subseteq N$ is the set of nodes that can be visited by vehicle type $k \in K$.

4.2.1 Solution representation

A solution s is a set of partial solutions s_{tkd}^p for each combination of period $t \in T$, vehicle type $k \in K$ and depot $d \in D$. A partial solution s_{tkd}^p is a set of routes r where each route r is a set of trips l and each trip l is as set of clients given the sequence used to serve them. Figure 4.1 presents an example illustrating all these elements. Each route r has a total distance $Dist(r, d)$ and a total time $Time(r, k, d)$ which is defined according to the vehicle type k. Each trip l has a total load $Load(l)$ that considers the served clients. If the maximum amount of routes M_k^r is not reached, there is an empty (available) route with an empty trip. For problems without a trip limit M_k^t, for each route, there is always an empty trip.

A solution s is valid if each client $i \in U$ is served with a delivery pattern $p(s, i) \in P_i$. For each partial solution s_{tkd}^p, there is a set of all served clients called Giant Tour $gt(s_{tkd}^p)$ that is defined as the union of its subsets, keeping each sequence of clients served without trip delimiters, , see example shown in Figure 4.2. For each solution s, there is a sorted multiset of edges $e(s)$ in increasing order, since the problem has symmetrical routes (any route can be taken in reverse for the same cost). For all trips in s, each time a vehicle goes from node $i \in N$ to node $j \in N$, if $i < j \rightarrow (i, j) \in e(s)$, otherwise $(j, i) \in e(s)$. We use a multiset of edges because it can contain repeated edges over the periods, as shown in Figure 4.2.

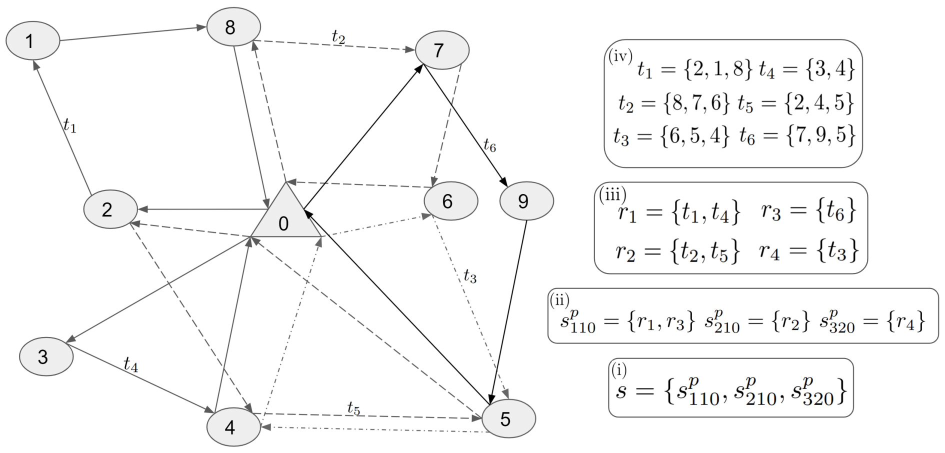

1
8
7
6
9
2
0
3
4
5
t_1
t_2
t_3
t_4
t_5
t_6
(iv) $t_1 = \{2,1,8\}$ $t_4 = \{3,4\}$
$t_2 = \{8,7,6\}$ $t_5 = \{2,4,5\}$
$t_3 = \{6,5,4\}$ $t_6 = \{7,9,5\}$
(iii) $r_1 = \{t_1, t_4\}$ $r_3 = \{t_6\}$
$r_2 = \{t_2, t_5\}$ $r_4 = \{t_3\}$
(ii) $s^p_{110} = \{r_1, r_3\}$ $s^p_{210} = \{r_2\}$ $s^p_{320} = \{r_4\}$
(i) $s = \{s^p_{110}, s^p_{210}, s^p_{320}\}$

Figure 4.1: Example of problem with three periods, two vehicle types and one depot. Solution s (i) has three partial solutions (ii), four routes (iii) and six trips (iv).

$$\boxed{\begin{aligned} gt(s^p_{110}) &= \{2,1,8,3,4,7,9,5\} \\ gt(s^p_{210}) &= \{8,7,6,2,4,5\} \\ gt(s^p_{320}) &= \{6,5,4\} \end{aligned} \quad \text{(v)}}$$

$$\boxed{\begin{aligned} e(s) = \{&(0,2),(0,2),(0,3),(0,4),(0,4),(0,5), \\ &(0,5),(0,6),(0,6),(0,7),(0,8),(0,8), \\ &(1,2),(1,8),(2,4),(3,4),(4,5),(4,5), \\ \text{(vi)} \quad &(5,6),(5,9),(6,7),(7,8),(7,9)\} \end{aligned}}$$

Figure 4.2: Example of all giant tours $gt(s^p_{tkd})$ (v) and the sorted multiset of edges $e(s)$ (vi) for the example of Figure 4.1.

4.3 Search mechanisms

The two metaheuristics proposed in this Chapter have many search mechanisms in common. The first one is the current state of the art for VRP presented in VIDAL *et al.* [16] and generalized for a wide range of VRPs in VIDAL *et al.* [76], called Unified Hybrid Genetic Search (UHGS). The UHGS includes advanced diversity control, feasibility control and a restart mechanism which were adapted here to solve the HSDMDMTPVRP and its VRP sub-problems (CVRP, SDVRP, MTVRP, HVRP, PVRP, MDVRP and SDMTPVRP).

The second one is original, was named Adaptive Variable Neighborhood Race (AVNR) and shares many mechanisms with UHGS. It considers multiple solutions in a race. At each step of the race, an Iterated oscillating search (IOS) algorithm is applied to each solution resulting in a new one, and the resulting pool of solutions is reduced to the next step given a survival selection process. With a smaller number of solutions, the next step will have more time to explore each solution. The main difference between the two proposed heuristics is the fact that UHGS crosses solutions in order to generate new ones and AVNR finds new solutions to be explored based on the current ones.

4.3.1 Feasibility and search space

The search space is the one described at Section 4.2 and by two types of violation: (i) load violations associated with vehicles exceeding their M_k^l; and (ii) time violations associated with vehicles performing trips exceeding their M_k^d.

Let EL_{km} be the total exceeding load, for vehicle type k and sequence of clients m and let ET_{krd} be the total exceeding time for vehicle type k, route r and depot d.

Each violation has an associated penalty weight $\omega^l > 0$ (load violation) and $\omega^t > 0$ (time violation). Hence, the costs for violations given a partial solution s^p are computed as:

$$\Omega^p(s_{tkd}^p) = \sum_{r \in s_{tkd}^p} (\omega^l \sum_{m \in r} EL_{km} + \omega^t ET_{krd}) \tag{4.1}$$

$$pCost(s_{tkd}^p) = \sum_{r \in s_{tkd}^p} (c_k^f + c_k^v \sum_{m \in r} Dist(m,d)) + \Omega^p(s^{p_{tkd}}) \tag{4.2}$$

Considering that the terms of Equation (4.1) are non-negative, a solution s is feasible if and only if $\Omega(s) = 0$ (Equation (4.3)). Finally, the penalized objective function cost for a solution s is given by Equation (4.4).

The initial values for the penalty weights are instance-dependent. Thus, $\omega^l = \frac{c_x^f + c_x^v D^{max}}{Q}$ and $\omega^l = \frac{c_x^f + c_x^v D^{max}}{M_x^{d2t} D^{max}}$, where x is the largest vehicle type available, D^{max} is the maximum distance between two clients, and Q is the maximum demand difference between two clients.

The penalty weights are dynamically adjusted by each metaheuristic by the penalty weight adjustments ω^+ and ω^-. For increasing the penalties, $\omega^+ > 1$, otherwise, $\omega^- \leq 1$.

$$\Omega(s) = \sum_{s_{tkd}^p \in s} \Omega^p(s_{tkd}^p) \tag{4.3}$$

$$C(s) = \sum_{s_{tkd}^p \in s} pCost(s_{tkd}^p) \tag{4.4}$$

For both algorithms, the penalty weights adjustments occur every ρ iterations. For the UHGS heuristic, each solution obtained after an education process, where a local search is applied (details in Section 4.3.12), can be feasible or not. If the proportion of infeasible solutions by load or time is out of the target proportion range $\xi^{Inf} \pm 0.05$, the respective penalty weight is updated. For example, if the proportion of infeasible solutions by load is higher, it is increased with $\omega^l \leftarrow \omega^+ \omega^l$, and if lower, it is decreased with $\omega^l \leftarrow \omega^- \omega^l$. The same process is applied to penalty weight ω^t.

For the AVNR heuristic, the update is based on a solution s and a violation tolerance ξ^v. For a solution s, we compute the violation levels $\Phi^l(s)$ for load (Equation (4.5)) and $\Phi^t(s)$ for time (Equation (4.6)). If s is feasible ($\Omega(s) = 0$), both penalty weights are decreased with ω^-, otherwise, they are updated according to their violation levels. For example, if $\Phi^l(s) > \xi^v$, the penalty weight for load is increased: $\omega^l \leftarrow \omega^+\omega^l$. This process is also applied to violation evolving time (ω^t).

$$\Phi^t(s) = \max_{s^p_{tkd} \in s, r \in s^p_{tkd}} \left\{ \frac{ET_{krd}}{M^t_k} \right\} \tag{4.5}$$

$$\Phi^l(s) = \max_{s^p_{tkd} \in s, r \in s^p_{tkd}} \left\{ \frac{\max_{m \in r} EL_{km}}{M^l_k} \right\} \tag{4.6}$$

4.3.2 Multi-trip split algorithm

The proposed metaheuristics use a multi-trip split algorithm during the search process, based on the algorithm presented in PRINS [77]. The procedure proposed by PRINS [77] receives a sequence of clients gt called a “giant tour” (see Section 4.2.1), a vehicle k and a depot d and returns a partial solution s^p_{tkd} serving all client in gt, with vehicle k, using depot d, by solving a minimal path problem, for an auxiliary directional acyclic graph.

In this work, we propose a multi-trip split algorithm that also solves a minimal path problem but with auxiliary directional acyclic multigraph. The result of this algorithm is still optimum for single trip problems but optimally is not ensured for multi-trip problems.

Algorithm 11 uses an auxiliary array of partial solutions S^p with one partial solution S^p_i for each item of gt. It starts by extending gt and performing the initialization of S^p (Lines 1 $-$3), and then the main loop (Lines 4$-$12) is started. The main idea is the same as presented by PRINS [77]. Each partial solution S^p_i is a set of routes servicing all clients from gt_1 until gt_i. When a new partial solution s' also servicing all clients from gt_1 until gt_i is found, S^p_i is updated.

Each step of the main loop has a reference index i with fixed partial solution S^p_i, a target index j, and a sequence $Trip$ which contains the clients from gt_{i+1} until gt_j. At each iteration of the inner loop (Lines 8$-$12), the algorithm asks if, by adding $Trip$ to S^p_i, there is a partial solution s' (Lines 8 and 11) with lower cost then the current S^p_j, either as a new route (Lines 8$-$9) or a new trip (Lines 10$-$12). If so, the partial solution S^p_j is updated. At the end, the best split of routes with the sequence of clients found S^p_n is returned.

Algorithm 11: Multi Trip Split

```
Input: Sequence $gt$ with $n$ clients, vehicle type $k \in K$ and depot $d \in D$
Output: Partial solution $S^p_n$
1  $gt \leftarrow \{0\} \cup gt$;
2  for $i = 1$ to $n$ do $pCost(S^p_i) \leftarrow +\infty$ ;
3  $S^p_0 \leftarrow \{\}$; $pCost(S^p_0) \leftarrow 0$;
4  for $i = 0$ to $n - 1$ do
5      $Trip \leftarrow \{\}$;
6      for $j = i + 1$ to $n$ do
7          $Trip \leftarrow Trip \cup \{gt_j\}$;
8          $s' \leftarrow S^p_i \cup \{\{Trip\}\}$;
9          if $pCost(s') < pCost(S^p_j)$ then $S^p_j \leftarrow s'$ ;
10         foreach $r \in S^p_i$ do
11             $s' \leftarrow (S^p_i \setminus r) \cup (r \cup \{Trip\})$;
12             if $pCost(s') < pCost(S^p_j)$ then $S^p_j \leftarrow s'$ ;
13 return $S^p_n$;
```

4.3.3 Neighborhood

The neighborhood η is a set of candidate insertion points computed based on a given solution s and period t to a client i. A candidate insertion point is a unique identifier in which a client i can be inserted after it.

The neighborhood has to be carefully chosen since if too small, the heuristic may not be able to find the best solutions given a search space too narrow. If the neighborhood is too large, the heuristic's performance can suffer with too many unnecessary search candidates. So, we try to balance the size of the neighborhood by using two parameters: the neighborhood threshold ξ^η and the minimum neighborhood size η^{rmin}. Let η^ξ_i be the number of clients from $U \setminus \{i\}$ with distance from i smaller than $\xi^D D^{max}$, so the neighborhood size η^{max}_i is the largest value between η^ξ_i and η^{rmin}.

Given a client i, a solution s and period t, Algorithm 12 starts with an empty neighborhood η (Line 1) and defines V which is a set with all other clients ($U \setminus \{i\}$) sorted by increasing distance from i (Line 2). Each client j can be added to η if it is served at period t and the vehicle that serves client j can also serves client i at the period (Lines 5).

The first loop (Lines 3–5) tries to add the first η^{max}_i clients from V to η. After that, for each vehicle type $k \in K$, that can serve i, and for each depot $d \in D$, if the depot is close enough to i ($Dist(d, i) < \xi^D D^{max}$), all trips (empty or not) will be added to η (Lines 6–8). Finally, the neighborhood η is returned and the algorithm ends (Line 9).

Algorithm 12: Neighborhood

```
Input: Solution $s$, period $t$ and client $i$
Output: Neighborhood $\eta$ of client $i$, for solution $s$ at period $t$
1 $\eta \leftarrow \{\}$; $\eta^{max} \leftarrow Min(\eta_i^{max} + n^-(s), |U| - 1)$;
2 $V \leftarrow U \setminus \{i\} \mid \forall V_j, V_k \in V, j < k \rightarrow Dist(i, V_j) < Dist(i, V_k)$;
3 for $i = 1$ to $\eta^{max}$ do
4     $j \leftarrow V_i$;
5     if $j.served(s,t)$ and $Car(s,t,j).CanService(i)$ then $\eta \leftarrow \eta \cup \{(j)\}$;
6 foreach $d \in D$, $k \in K \mid Dist(d,i) < \xi^D D^{max}$ and $k.CanService(i)$ do
7     foreach $l \in r$, $\forall r \in s_{tkd}$ do
8         $\eta \leftarrow \eta \cup \{(d,k,l)\}$;
9 return $\eta$;
```

4.3.4 Distance between solutions

The distance between solutions needs to be well-designed for a better population management, since that the survival selection procedure tends to remove the worst solutions for each cluster of solutions. So, given two solutions s_1 and s_2, the distance between them is defined according to Equation (4.7). It is the sum of the routing distance Δ_r, periodic distance Δ_p, fleet distance Δ_f, and depot distance Δ_d, divided by the current number of problem attributes n_a. For the problems considered in this work, $n_a = 1$ for CVRP, $n_a = 2$ for HVRP, $n_a = 3$ for SDMTPVRP and $n_a = 4$ for the HSDMDMTPVRP.

$$\Delta(s_1, s_2) = \frac{\Delta_r(s_1,s_2) + \Delta_p(s_1,s_2) + \Delta_f(s_1,s_2) + \Delta_d(s_1,s_2)}{n_a} \tag{4.7}$$

Each routing, periodic, fleet and depot term in Equation (4.7) belongs to interval $[0, 1]$. For example, if $\Delta_r(s_1, s_2) = 0$, both solutions have the same edges. So, if $\Delta_r(s_1, s_2) = 1$, the solutions do not have any edge in common. The periodic distance Δ_p is computed in Equation (4.8) where $\delta^p_{is_1s_2}$ is equal to 1 if client i is served with the same delivery pattern in both solutions s_1 and s_2. The fleet distance Δ_f is computed in Equation (4.9) where s_1^{ik} is the number of times that vehicle type k starts a route serving client i in solution s_1. The same valid for s_2^{ik} and s_2. The depot distribution distance Δ_d is computed in Equation (4.10) where s_1^{id} is the number of times that depot d starts a route serving client i in solution s_1. The same valid for s_2^{id} and s_2. Finally, the routing distance Δ_r is calculated by Algorithm 13. Initially it compares each sorted multiset of edges $e(s_1)$ and $e(s_2)$ (see Section 4.2.1) by counting the number of repeated edges S (Lines 3−7), and then calculating and returning the routing distance Δ_r (Lines 9 and 10).

$$\Delta_p(s_1, s_2) = \frac{1}{|U|} \sum_{i \in U} \delta^p_{is_1s_2} \tag{4.8}$$

$$\Delta_f(s_1, s_2) = \frac{1}{|U|} \sum_{i \in U} \sum_{k \in K} \frac{|s_1^{ik} - s_2^{ik}|}{2f_i} \tag{4.9}$$

$$\Delta_d(s_1, s_2) = \frac{1}{|U|} \sum_{i \in U} \sum_{d \in D} \frac{|s_1^{id} - s_2^{id}|}{2f_i} \tag{4.10}$$

Algorithm 13: Routing Distance $\Delta_r(s_1, s_2)$

```
   Input: Solutions $s_1$ and $s_2$
   Output: Routing distance between solutions $s_1$ and $s_2$
1  $a \leftarrow e(s_1)$; $b \leftarrow e(s_2)$;
2  $S \leftarrow 0$; $i \leftarrow 1$; $j \leftarrow 1$;
3  repeat
4      if $a_i = a_j$ then
5          $S \leftarrow S + 1$; $i \leftarrow i + 1$; $j \leftarrow j + 1$;
6      else
7          if $a_i < a_j$ then $i \leftarrow i + 1$ else $j \leftarrow j + 1$;
8  until $i > |a| \vee j > |b|$;
9  $\Delta_r \leftarrow 1 - \frac{2S}{|a||b|}$;
10 return $\Delta_r$;
```

4.3.5 Population Management

We use the same population management mechanism as the one proposed in VIDAL *et al.* [16]. Given a population p, it is responsible for choosing which solutions are used for the search. The procedure tries to keep the best solutions as well as some poor ones to provide a population with diversity.

A population p is a set of solutions s. Let μ^{base} be a base size which is the minimum number of solutions in p, μ^{curr} be the current number of solutions in p during the search, λ be the generation size, ϵ^{elite} be a proportional of elite solutions, and ϵ^{close} be a proportion of the closest solution.

When the maximum size of $\mu^{base} + \lambda$ solutions is reached, it triggers the survival selection process which keeps an elite set of $\mu^{elite} = \lceil \epsilon^{elite} \mu^{base} \rceil$ solutions based on cost, and discards λ solutions based in a biased score.

Given a solution s, its biased score is $BS(s)$ according to Equation (4.11) where $CR(s)$ is the cost rank which is an integer number between $[1, \mu^{curr}]$. A solution with the best rank (equal to 1) has the best cost value according to Equation (4.4). Besides, $DR(s)$ is the diversity rank which is calculated for each solution s based on the average distance $\Delta(s, s')$ (Section 4.3.4) for the $\mu^{close} = \lceil \epsilon^{close} \mu^{base} \rceil$ closest solutions.

The survival selection process eliminates first the solutions with clones ($\Delta(s, s') = 0$), and then the general population is evaluated. The solution s with the worst $BS(s)$ is removed. So, $CR(s)$, $DR(s)$ and $BS(s)$ are updated for each remained solutions in p. This process is repeated until we have μ^{base} solutions left in p.

$$BS(s) = CR(s) + (1 - \frac{\mu^{elite}}{\mu^{curr}})DR_s \tag{4.11}$$

4.3.6 Parent Selection and Crossover

The parent selection and crossover are the ones proposed by VIDAL *et al.* [16] adapted to the HSDMDMTPVRP and its sub-problems. It selects two parent solutions s^1 and s^2, and applies a periodic crossover PIX to generate a valid solution offspring s'.

More specifically, the parent selection process is a dual binary tournament. First, randomly it selects two solutions from the population and picks the one with higher biased fitness value as the first parent s^1. Thenceforth, this process is repeated to find parent s^2 until there are two different parents.

The PIX crossover is divided in four steps. The first one starts by selecting two random numbers $r_1, r_2 \in [0, 1)$, with $r_2 > r_1$, and then calculates $n_1 = \lfloor nr_1 \rfloor$ and $n_2 = \lfloor nr_2 \rfloor$ where $n = |K||T||D|$. After that, it splits $A = \{\{ktd\}, \forall k \in K, \forall t \in T, \forall d \in D\}$ into three partitions A_1, A_2 and A_3 with the sizes n_1, $n_2 - n_1$ and $n - n_2$ respectively. The content of each partition is selected randomly. At last, empty giant tours GT_{tkd} are initialized for each element of $\{ktd\} \in A$.

The second step handles data from A_1 to s'. For each $\{ktd\} \in A_1$, $GT_{tkd} \leftarrow gt(s^1_{tkd})$ copying completely the sequence of visits for the resulting solution (s'). Now, let $\rho \leftarrow |gt(s^1_{tkd})|$ be the size of $gt(s^1_{tkd})$. So, for each $\{ktd\} \in A_3$, it selects two random numbers $r_1, r_2 \in [0, 1]$, with $r_2 > r_1$, and calculates $n_1 = \lfloor \rho r_1 \rfloor$ and $n_2 = \lceil \rho r_2 \rceil$. And then, $GT_{tkd} \leftarrow gt(s^1_{tkd})_{n_1:n_2}$.

The third step handles data from s^2 to s'. For each $\{ktd\} \in A_2 \cup A_3$, let $b = \{i | i \in gt(s^2_{tkd}), i \notin GT_{tkd}\}$ a set of clients respecting the delivery patterns given the current client-period distribution of s' so far, then $GT_{tkd} \leftarrow GT_{tkd} \cup b$. The third step ends with Algorithm 11 applied to every GT_{tkd} with depot d and vehicle type k. The resulting partial solution s^p_{tkd} of Algorithm 11 is added to s'. By the end of the third step, s' may not be a valid solution.

The fourth and last step is executed if s' is not yet a valid solution. This means that not every client $i \in U$ has been served f_i times yet. Let L be the multiset of all clients to be served by s' randomly sorted. For each client $i \in L$, it is inserted into the trip that generates the smallest increase in the total solution cost $C(s')$, respecting at least one delivery pattern $p \in P_i$ and vehicle types in consideration.

At the end of the fourth step, s' is a valid solution and the crossover ends.

4.3.7 Removal and Insertion Operators

Let n_1^- and n_2^- be numbers in interval $[0, 1]$ with $n_2^- > n_1^-$. So, a removal operator o^- removes $n^- \in [n_1^-|U|, n_2^-|U|]$ clients from a solution s and a insertion operation o^+ inserts them back [57]. In this work, the removal of a client e from solution s means that e is removed from all of its services throughout the periods. This cycle is repeated in the Iterated oscillating search (IOS) algorithm (Section 4.3.11).

Algorithm 14 is the framework for the removal process which is used by five removal operators called $RO1$, $RO2$, $RO3$, $RO4$ and $RO5$. We also apply a random removal operator ($RO6$) which does not use Algorithm 14. Given a removal operator, it calls Algorithm 14 and calculates weights based on the current solution s. The weights are defined according to the heuristic $H(s)$. The Algorithm 14 starts by sorting the clients according to their weights given by $H(s)$ in increasing order (Lines 2−3). And then, it selects the number of clients to be removed from s (Line 4). For each client removal (Lines 5−8), the client with index $\lfloor r^p|C|\rfloor$ is removed, where r is a random number in the range $[0, 1]$, and $p > 1$. Higher values of p increase the probabilities to select clients with lower weight values. In most removal operators shown in the literature (see, for example, ROPKE and PISINGER [57]), Lines 2 and 3 are usually inside the main loop (Lines 5−8). This increases the time complexity from $O(n)$ to $O(n^2)$, increasing the algorithm's accuracy (for some cases), however in our experiments, this strategy did not improve the results.

Algorithm 14: Removal Framework

```
Input: Solution $s$, weight heuristic $H$ and set $U$ of served clients
Output: Set of removed clients $L^-$
1 $L^- \leftarrow \{\}; C \leftarrow U;$
2 $Weights \leftarrow H(s);$
3 $Rank \leftarrow \{i | i \in C, \forall j, l \in C, j < l \rightarrow Weight_{Rank_j} < Weight_{Rank_l}\};$
4 $n^- \leftarrow Random(n_1^-, n_2^-)|C|;$
5 for $i = 1$ to $n^-$ do
6     $r \leftarrow Random(0, 1); i \leftarrow \lfloor r^p|C|\rfloor; e \leftarrow Rank_i$ ;
7     $L^- \leftarrow L^- \cup \{e\}; C \leftarrow C \setminus \{e\}; Rank \leftarrow Rank \setminus \{e\};$
8     $Remove(s, e);$
9 return $L^-$;
```

For the removal operators, we use three Shaw variants (RO1 and RO2 and RO3), which are based on SHAW [58], a Average Cost per Unit Transferred (ACUT) removal (RO4), a worst-removal procedure (RO5) and a random removal (RO5). The Shaw variants try to remove similar clients. The similarity between clients $u, v \in U$ is given by Equation (4.12), where $|T|$ is the number of periods; θ_1, θ_2 and θ_3 are positive weights:

$$Shaw(u,v) = \theta_1 \frac{d_{uv}}{D^{max}} + \theta_2 \frac{|q_u - q_v|}{Q} + \theta_3 \frac{|f_u - f_v|}{|T|}. \tag{4.12}$$

Based on previous Shaw heuristics implementations [62, 63], we used the following Shaw variants and θ values:

- Shaw removal (RO1) uses $\theta_1 = 0.6$, $\theta_2 = 0.25$ and $\theta_3 = 0.10$;
- Shaw neighbours removal (RO2) uses $\theta_1 = 1$, $\theta_2 = 0.1$ and $\theta_3 = 0$; and
- Shaw load removal (RO3) uses $\theta_1 = 0$, $\theta_2 = 1$ and $\theta_3 = 0$.

The Average Cost per Unit Transferred (ACUT) removal (RO4) was proposed by PARASKEVOPOULOS *et al.* [65]. It gives to each route a cost per load. In our case, this was adapted and now for each client i, $ACUT_i$ is the negative average value for all routes that serve i (set s_i^r), defined according to Equation (4.13).

$$ACUT_i = \frac{-1}{f_i} \sum_{r \in s_i^r} \frac{c_k^f + c_k^v \sum_{m \in r} Dist(m,d)}{\sum_{m \in r} Load(m)}. \tag{4.13}$$

The Worst removal (RO5) selects and removes clients based on each removal cost, which is the difference of the solution cost without and with it. In this case, each client $i \in U$ has a assigned weight equals to the average removal cost (always negative) per service.

The Random removal (RO6) randomly removes n^- clients. This operator generates a poor set of removed clients, but this helps diversifying the search.

Conversely, every insertion operator o^+ starts with the set of removed clients L^- and inserts them back into solution s. The insertion operator $\Phi(u,s)$ inserts the client i into the best position of solution s. That is, $\Phi(i,s)$ finds the best position to insert i considering the incremental cost for solution s. All delivery patterns and neighborhood $\eta(s,t,i)$ for each period $t \in T$ are analysed to define this best position. Lastly, let $C(\Phi(i,s))$ be the incremental cost for solution s if i is inserted into the best currently available position and $p(\Phi(i,s))$ be the best delivery pattern for i to be inserted into s.

The first insertion operator, called Random greedy insertion (IO1), randomly selects a client $i \in L^-$ and inserts it into the best position $\Phi(i,s)$. This process ends when all removed clients have been inserted back.

The Partial greedy insertion (IO2), randomly selects a subset of L^- of size κ, which is a parameter called "Greed Level". Then, IO2 finds $i \in A \,|\, \forall v \in L^- \Rightarrow C(\Phi(i,s)) \leq C(\Phi(v,s))$ and inserts it into the best position $\Phi(u,s)$. This process is repeated until all removed clients have been inserted back. As L^- decreases in size, there comes a time when $\kappa > |L^-|$. From that moment on, $\kappa \leftarrow |L^-|$.

Finally, the Partial k-regret insertion (IO3) is based on a k-regret criterion as in ROPKE and PISINGER [63]. Given a set of removed clients L^-, it also randomly selects a subset of L^- of size κ. For each client $i \in A$, the k-best insertion positions per period are obtained. Each position has a cost increment, and therefore a k-regret cost can be found for each period $t \in p(\Phi(i, s))$. Thus, the k-regret for i is computed by averaging the k-regret generated for each period defined for the corresponding best delivery pattern. The client $i \in A$ with the largest regret value is than chosen to be inserted into solution s. Again, there comes a time when $\kappa > |L^-|$. Thenceforth, $\kappa \leftarrow |L^-|$.

4.3.8 Local search and repair

The local search procedure *LocalSearch* proposed in this work contains three main steps: route improvement (RI), delivery pattern improvement (PI) and single swaps improvement (SI), in which RI and PI were proposed at VIDAL *et al.* [16]. It applies sequentially these three steps on a current solution s, with the first one selected randomly. The algorithm ends when a maximum number of improvements is reached or if no improvement can be done for solution s.

Step RI tries to improve the total distance travelled by all vehicles per period by applying swap moves. For a solution s, it starts by sorting in random order the pairs (u, t), $u \in U$, for those $t \in T$ in which client u is served. Then, a pair (u, t) is chosen at random. For each $v \in \eta(s, t, u)$ all swap moves are tried in a random order. If any swap move results in a improvement to s, it is applied and RI is terminated.

We use three types of swap moves and all of them need to know the node $a \in N$ (client or depot) which precedes client u and the node $b \in N$ which precedes node v. The first move SM_1 tries to swap, if possible, n_a clients after a with n_b clients after b. The following values are used for (n_a, n_b): $(1, 0)$, $(2, 0)$, $(1, 1)$, $(2, 1)$ and $(2, 2)$. The second move SM_2 requires that u and v are in different routes, and it tries to swap all the clients after a with all clients after b. The last move SM_3 tries to swap all clients after a with all clients before v. In this case, u and v can be in the same route. Since the HSDMDMTPVRP and its sub-problems are symmetrical, when n_a or n_b are larger than 1, all moves check all permutations of the n_a clients and n_b clients during the swapping process.

Step PI tries to find a better delivery pattern distribution for the clients. In a random sequence, for each client $u \in U$, u is removed from s and reinserted according to position $\Phi(u, s)$ (Section 4.3.7), generating solution s'. If the cost of solution s' is better than the one of solution s, Step PI ends with $s \leftarrow s'$. Otherwise, the process continues until all clients are evaluated.

Finally, Step SI tries to improve s by swapping clients with single service at different periods. Let $C = \{i | i \in U, f_i = 1\}$ be a set of clients in a random order. For each client $i \in C$, let $C_i = \{j | j \in C, f_i = 1, Dist(i,j) < D^{max}\xi^D, i < j, p(s,i) \neq p(s,j)\}$ be a (neighbor) set of clients for i in a random order. For each client $j \in C_i$, SI tries to swap i and j. If it improves the solution, the move is kept, otherwise the process continues with the next neighbor.

The repair procedure *Repair* uses the local search presented above to try to find a feasible solution from an infeasible one. As the *Repair* proposed in [16], the *LocalSearch* is applied using 10 times the current penalty weights (Section 4.3.1). If no feasible solution is found, the *LocalSearch* is applied again now using 100 times the current penalty weights.

4.3.9 Initial solution and population

The initial solutions are generated in two phases called Fix and Split, which will compose the initial population. The Fix phase assigns to each client unit $i \in U$, a randomly selected delivery pattern $p \in P_i$, a randomly selected vehicle type $k \in K$ and a randomly selected depot $d \in D$. And then, for each period $t \in p$, i is added to the corresponding giant tour gt_{tkd}. In the Split phase, for each period $t \in T$, each depot $d \in D$ and each vehicle type $k \in K$, the resulting giant tour gt_{tkd}, if not empty, is randomly shuffled and then the Multi Trip Split algorithm (Section 4.3.2) is applied, returning a partial solution s^p_{tkd}. The set with all resulting partial solutions s^p_{tkd} is the initial solution.

The initial population is created in a way that, at the end, we have more diversified solutions as possible. So, it consists of generating $4\mu^{base}$ initial solutions. For the AVNR, each solution is improved using the *LocalSearch* procedure limited to $3|U|$ improvements. For the UHGS, each solution a *Repair* is applied with probability P_{repair} as described at the end of Section 4.3.8 and the resulting solution is added to the correct population.

Using the survival selection process and the initial population creation, the UHGS implements a diversification procedure, that consists of reducing μ^{base} to $\mu^{base}/3$, applying the survival selection process to both populations, restores μ^{base} to its previous value and then initial population procedure executed. The diversification process works like a restart mechanism, keeping most of the previous best solution and introducing a large set of new solutions to the genetic search.

4.3.10 Adaptive layer

The search is divided into steps. When a step ends, the weights used to select the removal and insertion operators are updated based on the score obtained in the last step, considering the number of times that each operator was used. Initially, all operators have the same weight.

The score of an operator is increased by a parameter equal to σ_1 or σ_2 when it identifies a new solution. If a pair of removal-insertion operator finds a new best solution, their scores are increased by σ_1; diversely, if it finds a solution better than the current one, their scores are increased by σ_2. Thus, the weight ϕ_i of operator i (removal or insertion) is updated by Equation (4.14), where π_i is the resulting score, ξ_i is the number of times that operator i has been used in the last step, and the parameter α_A is called *reaction factor.*

$$\phi_i = (1 - \alpha_A)\phi_i + \alpha_A \frac{\pi_i}{\xi_i}. \tag{4.14}$$

4.3.11 Iterated oscillating search (IOS)

Considering the ALNS heuristic presented in Section 3.4, the Iterated Oscillating Search (IOS) is also based on the principle of adaptive destruction and reconstruction, with spaced local searches and oscillating penalty weights. At each iteration, IOS selects with a weighted roulette and applies an operator to *destroy* a solution s and *reconstruct* it in a different way, thus generating a new solution s'. If s' is better than s, the search continues from s'. The search space oscillates with the penalty weights, i.e., when the the penalty weights decrease, the search space is expanded, otherwise, it is reduced.

Algorithm 15 presents the general framework of the IOS implemented in this work. It receives an initial solution s_{prom} as the promising one, and the maximum number of iterations N. It then sets s_{best}, s_{prom} and s_{curr} with very high values. This means that when compared with any other solution, their costs will be artificially higher in order to be discarded (Line 1). This strategy is also applied at Line 10, discarding the solution provided by the local search, in order to diversify the search.

Lines 2−13 constitute the main loop. It starts by defining the auxiliary solution s_{aux} which is modified by the removal and insertion operators selected. Then, solution s_{prom} is updated (Line 7). If the current iteration is multiple of ρ, a local search is applied to s_{prom}, generating a new solution s_{aux}, and s_{prom} is reset (Line 8−10).

The best solution found s_{best} is updated in Line 11, and the current solution s_{curr} is updated in Line 12. If the iteration is multiple of ρ, the penalty weights are updated (Line 13).

After the main loop, if no feasible solution was found (Line 14), s_{best} is updated and can be repaired with a certain probability; otherwise, if the current solution s_{curr} has smaller cost than s_{best}, a repair is applied to s_{curr} and s_{best} is updated. Finally, the best solution found s_{best} is returned.

Algorithm 15: IOS(s_{prom}, N)

```
Input: Initial Solution $s_{prom}$ and maximum number of iterations $N$
Output: Best solution found $s_{best}$
1  $C(s_{best}) \leftarrow \infty$; $C(s_{curr}) \leftarrow \infty$; $C(s_{prom}) \leftarrow \infty$;
2  for $iteration = 1$ to $N$ do
3      $s_{aux} \leftarrow s_{prom}$;
4      Select a removal operator $o^-$ and apply it to $s_{aux}$;
5      Select a insertion operator $o^+$ and apply it to $s_{aux}$;
6      Update scores $\pi_{o^-}$ and $\pi_{o^+}$;
7      if $C(s_{aux}) \leq C(s_{prom})$ then $s_{prom} \leftarrow s_{aux}$;
8      if iteration is multiple of $\rho$ then
9          $s_{aux} \leftarrow LocalSearch(s_{prom})$;
10         $C(s_{prom}) \leftarrow \infty$;
11     if $s_{aux}$ is feasible $\wedge$ $C(s_{aux}) \leq C(s_{best})$ then $s_{best} \leftarrow s_{aux}$;
12     if $C(s_{aux}) \leq C(s_{curr})$ then $s_{curr} \leftarrow s_{aux}$;
13     if iteration is multiple of $\rho$ then Update penalty weights with $s_{curr}$;
14 if $C(s_{best}) = \infty$ then
15     $s_{best} \leftarrow s_{curr}$;
16     if $Random(0,1) < P_{repair}$ then $s_{best} \leftarrow Repair(s_{best})$;
17 else
18     if $C(s_{curr}) < C(s_{best})$ then
19         $s_{curr} \leftarrow Repair(s_{curr})$;
20         if $s_{curr}$ is feasible $\wedge$ $C(s_{curr}) \leq C(s_{best})$ then $s_{best} \leftarrow s_{curr}$;
21 return $s_{best}$.
```

4.3.12 Unified hybrid genetic search

The Unified Hybrid Genetic Search (UHGS) was initially presented in VIDAL *et al.* [16] and then adapted to a wide range of vehicle routing problems in VIDAL *et al.* [76]. It is based on a Genetic Algorithm (GA) [78] which includes a set of search mechanisms such as an advanced diversity control, feasibility control and a restart procedure.

Algorithm 16 details the UHGS. It starts by initializing the penalty weights (Section 4.3.1) and builds the initial population (Section 4.3.9). The main loop occurs between lines 2−11 and stops when the time limit is reached. During each iteration of the main loop, one offspring s' is generated. The selection of parents and the crossover algorithm is described in Section 4.3.6. After the crossover algorithm, the offspring receives an education process, i.e., a *LocalSearch* is applied on it (Section 4.3.8). If $'$ is not feasible, it is added to the population p_{inf} and may be repaired with probability P_{repair} (Lines 4−6). Then, if s' is feasible, it is added to the population p_{feas}. The penalty weights are updated if the current iteration is multiple of ρ, as described in Section 4.3.1 (Line 8). At Line 9, the best solution s_{best} can be updated by the condition computed at Equation (4.15). It accepts s' to update s_{best} if s' is the first feasible solution found; otherwise, given that s' and s_{best} are feasible (or infeasible) at the same time, it accepts s' to update s_{best} if $C(s') < C(s_{best})$. At the end of each iteration, if s_{best} did not change in the last Itr_{div} iterations, it triggers the diversification process described in Section 4.3.9. The algorithm ends at Line 12 returning s_{best}.

$$\begin{aligned} UpdateBS(s_{best}, s') = &(\Omega(s') = 0 \land \Omega(s_{best}) > 0) \lor [(C(s') < C(s_{best}) \land \\ &(\Omega(s') = 0 \land \Omega(s_{best}) = 0) \lor (\Omega(s') > 0 \land \Omega(s_{best}) > 0)] \end{aligned} \tag{4.15}$$

Algorithm 16: Unified Hybrid Genetic Search

```
   Input: Problem and set of parameters
   Output: Best solution found $s_{best}$
1  Startup penalty weights and builds the initial populations $p_{feas}$ and $p_{inf}$;
2  repeat
3      $s'$ ← Select parents, crossover and Educate offspring;
4      if $s'$ is infeasible then
5          $p_{inf}.Add(s')$;
6          if $Random(0,1) < P_{repair}$ then $s' \leftarrow Repair(s')$;
7      if $s'$ is feasible then $p_{feas}.Add(s')$;
8      if iteration is multiple of $\rho$ then Updates penalty weights;
9      if $UpdateBS(s_{best}, s')$ then $s_{best} \leftarrow s$;
10     if $s_{best}$ did not change in the last $Itr_{div}$ iterations then $Diversify(p_{feas}, p_{inf})$;
11 until $T_{current} < T_{limit}$;
12 return $s_{best}$.
```

4.3.13 Adaptive variable neighborhood race

The Adaptive Variable Neighborhood Race (AVNR) is a follow up of the Multi-start scheme proposed in Section 3.4 and is presented in Algorithm 17. It starts with a population size μ^{base}, a proportion of elite solutions ϵ^{elite} and a range $[n_1^-, n_2^-]$, which is used during the removal and insertion operators. These values are updated linearly throughout the steps. During each step, the population is doubled by searching around its solutions as highlighted in Algorithm 15.

Figure 4.3 shows the relationship between the time spent searching around each solution (Itr_{step}), and also shows the number of solutions per step, where the gray ones will be discarded to the next step, based on the survival selection parameters. As we can see, the AVNR heuristic tries to keep the CPU time constant per step.

Figure 4.4 shows an abstraction of the search space where each point represents a solution. For each step, the population (i) passes by a survival selection processes (ii) which discards some solutions (see points marked with a red cross), reducing its size. After that, the IOS is applied to each surviving solution (iii), given the current search parameters, and then the new solutions found (diamonds) are added to the population.

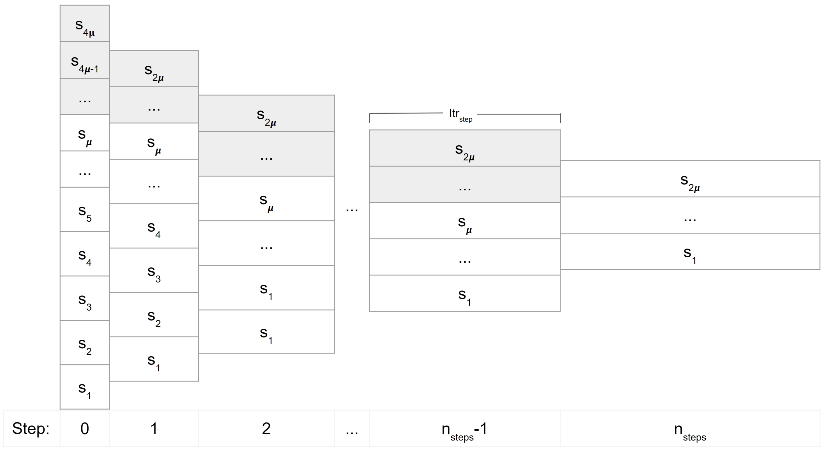


Figure 4.3: Population size and search length per solution thought the steps for the AVNR.

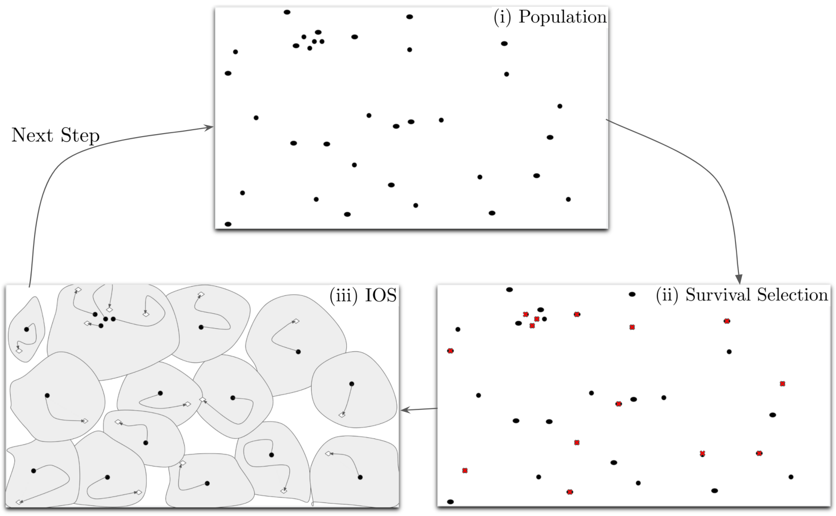


Figure 4.4: Search flow for each step in the AVNR.

Algorithm 17 details the AVNR heuristic. The input includes: the final population size μ_F^{base}, a range $[\epsilon_0^{elite}, \epsilon_F^{elite}]$ for the proportion of elite solutions ϵ^{elite}, a range $[n_{10}^-, n_{1F}^-]$ for n_1^-, a range $[n_{20}^-, n_{2F}^-]$ for n_2^-, the time limit T_{limit}, and the number of "warm-up" iterations Itr_{warm}.

The algorithm starts by initializing $\epsilon^{elite} \leftarrow \epsilon_0^{elite}$, $n_1^- \leftarrow n_{10}^-$ and $n_2^- \leftarrow n_{20}^-$ (Line 1). Then, s_{best} and penalty and operators weights are initialized within the "warm-up phase" (Line 2). Solution s_{best} receives a initial solution (Section 4.3.9) which is updated by applying Algorithm 15 with Itr_{warm} iterations. After Algorithm 15, the operator weights are updated. So, $Itr_{step} \leftarrow \rho$; $\mu^{base} \leftarrow \mu_0^{base}$, which is computed by Equation (4.16) where T_{warm} is the CPU time spent during Algorithm

15; $n_{steps} \leftarrow \lceil \mu_0^{base} \epsilon^{n\mu} \rceil$, where ζ is the proportion between μ_0^{base} and n_{steps}; and λ is defined as $2\mu_0^{base}$.

$$\mu_0^{base} = \left\lceil \sqrt{\frac{Itr_{warm}(T_{limit} - T_{warm})}{\zeta \rho T_{warm}}} \right\rceil \tag{4.16}$$

After the initial population is built as described at Section 4.3.9 (Line 3), we have the main loop between Lines 4−11. For each *step*, μ^{base}, ϵ^{elite}, n_1^- and n_2^- are updated by Equation (4.17) within a linear progression. For example, consider ϵ^{elite}. So, to update it, x is ϵ^{elite}, x_0 is ϵ_0^{elite} and x_F is ϵ_F^{elite} in Equation (4.17). However, since that μ^{base} is an integer number, its result must be rounded. Itr_{step} is also updated according to Equation 4.18 for $step > 1$. It is based on the remaining CPU time $(T_{limit} - T_{current})$, number of IOS iterations spent in the previous step N_{itr}, and CPU time spent in the last step T_{step}.

The $SelectSurvivals(p)$ function is executed to shrink p to μ^{base} solutions (Line 6). Then, for each solution $s \in p$ (Lines 7−10), Algorithm 15 is applied with Itr_{step} iterations, the resulting solution s' is added to the population p, and the best feasible solution is updated as long as condition (4.15) is satisfied.

At the end of each step (Line 11), the operator weights are updated and the number of iterations spent in the current step $N_{itr} \leftarrow \mu^{base} Itr_{step}$ is updated as well. When the algorithm ends, the best solution found s_{best} is returned (Line 12).

$$x \leftarrow x_0 + \frac{step}{n_{step}}(x_F - x_0) \tag{4.17}$$

$$Itr_{step} \leftarrow \rho \left\lceil \frac{N_{itr}(T_{limit} - T_{current})}{T_{step}(n_{steps} - step)(\rho\mu^{base})} \right\rceil \tag{4.18}$$

Algorithm 17: Adaptive Variable Neighborhood Race

```
   Input: Problem and set of parameters
   Output: Best feasible solution found $s_{best}$
1  Initializes $\epsilon^{elite}$, $n_1^-$ and $n_2^-$;
2  Startup $s_{best}$, $Itr_{step}$, $\mu^{base}$, $\lambda$, $n_{steps}$, penalty and operators weights;
3  Build the initial population $p$;
4  for $step = 1$ to $n_{steps}$ do
5      Updates $\mu^{base}$, $\epsilon^{elite}$, $n_1^-$, $n_2^-$ and $Itr_{step}$;
6      $SelectSurvivals(p)$;
7      foreach $s \in p$ do
8          $s' \leftarrow IOS(s, Itr_{step})$;
9          $p.add(s')$;
10         if $UpdateBS(s_{best}, s')$ then $s_{best} \leftarrow s'$;
11     Updates operator and penalty weights and $N_{itr}$;
12 return $s_{best}$.
```

4.4 Computational experiments

This section presents the results of our computational experiments. All heuristics implemented in this chapter share the same code for all mechanisms presented, run on a single thread and all CPU times are expressed in minutes. They were implemented in C programming language using the gcc 10.2 compiler with -O3 option. The computer used in all experiments was an AMD Threadripper 3960x 24c/48t with static clocks @ 4.0Ghz processor, 128GB DDR4 of RAM, running Ubuntu 20.04 x64 operating system.

4.4.1 Parameters tuning

We used the *irace* [79] software for tuning the parameters. This software implements the Iterated Race method, which is a generalization of the Iterated F-race method, establishing correlations between the variables and the objective function, for the automatic configuration of optimization algorithms. It also uses restart mechanisms, truncated sampling distributions to handle correctly parameter bounds, and an elitist racing procedure. We ran 80.000 experiments for tuning each heuristic (UHGS and AVNR), with a time limit of 5 minutes per execution, 24 simultaneous experiments (one per CPU core), and kept the default values for the remaining *irace* parameters.

Besides, we created three instances for tuning different VRP classes: one for HSDMDMTPVRP, one for PVRP and one for HVRP. After some experiments with UHGS and AVNR on each one of the tuning instances, each objective function was normalized by dividing it by its best-known solution value. Thus, the tuning software was able to optimize the set of parameters equally for each one of the instances. By the end of the tuning process, the software returns a set of elite parameters' configurations.

Lastly, for each elite parameter configuration, UGHS and AVNR was executed 10 times on a second set instances to conclude the tuning phase. This new set includes one instance (a sample) for each VRP class (CVRP, SDVRP, HVRP, PVRP, MTVRP, MDVRP, SDMTPVRP and HSDMDMTPVRP) available in literature. We used a normalized objective function and a time limit of 5 minutes for each test. The elite parameter configuration with the best average result was chosen.

Tables 4.1 and 4.2 show, for UHGS and AVNR heuristics respectively, each parameter symbol, description, range of values given to the tuning software (where a value "-" means that the parameter was not tuned) and the final value chosen.

Analysing the UHGS results (Table 4.1), the tuned parameters values are similar to the ones used in VIDAL *et al.* [16], given the similarity between the problems. Now, taking into account the AVNR results (Table 4.2), it shows that the heuristic benefits from a large neighborhood search with a very diverse set of solutions at the beginning (with high values of n_0^1, n_0^2 and low ϵ_0^{elite}), moving towards at a small neighborhood search with a cost focused search at the end (with low values of n_F^1, n_F^2 and high ϵ_F^{elite}).

Table 4.1: List of parameters and values for the UHGS.

Parameter	**Description**	**Range**	**Final Value**
μ^{base}	Base population size	[1,200]	25
λ	Generation size	[1,200]	20
ϵ^{elite}	Proportion of elite solutions	[0,1]	0.45
ϵ^{close}	Proportion of the closest solutions	[0,1]	0.25
ξ^{Inf}	Target proportion of infeasible solutions	[0,1]	0.80
P_{repair}	Repair probability	[0,1]	0.65
Itr^{Div}	Diversification iteration threshold	[500,15000]	4500
η^{base}	Neighborhood base size	[0,50]	45
ξ^D	Proportional distance threshold	[0,1]	0.35
ρ	Penalty weights update interval	-	100
(ω^-, ω^+)	Penalty weights adjustment	-	(0.75, 1.12)

Table 4.2: List of parameters and values for the AVNR.

Parameter	**Description**	Range	Final Value
n_{10}^-	Initial lower rate of removed clients	[0.01,0.50]	0.1
n_{1F}^-	Final lower rate of removed clients	[0.01,0.50]	0.06
n_{20}^-	Initial higher rate of removed clients	[0.01,0.50]	0.28
n_{2F}^-	Final higher rate of removed clients	[0.01,0.50]	0.11
ϵ_0^{elite}	Initial proportion of elite solutions	[0,1]	0.19
ϵ_F^{elite}	Final proportion of elite solutions	[0,1]	0.79
ϵ^{close}	Proportion of the closest solutions	[0,1]	0.15
ζ	Proportion between μ_0^{base} and n_{steps}	[0.01, 10]	0.7
μ_F^{base}	Final base population size	[1,50]	14
ξ^v	Violation tolerance	[0,0.5]	0,055
ρ	Segment size	[10,200]	45
α^A	Adaptivity rate	[0,1]	0.15
η^{base}	Neighborhood base size	[0,50]	35
ξ^D	Proportional distance threshold	[0,1]	0.18
κ	Greed Level	-	$\left\lceil \frac{\lvert U \rvert}{100} \right\rceil$
P_{repair}	Repair probability	-	0.5
w_0	Initial adaptive weights	-	5
Itr_{warmup}	Initial iteration number	-	2000
μ_0^{base}	Initial base population size	-	Equation (4.16)
n_{steps}	Number of steps	-	$\lceil \mu_0^{base} \zeta \rceil$
(ω_b, ω_c)	Adaptive scores	-	(15,10)
(ω^-, ω^+)	Penalty weights adjustment	-	(0.75, 1.12)

The final values of the parameters reported in Tables 4.1 and 4.2 were used to test all the 414 instances, as shown in the next sections. Since both heuristics were designed to run at a fixed time limit, the time used in each instance is based on H^T computed by Equation (4.19) and Table 4.3. For example, for an instance having $H^T = 55$, we used $T_{Limit} = 2.5$ min.

$$H^T = n_{ts} + 10|K| + 5(|D| - 1) \tag{4.19}$$

After the parameters tuning, tests were executed in order to evaluate how each proposed mechanism impacts the AVNR performance for each instance of the second set of training instances. These results are presented in Appendix B.

Table 4.3: Relation between H^T and the time limit T_{Limit}.

H^T	T_{Limit}
$[0, 100)$	2.5
$[100, 150)$	5.0
$[150, 250)$	10.0
$[250, 350)$	20.0
$[350, 450)$	30.0
$[450, 550)$	40.0
$[550, 700)$	50.0
$[700, \infty)$	60.0

4.4.2 Sets of instances and computational results

We tested all the literature instances related to the VRP subclasses from the HS-DMDMTPVRP that we have found in literature. It includes 34 instances for the CVRP, 104 for the MTVRP, 35 for the SDVRP, 42 for the PVRP, 140 for the HVRP, 33 for the MDVRP, 10 for the SDMTPVRP, two real-world instances, and 14 new instances generated for the HSDMDMTPVRP, totaling 414 instances.

In this section, all tables present the instance data, with name and basic attributes, followed by the literature results, with the best-known solution (BKS) (indicated with a * if it is the optimal one), and the average solution values and CPU times for the current best available algorithm. Besides, we show the results of the UHGS and AVNR heuristics for 10 runs: best solution, average solution and average number of iterations (column Itr). The last column shows the average CPU time and the last line reports the percent average deviations to the best-known solutions. The best results are presented in bold. If the BKS is improved, the value is also underlined.

Table 4.4 shows the results for the Capacitated Vehicle Routing Problem (CVRP) instances. The CMT1−CMT14 instances were proposed by CHRISTOFIDES *et al.* [80] and have $50 - 199$ clients, while GWKC1−GWKC20 instances were proposed by GOLDEN *et al.* [81] and have $200-483$ clients. The literature average results are presented by NAGATA and BRÄYSY [82] (column NB) considering 10 runs with an AMD Opteron 6136 CPU. The results are very competitive compared to the current state-of-the-art for the CVRP. The AVNR provided results on average 0.6% better than the UHGS and found a new best solution for instance GWKC6. There is a specific UHGS implementation for the CVRP [16] with an average deviation to BKS of 0.18%.

Table 4.5 depicts the results for the Site-Dependent Vehicle Routing Problem (SDVRP) instances. The first set of SDVRP instances, p01−p23, was proposed by NAG *et al.* [83], with a range of $52 - 324$ clients and $2 - 8$ vehicle classes. The second set of instances, pr01−pr12, was proposed by CHAO *et al.* [84], with a range of $48-1008$ clients and $4-6$ vehicle classes. The optimal solutions and best-known

Table 4.4: Results for the CVRP instances.

		Literature			UHGS			AVNR			
Instance	$\|U\|$	BKS	NB	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
CMT1	50	524.61	**524.61**	0.1	**524.61**	**524.61**	50689	**524.61**	**524.61**	2391203	2.5
CMT2	75	835.26	**835.61**	0.4	**835.26**	**835.26**	30507	**835.26**	**835.26**	1267661	2.5
CMT3	100	826.14	**826.14**	0.3	**826.14**	**826.14**	21191	**826.14**	**826.14**	1675254	5.0
CMT4	150	1028.42	1028.42	1.3	**1028.42**	**1028.42**	19147	**1028.42**	**1028.42**	1453988	10.0
CMT5	199	1291.45	**1291.84**	5.0	1297.89	1300.32	10017	1291.71	1292.37	616101	10.0
CMT6	50	555.43	**555.43**	0.1	**555.43**	**555.43**	44769	**555.43**	**555.43**	2377650	2.5
CMT7	75	909.68	910.41	0.6	**909.68**	**909.68**	19095	**909.68**	**909.68**	1086564	2.5
CMT8	100	865.94	**865.94**	0.4	**865.94**	**865.94**	16169	**865.94**	**865.94**	1465527	5.0
CMT9	150	1162.55	1162.56	2.3	**1162.55**	**1162.56**	19024	**1162.55**	**1162.55**	796202	10.0
CMT10	199	1395.85	**1398.30**	6.5	1397.20	1404.04	9644	1396.89	1398.39	513270	10.0
CMT11	120	1042.11	**1042.11**	0.4	**1042.11**	**1042.11**	14152	**1042.11**	**1042.11**	881124	5.0
CMT12	100	819.56	**819.56**	0.1	**819.56**	**819.56**	20475	**819.56**	**819.56**	1249976	5.0
CMT13	120	1541.14	1542.99	1.8	1543.12	1544.99	13764	1541.14	1542.47	943036	5.0
CMT14	150	866.37	**866.37**	0.2	**866.37**	**866.37**	16751	**866.37**	**866.37**	1058727	5.0
GWKC1	240	5623.47	5632.05	56.6	5631.76	5638.34	11055	**5623.47**	**5625.70**	1049444	20.0
GWKC2	320	8404.61	8440.25	28.8	8447.92	8449.12	5384	8410.34	**8423.73**	595617	20.0
GWKC3	400	11036.20	**11036.22**	43.4	11036.22	11037.78	4522	11036.22	11037.42	491601	30.0
GWKC4	480	13592.88	**13618.55**	64.0	13625.72	13627.24	3849	13616.88	13624.90	452795	40.0
GWKC5	200	6460.98	**6460.98**	2.7	**6460.98**	**6460.98**	9245	**6460.98**	**6460.98**	1077718	10.0
GWKC6	280	8404.26	8413.41	13.8	8412.90	8412.90	7200	**$\underline{\text{8402.24}}$**	**8408.10**	1040654	20.0
GWKC7	360	10102.70	10186.93	36.3	10195.59	10195.59	6073	10105.76	**10117.34**	714451	30.0
GWKC8	440	11635.30	11691.54	96.3	11675.20	11695.31	4809	11635.34	**11639.02**	507329	40.0
GWKC9	255	579.71	**581.46**	17.4	588.20	589.98	12660	581.46	582.00	912992	20.0
GWKC10	323	736.26	**739.56**	27.0	752.66	754.79	6931	741.01	743.78	416353	20.0
GWKC11	399	912.84	**916.27**	39.0	935.78	937.47	6112	917.97	921.28	242699	30.0
GWKC12	483	1102.69	**1108.21**	59.4	1132.70	1137.72	5127	1114.11	1115.81	208387	40.0
GWKC13	252	857.19	**858.42**	15.4	865.30	870.44	17472	860.14	860.74	761597	20.0
GWKC14	320	1080.55	**1080.84**	20.7	1091.57	1101.76	9950	**1080.55**	1084.43	426800	20.0
GWKC15	396	1337.92	**1344.32**	31.2	1370.47	1373.35	9136	1348.01	1349.16	333328	30.0
GWKC16	480	1612.50	**1622.26**	43.6	1661.81	1664.54	7152	1628.02	1631.64	259483	40.0
GWKC17	240	707.76	**707.78**	9.7	708.88	709.74	13303	**707.76**	707.80	1063835	20.0
GWKC18	300	995.13	**995.91**	24.4	1007.81	1010.41	7347	998.31	999.43	334194	20.0
GWKC19	360	1365.60	**1366.70**	35.3	1385.42	1387.70	6417	1370.01	1371.58	451951	30.0
GWKC20	420	1818.32	**1821.65**	47.1	1853.82	1861.09	4109	1832.39	1834.16	588169	30.0
Avg. dev. to BKS (%)			**0,17**		0,71	0,87		0,19	0,27		

results were obtained, respectively, in PESSOA *et al.* [85] and CORDEAU and MAISCHBERGER [86]. The literature average results are from CORDEAU and MAISCHBERGER [86] represented in column CM which were obtained with a parallel algorithm running on a cluster of 64 computers with Intel Xeon E5 472 CPUs. The results show better average results for the AVNR which finds the BKS/optimal solutions in most of the runs and six new best-known solutions.

Tables 4.6 and 4.7 present the results for the Multi Trip Vehicle Routing Problem (MTVRP) instances structured by TAILLARD *et al.* [72], based on CVRP instances with $50-199$ clients (CMT1$-$CMT5, CMT11, CMT12, F11 and F12). The MTVRP instances are built by limiting M_k^t, which can be $T_H^1 = \left[\frac{1.05z^*}{m}\right]$ or $T_H^2 = \left[\frac{1.1z^*}{m}\right]$, with multiples integer values of m. MINGOZZI *et al.* [87] found 42 optimal solutions out of the 104 instances. The literature average solution and time results, with an Intel Xeon 2.8Ghz CPU, are from CATTARUZZA *et al.* [88] represented by column CAFV and the value "-" indicates that no feasible solution was found. Our results show that both heuristics presented in this chapter provide equal or better solutions than the current best-known ones. For all instances, the AVNR heuristic found the best-know solution or improved it. Both heuristics found 43 new best-known solutions: 24 with AVNR and 19 with UHGS and AVNR. For the feasible solutions found, considering just those instances for which there are known-feasible solutions,

Table 4.5: Results for the SDVRP Instances.

				Literature		UHGS			AVNR			
Instance	$\|U\|$	$\|K\|$	M_k^r	BKS	CM	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
p01	55	3	5	640.32*	**640.32**	**640.32**	**640.32**	38991	**640.32**	**640.32**	1879440	2.5
p02	52	2	5	598.10*	**598.10**	**598.10**	**598.10**	43228	**598.10**	**598.10**	2082747	2.5
p03	80	3	7	954.32*	955.13	**954.32**	**954.58**	36516	**954.32**	**954.32**	2036316	5.0
p04	76	2	6	854.43*	**854.43**	**854.43**	**854.43**	33986	**854.43**	**854.43**	2360199	5.0
p05	103	3	6	1003.57*	**1003.57**	**1003.57**	**1003.57**	24516	**1003.57**	**1003.57**	1046836	5.0
p06	104	8	2	1028.52*	**1028.83**	**1028.52**	**1028.52**	25295	**1028.52**	**1028.52**	1079839	5.0
p07	27	3	1	391.30*	**391.30**	**391.30**	**391.30**	446700	**391.30**	**391.30**	3709092	2.5
p08	54	3	2	664.46*	**664.46**	**664.46**	**664.46**	70646	**664.46**	**664.46**	4098183	2.5
p09	81	3	3	948.23*	**948.23**	**948.23**	**948.23**	44777	**948.23**	**948.23**	3334235	5.0
p10	108	3	4	1218.75*	**1218.75**	**1218.75**	**1218.75**	30603	**1218.75**	**1218.75**	1928655	5.0
p11	135	3	5	1448.17*	1458.35	**1448.17**	**1448.17**	47683	**1448.17**	**1448.17**	2290676	10.0
p12	162	3	6	1665.55*	1669.84	**1665.55**	**1665.55**	31127	**1665.55**	**1665.55**	1744259	10.0
p13	54	3	2	1194.18*	**1194.18**	**1194.18**	**1194.18**	74551	**1194.18**	**1194.18**	4262449	2.5
p14	108	3	4	1959.96*	1960.03	**1959.96**	**1959.96**	25389	**1959.96**	**1959.96**	1917915	5.0
p15	162	3	6	2685.09*	**2685.09**	**2685.09**	**2685.09**	24236	**2685.09**	**2685.09**	2178632	10.0
p16	216	3	8	3393.31*	3399.42	3396.70	3399.21	28210	**3393.31**	**3394.82**	1747682	20.0
p17	270	3	10	4066.15	4088.60	4098.30	4106.18	18561	4071.89	**4076.17**	744572	20.0
p18	324	3	12	4747.75	4784.51	4792.17	4802.54	17739	4753.56	**4757.09**	761012	30.0
p19	104	3	4	843.15*	848.27	**843.15**	**843.15**	27287	**843.15**	**843.15**	1412174	5.0
p20	156	3	6	1030.78*	1033.98	**1030.78**	**1030.78**	22674	**1030.78**	**1030.78**	1677981	10.0
p21	209	3	9	1260.01*	1268.06	1262.49	1269.56	11842	**1260.01**	**1262.77**	721376	10.0
p22	122	3	3	1008.71*	**1008.71**	**1008.71**	**1008.71**	30903	**1008.71**	**1008.71**	2092864	10.0
p23	102	3	4	803.29*	**803.29**	**803.29**	**803.29**	25737	**803.29**	**803.29**	1666776	5.0
pr01	48	4	1	1380.77	**1380.77**	**1380.77**	**1380.77**	117874	**1380.77**	**1380.77**	4334073	2.5
pr02	96	4	2	2303.90	2306.92	**2303.90**	**2303.90**	31769	**2303.90**	**2303.90**	1018434	5.0
pr03	144	4	3	2575.36	2575.51	2574.56	2575.33	30229	**2574.56**	**2574.89**	1998293	10.0
pr04	192	4	4	3449.84	3454.90	3459.70	3464.91	12801	**3449.84**	**3452.92**	831172	10.0
pr05	240	4	5	4352.84	4377.35	4376.03	4379.56	20443	**4375.67**	**4375.67**	1200827	20.0
pr06	288	4	6	4422.02	4445.85	4433.11	4460.18	12888	**4406.35**	**4414.40**	928160	20.0
pr07	72	6	1	1889.82	**1889.82**	**1889.82**	**1889.82**	53445	**1889.82**	**1889.82**	3752527	5.0
pr08	144	6	2	2971.01	2977.06	**2969.92**	2972.35	34418	**2969.92**	**2969.92**	1925842	10.0
pr09	216	6	3	3536.20	3558.12	3538.25	3538.72	29797	**3536.20**	**3536.20**	2635914	20.0
pr10	288	6	4	4639.62	4666.01	4650.90	4663.27	22976	**4627.66**	**4634.56**	1439997	30.0
pr11	1008	4	21	12719.65	**12939.70**	13235.62	13268.32	1780	12891.11	13091.74	83682	60.0
pr12	720	6	10	9388.07	9460.36	9623.17	9656.12	5161	**9350.55**	**9445.48**	230420	60.0
Average dev. to BKS (%)					0.26	0.27	0.35		0.01	**0.11**		

AVNR found feasible solutions in all runs, and UHGS did not found feasible solutions in 5 of 10 runs for the CMT2-6-146, 9 of 10 runs for the CMT4-7-154, 1 of 10 runs for the CMT4-8-135, and 8 of 10 runs for the CMT5-10-136. The infeasible results were discarded from all average results.

Table 4.8 depicts the results for the most tested instances of the Heterogeneous Vehicle Routing Problem (HVRP). We considered four groups of instances: the "VFMP-F", which were the first instances proposed in the literature by GOLDEN *et al.* [71], comprising $20-100$ clients and $3-6$ vehicle types, and their variations "VFMP-V", "VFMP-FV' and "HVRP", created by TAILLARD [89]. All best-known solutions were proven to be optimal in PESSOA *et al.* [85]. The literature average solution and time results, with an Intel Core i7-870 CPU, for 30 runs are from PENNA *et al.* [90] and are represented at column PSO. The results show a good performance of both AVNR and UHGS algorithms, with the worst performance of 0.12% from the optimal values found by AVNR, with a clear advantage for the UHGS. There is a specific UHGS implementation for the HVRP [76], for the instances tested in both works, which presents an identical average deviation to BKS of 0.028%.

Considering that the modern heuristics are able to consistently find the optimal solutions for the classic HVRP instances (Table 4.8), a second set of instances,

Table 4.6: Results for the MTVRP instances - Part 1.

				Literature			UHGS			AVNR			
Instance	$\|U\|$	m	M_k^t	BKS	CAFV	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
CMT1	50	1	551	524.61*	**524.61**	0.50	**524.61**	**524.61**	33817	**524.61**	**524.61**	3584618	2.5
		2	275	533.00*	**533.00**	0.50	**533.00**	**533.00**	26057	**533.00**	**533.00**	2253806	2.5
		3	184	-	-	0.50		-	21960	-	-	1598231	2.5
		4	138	-	-	0.50		-	5495	-	-	1395486	2.5
		1	577	524.61*	**524.61**	0.50	**524.61**	**524.61**	33830	**524.61**	**524.61**	3300213	2.5
		2	289	529.85*	530.67	0.50	**529.85**	**529.85**	26489	**529.85**	**529.85**	3483107	2.5
		3	192	552.68	**552.68**	0.50	**552.68**	**552.68**	23778	**552.68**	**552.68**	3752873	2.5
		4	144	546.29*	**546.29**	0.50	**546.29**	**546.29**	22517	**546.29**	**546.29**	3129100	2.5
CMT2	75	1	877	835.26*	838.40	2.0	**835.26**	**835.26**	18715	**835.26**	**835.26**	1804599	2.5
		2	439	835.26*	838.59	2.0	**835.26**	**835.26**	17071	**835.26**	**835.26**	2001082	2.5
		3	292	835.26*	838.58	2.0	**835.26**	**835.26**	15936	**835.26**	**835.26**	1864314	2.5
		4	219	835.26*	839.77	2.0	**835.26**	**835.26**	14760	**835.26**	**835.26**	1663617	2.5
		5	175	835.80*	836.52	2.0	**835.80**	836.11	14503	**835.80**	**835.80**	2039718	2.5
		6	146	858.58	859.42	2.0	857.00	857.52	5347	**855.34**	**856.61**	978881	2.5
		7	125	-	-	2.0	-	-	5400	-	-	919385	2.5
		1	919	835.26*	835.48	2.0	**835.26**	**835.26**	18974	**835.26**	**835.26**	2115342	2.5
		2	459	835.26*	836.46	2.0	**835.26**	**835.26**	17826	**835.26**	**835.26**	1632817	2.5
		3	306	835.26*	837.40	2.0	**835.26**	**835.26**	17479	**835.26**	**835.26**	1647409	2.5
		4	230	835.26*	837.73	2.0	**835.26**	**835.26**	16124	**835.26**	**835.26**	1965753	2.5
		5	184	835.26*	837.99	2.0	**835.26**	**835.26**	15886	**835.26**	**835.26**	1858193	2.5
		6	153	839.22*	846.02	2.0	**839.22**	**839.22**	12392	**839.22**	**839.22**	1620253	2.5
		7	131	844.70	854.70	2.0	**844.54**	844.61	11397	**844.54**	**844.54**	1619950	2.5
CMT3	100	1	867	826.14*	827.96	2.9	**826.14**	**826.14**	18377	**826.14**	**826.14**	2049129	2.5
		2	434	826.14*	827.75	2.9	**826.14**	**826.14**	16882	**826.14**	**826.14**	1968302	5.0
		3	289	826.14*	828.53	2.9	**826.14**	**826.14**	15766	**826.14**	**826.14**	1877735	5.0
		4	217	829.45	829.45	2.9	**828.74**	**828.74**	15115	**828.74**	**828.74**	1707872	5.0
		5	173	832.89	843.72	2.9	**832.88**	**832.88**	10440	**832.88**	**832.88**	1781880	5.0
		6	145	836.22	836.22	2.9	**835.54**	**835.54**	13783	**835.54**	835.61	1688057	5.0
		1	909	826.14*	829.53	2.9	**826.14**	**826.14**	19104	**826.14**	**826.14**	2080292	5.0
		2	454	826.14*	827.96	2.9	**826.14**	**826.14**	17841	**826.14**	**826.14**	1994452	5.0
		3	303	826.14*	829.09	2.9	**826.14**	**826.14**	16652	**826.14**	**826.14**	1975760	5.0
		4	227	826.14*	827.55	2.9	**826.14**	**826.14**	16125	**826.14**	**826.14**	1939159	5.0
		5	182	832.34	832.88	2.9	**831.20**	**831.20**	12935	**831.20**	**831.20**	1615966	5.0
		6	151	834.35	834.35	2.9	**833.98**	**834.02**	13078	**833.98**	834.05	1687561	5.0
CMT4	150	1	1080	1031.00	1034.22	8.2	**1028.42**	**1028.42**	16857	**1028.42**	**1028.42**	1998796	5.0
		2	540	1031.07	1037.89	8.2	**1028.42**	**1028.42**	15085	**1028.42**	**1028.42**	1826756	10.0
		3	360	1028.42	1032.79	8.2	**1028.42**	1028.46	14578	**1028.42**	**1028.42**	1830993	10.0
		4	270	1031.10	1037.09	8.2	**1028.42**	**1028.42**	13898	**1028.42**	**1028.42**	1763895	10.0
		5	216	1031.07	1037.41	8.2	**1028.42**	**1028.42**	13665	**1028.42**	**1028.42**	1710040	10.0
		6	180	1034.61	1041.82	8.2	**1032.40**	**1032.95**	10629	**1032.40**	1033.13	1557258	10.0
		7	154	1068.59	1068.59	8.2	1070.12	1070.12	5370	**1056.86**	**1059.45**	922090	10.0
		8	135	1056.54	1059.68	8.2	1056.61	1057.78	5423	**1055.83**	**1055.83**	1463688	10.0
		1	1131	1031.07	1038.77	8.2	**1028.42**	1028.49	15570	**1028.42**	**1028.42**	1962381	10.0
		2	566	1030.45	1040.39	8.2	**1028.42**	1028.46	14785	**1028.42**	**1028.42**	1929766	10.0
		3	377	1031.59	1032.92	8.2	**1028.42**	**1028.42**	14359	**1028.42**	**1028.42**	1839270	10.0
		4	283	1031.07	1036.33	8.2	**1028.42**	1028.50	14018	**1028.42**	**1028.42**	1861834	10.0
		5	226	1030.86	1035.52	8.2	**1028.42**	1028.56	13555	**1028.42**	**1028.42**	1863797	10.0
		6	189	1030.45	1037.10	8.2	**1029.56**	1029.73	12612	**1029.56**	**1029.56**	1633114	10.0
		7	162	1036.08	1043.60	8.2	**1032.07**	1033.32	9819	**1032.07**	**1032.72**	1566709	10.0
		8	141	1044.32	1048.08	8.2	1044.32	1044.32	10046	**1043.70**	**1043.90**	1616249	10.0

named HVRP DLP, was proposed by DUHAMEL *et al.* [91], representing real-world cases based on 96 French counties, yielding larger instances, comprising $20 - 256$ clients and $2 - 8$ vehicle types. Tables 4.9 and 4.10 show the results for the HVRP DLP instances. The best-known results are from DUHAMEL *et al.* [91] and the average literature solution and CPU time results, using an Intel Xeon 2.80 Ghz CPU, are from the algorithm "GRASP x ELS with BFS split" since it had the better average performance, presented at column DLP [91]. The authors only reported the best solutions out of multiple runs. The results show that UHGS and AVNR achieve better performance than "GRASP x ELS with BFS split" in the several instances. The AVNR heuristic outperformed UGHS in most cases. Considering all 96 instances, our heuristics found the best-known solutions or improved them. We found 12 previous best-known solutions and 84 new best-known solutions: 62 with AVNR, 7 with UHGS, and the 15 remaining ones with both.

Table 4.7: Results for the MTVRP instances - Part 2.

				Literature			UHGS			AVNR			
Instance	$\|U\|$	m	M_k^t	BKS	CAFV	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
CMT5	199	1	1356	1302.43	1308.27	21.4	1336.15	1341.34	4436	**1294.25**	**1295.97**	877113	10.0
		2	678	1302.15	1309.66	21.4	1296.83	1299.77	6966	**1292.11**	**1293.43**	835534	10.0
		3	452	1301.29	1307.85	21.4	1297.80	1300.89	6682	**1293.66**	**1295.45**	810799	10.0
		4	339	1304.78	1308.07	21.4	1295.24	1300.49	6409	**1291.45**	**1293.67**	818758	10.0
		5	271	1300.02	1307.10	21.4	1298.94	1300.87	6139	**1291.50**	**1294.59**	758480	10.0
		6	226	1303.37	1311.16	21.4	1294.17	1299.43	6011	**1293.36**	**1295.97**	758300	10.0
		7	194	1309.40	1313.06	21.4	1295.72	1301.39	5832	**1291.50**	**1296.53**	761678	10.0
		8	170	1303.91	1308.98	21.4	1296.83	1299.26	5625	**1291.71**	**1296.19**	789053	10.0
		9	151	1307.93	1317.03	21.4	1307.67	1322.30	4163	**1296.65**	**1305.35**	706289	10.0
		10	136	1323.01	1329.00	21.4	1336.39	1336.39	4034	**1304.45**	**1314.47**	600225	10.0
		1	1421	1299.86	1310.43	21.4	1339.88	1343.71	6090	**1294.52**	**1296.38**	885184	10.0
		2	710	1305.35	1314.05	21.4	1297.82	1301.97	7196	**1292.36**	**1294.19**	873938	10.0
		3	474	1301.03	1310.93	21.4	1296.04	1302.45	7102	**1291.71**	**1293.35**	786245	10.0
		4	355	1303.65	1312.40	21.4	1301.12	1302.81	6974	**1291.50**	**1294.51**	792777	10.0
		5	284	1300.62	1308.75	21.4	1299.20	1301.37	6901	**1291.74**	**1294.46**	797497	10.0
		6	237	1306.17	1311.40	21.4	1299.87	1304.85	6706	**1291.66**	**1295.35**	782279	10.0
		7	203	1301.54	1313.66	21.4	1298.97	1302.26	6628	**1293.64**	**1295.83**	808449	10.0
		8	178	1308.78	1310.61	21.4	1297.62	1302.41	6414	**1292.59**	**1295.03**	793204	10.0
		9	158	1307.25	1311.32	21.4	1300.96	1304.61	5898	**1291.50**	**1293.49**	765429	10.0
		10	142	1308.81	1316.80	21.4	1302.35	1308.42	5147	**1291.45**	**1297.20**	771710	10.0
CMT11	120	1	1094	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	13314	**1042.11**	**1042.11**	1030810	5.0
		2	547	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	12351	**1042.11**	**1042.11**	981852	5.0
		3	365	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	12205	**1042.11**	**1042.11**	1008183	5.0
		4	274	1078.64	1080.38	5.0	**1078.64**	1078.73	9800	**1078.64**	**1078.64**	706239	5.0
		5	219	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	11827	**1042.11**	**1042.11**	960665	5.0
		1	1146	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	13457	**1042.11**	**1042.11**	1024183	5.0
		2	573	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	12882	**1042.11**	**1042.11**	1025362	5.0
		3	382	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	12242	**1042.11**	**1042.11**	1007836	5.0
		4	287	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	11809	**1042.11**	**1042.11**	1025016	5.0
		5	229	1042.11*	**1042.11**	5.0	**1042.11**	**1042.11**	12113	**1042.11**	**1042.11**	1004847	5.0
CMT12	100	1	861	819.56*	**819.56**	2.3	**819.56**	**819.56**	20257	**819.56**	**819.56**	1539639	5.0
		2	430	819.56*	**819.56**	2.3	**819.56**	**819.56**	19246	**819.56**	**819.56**	1474878	5.0
		3	287	819.56*	**819.56**	2.3	**819.56**	**819.56**	18383	**819.56**	**819.56**	1476092	5.0
		4	215	819.56*	**819.56**	2.3	**819.56**	**819.56**	17852	**819.56**	**819.56**	1532883	5.0
		5	172	845.56	847.73	2.3	**845.37**	**845.37**	11982	**845.37**	**845.37**	1296097	5.0
		6	143	-	-	2.3	-	-	5460	-	-	334742	5.0
		1	902	819.56*	**819.56**	2.3	**819.56**	**819.56**	20433	**819.56**	**819.56**	1553969	5.0
		2	451	819.56*	**819.56**	2.3	**819.56**	**819.56**	19538	**819.56**	**819.56**	1517073	5.0
		3	301	819.56*	**819.56**	2.3	**819.56**	**819.56**	19180	**819.56**	**819.56**	1520691	5.0
		4	225	819.56*	**819.56**	2.3	**819.56**	**819.56**	18344	**819.56**	**819.56**	1495356	5.0
		5	180	824.78*	**824.78**	2.3	**824.78**	**824.78**	16576	**824.78**	**824.78**	1479084	5.0
		6	150	823.14*	**823.15**	2.3	**823.14**	**823.15**	16300	**823.14**	**823.15**	1586380	5.0
F11	71	1	254	241.97	**241.97**	0.7	**241.97**	**241.97**	22348	**241.97**	**241.97**	2529825	2.5
		2	127	250.85	**250.85**	0.7	**250.85**	**250.85**	16151	**250.85**	**250.85**	2926551	2.5
		3	85	-	-	0.7	-	-	15437	-	-	1082011	2.5
		1	266	241.97	**241.97**	0.7	**241.97**	**241.97**	23187	**241.97**	**241.97**	2627013	2.5
		2	133	241.97	**241.97**	0.7	**241.97**	**241.97**	21187	**241.97**	**241.97**	2392507	2.5
		3	89	254.07	**254.07**	0.7	**254.07**	**254.07**	15102	**254.07**	**254.07**	2247250	2.5
F12	134	1	1221	1162.96	**1162.96**	2.7	**1162.96**	**1162.96**	8096	**1162.96**	**1162.96**	520826	5.0
		2	611	1162.96	**1162.96**	2.7	**1162.96**	**1162.96**	7656	**1162.96**	**1162.96**	507103	5.0
		3	407	1162.96	**1162.96**	2.7	**1162.96**	**1162.96**	7407	**1162.96**	**1162.96**	486850	5.0
		1	1279	1162.96	**1162.96**	2.7	**1162.96**	**1162.96**	8397	**1162.96**	**1162.96**	542838	5.0
		2	640	1162.96	**1162.96**	2.7	**1162.96**	**1162.96**	7978	**1162.96**	**1162.96**	510942	5.0
		3	426	1162.96	**1162.96**	2.7	**1162.96**	**1162.96**	7646	**1162.96**	**1162.96**	501940	5.0
Average dev. to BKS (%)					0,28		-0,04	0,02		-0,23	**-0,17**		

Table 4.11 presents the results for two groups of instances of the Periodic Vehicle Routing Problem (PVRP). The first group, p01−p32, represents instances proposed in four papers: p1−p10 was presented in CHRISTOFIDES and BEASLEY [92], p11 in RUSSELL and IGO [93], p12 and p13 in RUSSELL and GRIBBIN [94], and p14−p32 in CHAO *et al.* [95]. The second group of instances, pr01−pr10, was proposed by CORDEAU *et al.* [70]. The main difference from the first group is that the second one has a total time limit per route. The solutions highlighted as optimal in Table 4.11 were retrieved from BALDACCI *et al.* [41], and the best literature average results for 10 runs are from VIDAL *et al.* [16], that is a specific UHGS implementation, which are presented at column VCGLR. The results reveal a

Table 4.8: Results for the HVRP classic instances.

			Literature			UHGS			AVNR			
Instance	$\|U\|$	$\|K\|$	Optimal	PSO	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
HVRP-13	50	6	1517.84	1518.58	0.3	**1517.84**	**1517.84**	70946	**1517.84**	**1517.84**	1978503	5.0
HVRP-14	50	3	607.53	607.64	0.2	**607.53**	**607.53**	36041	**607.53**	**607.53**	2131040	2.5
HVRP-15	50	3	1015.29	1015.33	0.2	**1015.29**	**1015.29**	43383	**1015.29**	**1015.29**	3350159.5	2.5
HVRP-16	50	3	1144.94	1145.04	0.2	**1144.94**	**1144.94**	34849	**1144.94**	**1144.94**	1844147	2.5
HVRP-17	75	4	1061.96	1065.27	0.5	**1061.96**	**1061.96**	39479	**1061.96**	**1061.96**	1950303.5	5.0
HVRP-18	75	6	1823.58	1832.52	0.6	**1823.58**	**1823.58**	25552	**1823.58**	**1823.58**	2129748	5.0
HVRP-19	100	3	1120.30	**1120.34**	1.1	**1120.34**	**1120.34**	15834	**1120.34**	**1120.34**	1931483.5	5.0
HVRP-20	100	3	1534.17	1544.08	1.1	**1534.17**	**1534.17**	16588	**1534.17**	**1534.17**	1203288	5.0
VFMP-F-03	20	5	961.03	961.10	0.1	**961.03**	**961.03**	187541	**961.03**	**961.03**	2078307	2.5
VFMP-F-04	20	3	6437.33	6437.63	0.1	**6437.33**	**6437.33**	295117	**6437.33**	**6437.33**	6135141	2.5
VFMP-F-05	20	5	1007.05	**1007.05**	0.1	**1007.05**	**1007.05**	356746	**1007.05**	**1007.05**	5449813	2.5
VFMP-F-06	20	3	6516.47	**6516.47**	0.1	**6516.47**	**6516.47**	142215	**6516.47**	**6516.47**	2177246	2.5
VFMP-F-13	50	6	2406.36	2419.38	0.5	**2406.36**	**2406.36**	42160	2408.41	2412.43	1896386.5	5.0
VFMP-F-14	50	3	9119.03	**9119.03**	0.2	**9119.03**	**9119.03**	37143	**9119.03**	**9119.03**	3219886	2.5
VFMP-F-15	50	3	2586.37	2586.80	0.3	**2586.37**	**2586.37**	30456	**2586.37**	2586.42	1669057	2.5
VFMP-F-16	50	3	2720.43	2737.59	0.3	**2720.43**	**2720.43**	42204	**2720.43**	**2720.43**	2761459.5	2.5
VFMP-F-17	75	4	1734.53	1748.06	0.9	**1734.53**	**1734.53**	25095	1744.93	1754.80	1715899.5	5.0
VFMP-F-18	75	6	2369.65	2380.98	0.9	2376.16	2376.77	32048	**2369.65**	**2373.88**	1350516	5.0
VFMP-F-19	100	3	8661.81	**8665.31**	1.1	8668.23	8672.46	15672	8662.86	8681.73	1456241	5.0
VFMP-F-20	100	3	4029.61	4051.11	1.6	4039.30	4043.11	15302	4037.90	**4042.89**	1164344	5.0
VFMP-FV-03	20	5	1144.22	**1144.22**	0.1	**1144.22**	**1144.22**	406668	**1144.22**	**1144.22**	4283056.5	2.5
VFMP-FV-04	20	3	6437.33	6437.66	0.1	**6437.33**	**6437.33**	291212	**6437.33**	**6437.33**	6330658	2.5
VFMP-FV-05	20	5	1322.26	**1322.26**	0.1	**1322.26**	**1322.26**	194743	**1322.26**	**1322.26**	1688400.5	2.5
VFMP-FV-06	20	3	6516.47	**6516.47**	0.1	**6516.47**	**6516.47**	139108	**6516.47**	**6516.47**	2268770.5	2.5
VFMP-FV-13	50	6	2964.65	2971.32	0.5	**2964.65**	**2964.65**	96280	**2964.65**	2970.28	3028852.5	5.0
VFMP-FV-14	50	3	9126.90	9126.91	0.2	**9126.90**	**9126.90**	29816	**9126.90**	**9126.90**	1991671	2.5
VFMP-FV-15	50	3	2634.96	2635.02	0.2	**2634.96**	**2634.96**	63680	**2634.96**	**2634.96**	2795710.5	2.5
VFMP-FV-16	50	3	3168.92	3170.81	0.3	**3168.92**	**3168.92**	36757	**3168.92**	**3168.92**	1691732	2.5
VFMP-FV-17	75	4	2004.48	2012.23	0.7	**2004.48**	**2004.48**	25088	2011.12	2014.83	1764234	5.0
VFMP-FV-18	75	6	3147.99	3158.24	0.8	3148.99	**3148.99**	40011	3148.99	3161.99	1598615	5.0
VFMP-FV-19	100	3	8661.81	8664.81	1.0	8668.41	8673.32	16470	8662.86	**8662.86**	1108329.5	5.0
VFMP-FV-20	100	3	4153.02	4155.90	1.0	4154.20	**4154.76**	19178	4217.44	4241.57	1014648	5.0
VFMP-V-03	20	5	623.22	**623.22**	0.1	**623.22**	**623.22**	465862	**623.22**	**623.22**	4099450	2.5
VFMP-V-04	20	3	387.18	**387.18**	0.0	**387.18**	**387.18**	210554	**387.18**	**387.18**	2336193.5	2.5
VFMP-V-05	20	5	742.87	**742.87**	0.1	**742.87**	**742.87**	226824	**742.87**	**742.87**	2533396	2.5
VFMP-V-06	20	3	415.03	**415.03**	0.1	**415.03**	**415.03**	398414	**415.03**	**415.03**	5397642	2.5
VFMP-V-13	50	6	1491.86	1495.61	0.5	**1491.86**	**1491.86**	77293	**1491.86**	**1491.86**	2399844.5	5.0
VFMP-V-14	50	3	603.21	**603.21**	0.2	**603.21**	**603.21**	47567	**603.21**	**603.21**	3353470.5	2.5
VFMP-V-15	50	3	999.82	1001.70	0.3	**999.82**	**999.82**	48894	**999.82**	**999.82**	3084219	2.5
VFMP-V-16	50	3	1131.00	1134.52	0.3	**1131.00**	**1131.00**	50617	**1131.00**	**1131.00**	3124091	2.5
VFMP-V-17	75	4	1038.60	1041.12	0.8	**1038.60**	**1038.60**	38467	**1038.60**	**1038.60**	2033575	5.0
VFMP-V-18	75	6	1800.80	1804.07	0.9	**1800.80**	**1800.80**	25559	**1800.80**	1800.86	2450636.5	5.0
VFMP-V-19	100	3	1105.44	1108.21	1.3	**1105.44**	**1105.44**	23275	**1105.44**	**1105.44**	1476577.5	5.0
VFMP-V-20	100	3	1530.43	1540.32	1.5	1530.52	1530.52	16913	**1530.43**	**1530.45**	1250411.5	5.0
Average dev. to BKS (%)				0.176		0.017	**0.022**		0.064	0.124		

good performance for both UGHS and AVNR heuristics. UGHS provided the worst performance, which is 0.76% far from the best-known results. Although VCGLR provided a good average deviation (0.43%) from the best-known results, AVNR was the best heuristic with an average deviation of 0.26%.

Table 4.12 shows the results for the Multi-Depot Vehicle Routing Problem (MDVRP) instances proposed by CORDEAU *et al.* [70], which have $50-288$ clients and $2-6$ depots. The marked known-optimal solutions and best known-solutions are, respectively, from PESSOA *et al.* [85] and CHRISTIAENS and VANDEN BERGHE [96]. The literature average solution and CPU time results, with an AMD Opteron 250 CPU, for 10 runs are from CHRISTIAENS and VANDEN BERGHE [96] and are presented at column CB. Similarly to the HVRP Classic results, the modern MDVRP heuristics provide a good performance, finding solutions very close to the best-known ones. The UHGS heuristic had the worst performance, ie., a average deviation of 0.10% from the best-known results. In general, the AVNR heuristic

Table 4.9: Results for the HVRP DLP instances. - Part 1

			Literature			UHGS			AVNR			
Instance	$\|U\|$	$\|K\|$	BKS	DLP	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
01	92	4	9210,14	**9210,14**	0.9	**9210.14**	9210.81	17041	**9210.14**	**9210.14**	838831	5.0
02	181	4	11735,97	**11735,97**	16.6	11764.54	11793.60	8412	**11716.59**	11798.20	700410	10.0
03	124	4	10828,83	10828,83	7.4	10760.90	10772.38	19319	**10726.70**	**10753.43**	1745495	10.0
04	183	4	10891,18	10891,18	15.7	10829.99	10857.47	8606	**10741.63**	**10776.95**	505978	10.0
05	116	5	10939,58	10939,58	6.8	10891.24	**10901.42**	23875	**10869.04**	10913.90	1460149	10.0
06	121	8	11783,83	12296,4	6.0	11838.66	11864.06	19294	**11700.06**	**11805.24**	1795682	10.0
07	108	4	8117,44	8117,44	3.3	8089.65	8103.38	28153	**8071.97**	**8077.69**	1692372	10.0
08	84	3	4598,49	4598,49	5.1	4594.07	4596.08	25837	**4591.75**	**4592.01**	1276535	5.0
09	167	5	7728,34	7734,83	10.6	7717.24	7733.33	10876	**7660.54**	**7688.97**	1899838	10.0
10	69	4	2107,55	**2107,55**	0.4	**2107.55**	**2107.55**	29546	**2107.55**	**2107.55**	2033575	5.0
11	95	4	3370,47	3370,47	4.4	**3367.41**	**3367.41**	19424	**3367.41**	**3367.41**	1110463	5.0
12	112	4	3543,99	**3543,99**	4.4	**3543.99**	**3543.99**	20044	**3543.99**	**3543.99**	2389203	10.0
13	119	5	6716,69	6770,99	8.6	**6696.43**	6697.38	26263	**6696.43**	**6697.01**	1240278	10.0
14	176	4	5673,90	5706,42	7.1	5738.74	5751.09	8160	**5672.10**	**5700.47**	683234	10.0
15	188	7	8282,56	8282,56	15.4	8309.69	8350.07	19218	**8208.59**	**8237.10**	1658158	20.0
16	129	6	4156,97	**4156,97**	3.1	**4156.97**	**4156.97**	19782	**4156.97**	**4156.97**	975292	10.0
17	105	3	5370,05	5378,52	5.5	5367.48	5368.64	11865	**5362.83**	**5365.44**	711486	5.0
18	256	5	9783,95	9832,84	18.3	9733.33	9782.77	7403	**9670.00**	**9727.28**	822151	20.0
19	224	5	11875,83	11899,61	24.4	11774.47	11793.12	12771	**11754.92**	**11789.90**	1139686	20.0
21	126	3	5175,03	5175,03	10.3	**5139.84**	**5139.84**	24861	**5139.84**	**5139.84**	1395576	10.0
22	239	2	13129,61	13129,61	16.1	13114.91	13143.75	12362	**13100.66**	**13123.90**	1504221	20.0
23	203	4	7822,02	7822,02	19.2	7828.49	7841.56	6469	**7745.49**	**7768.41**	600190	10.0
24	163	4	9192,93	9226,5	6.9	9131.55	9146.91	11404	**9116.20**	**9136.13**	1258146	10.0
25	143	6	7274,26	7274,26	10.4	**7134.69**	7196.58	14536	7143.38	**7172.67**	986317	10.0
26	126	5	6501,16	6501,16	7.6	**6415.71**	**6423.08**	20285	6428.66	6451.94	1234330	10.0
27	220	5	8514,25	8514,25	24.2	8467.44	8488.86	12518	**8449.92**	**8489.52**	938078	20.0
28	141	5	5555,15	6486,58	5.5	**5532.61**	**5537.28**	20093	5534.47	5546.95	1336312	10.0
29	164	4	9170,59	9170,59	12.0	9150.30	9158.84	15048	**9135.66**	**9144.15**	1121601	10.0
2A	113	6	7856,67	7856,67	8.8	**7793.16**	**7793.98**	26503	**7793.16**	7822.39	1151243	10.0
2B	107	6	8535,50	8702,25	2.0	**8453.35**	8457.26	29347	8463.90	**8487.02**	1742946	10.0
30	112	3	6349,53	6354,96	2.6	**6315.51**	6344.09	14317	6320.76	**6333.66**	813132	5.0
31	131	8	4091,81	4141,78	5.0	**4091.52**	**4091.52**	19128	**4091.52**	**4091.52**	1083799	10.0
32	244	8	9510,03	9606,39	27.3	9450.18	9508.17	9426	**9367.64**	**9423.66**	676057	20.0
33	189	7	9494,43	9494,43	6.1	9417.79	9471.66	14394	**9411.01**	**9437.02**	956449	20.0
34	136	6	5803,68	5854,57	5.0	5755.99	5770.48	16424	**5752.99**	**5768.57**	921064	10.0
35	168	6	9625,28	9848,4	11.1	9683.21	9697.46	10710	**9566.61**	**9623.13**	1036831	10.0
36	85	6	5752,34	5759,34	1.7	**5684.61**	5685.07	25482	**5684.61**	**5684.61**	1444626	5.0
37	161	5	6920,07	6967,63	14.8	6935.31	6947.64	12321	**6867.35**	**6894.84**	1032612	10.0
38	205	5	11371,83	11474,18	20.3	11293.52	11321.32	11424	**11265.96**	**11315.94**	1762036	20.0
39	77	5	2934,55	2934,55	3.0	**2918.87**	2919.18	31838	**2918.87**	**2918.87**	1439900	5.0
40	132	5	11178,31	11225,34	10.6	11201.94	11234.79	17684	**11122.37**	**11153.32**	2040287	10.0
41	135	7	7629,55	7676,85	8.9	7562.31	7577.50	15713	**7553.08**	**7556.41**	804038	10.0
42	178	7	11068,07	11068,07	20.7	10770.66	**10789.91**	18127	**10762.59**	10793.96	1035175	20.0
43	86	7	8762,60	8764,75	3.7	**8737.02**	**8737.02**	40710	**8737.02**	**8737.02**	2389595	10.0
44	172	3	12459,77	12459,77	14.5	12199.97	12228.13	12156	**12171.02**	**12181.82**	965244	10.0
45	170	3	10515,01	10515,01	4.6	10561.33	10601.84	9776	**10428.57**	**10459.74**	661800	10.0
46	250	5	24751,07	24751,07	18.8	24537.34	24615.45	8950	**24315.07**	**24446.54**	516903	20.0

was better than all approaches, finding the best-known solutions for all instances, except for p08 and pr10. There is a specific UHGS implementation for the CVRP [16] with an average deviation to BKS of 0.09%.

Table 4.13 presents the results for the Site Dependant Multi-trip Periodic Vehicle Routing Problem (SDMTPVRP), proposed by ALONSO *et al.* [75], comprising $50-1000$ clients, $2-6$ periods, and $2-13$ vehicle types. The best-known solutions, averages and CPU time results, with an Intel Pentium 4 1.6Ghz CPU, for 10 runs found in [75] are reported in Table 4.13 by columns BKS, AAB and T(min), respectively. Again, the results found by UHGS and AVNR heuristics were very good: AVNR found eight new best solutions and UHGS one. Besides, AVNR has the best overall results, as shown in the last line of Table 4.13.

The last computational experiments comprise a set of 14 instances proposed for the Heterogeneous Site-Dependent Multi-depot Multi-trip Periodic Vehicle Routing Problem (HSDMDMTPVRP), which are based on a real-world case coming from

Table 4.10: Results for the HVRP DLP Instances. - Part 2

			Literature			UHGS			AVNR			
Instance	$\|U\|$	$\|K\|$	BKS	DLP	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
47	111	5	16290,51	16290,51	5.4	**16156.11**	16165.68	27225	**16156.11**	**16156.11**	1740633	10,0
48	111	5	21470,94	21470,94	6.1	**21257.38**	21291.38	24870	**21257.38**	**21280.88**	1530657	10,0
49	246	8	16302,37	16302,37	38.3	16417.76	16476.04	6753	**16118.25**	**16191.00**	446612	20,0
50	187	6	12510,95	12510,95	12.3	12317.41	**12334.25**	15593	**12295.77**	12336.81	914315	20,0
51	129	3	7797,58	7797,58	5.4	**7721.47**	**7722.24**	20034	**7721.47**	7737.79	1151670	10,0
52	59	3	4029,42	4029,42	0.7	**4027.27**	**4027.27**	33753	**4027.27**	**4027.27**	2639870	2,5
53	115	3	6434,83	6455,54	2.7	**6434.83**	**6434.83**	12697	**6434.83**	**6434.83**	805719	5,0
54	172	4	10421,44	10421,44	12.3	10366.70	10389.39	8920	**10309.09**	**10319.43**	503818	10,0
55	56	3	10244,34	10247,86	3.2	**10244.34**	**10244.34**	38107	**10244.34**	**10244.34**	2573405	2,5
56	153	4	31231,62	31231,62	3.5	31237.04	31332.32	11776	**30989.83**	**31093.75**	1165444	10,0
57	163	4	44974,37	45071,82	13.3	45360.38	45463.21	9220	**44840.48**	**45042.28**	1196371	10,0
58	220	6	23537,74	23537,74	20.3	23485.38	23549.03	11667	**23299.28**	**23339.36**	756652	20,0
59	193	6	14326,01	14326,01	16.9	14254.91	**14264.95**	15839	**14251.91**	14286.26	926617	20,0
60	137	4	17090,08	17119,69	7.7	17074.77	17092.77	15621	**17045.33**	**17052.85**	894166	10,0
61	111	3	7295,67	7308,2	5.2	**7289.74**	7289.74	11320	**7289.74**	**7289.74**	1404786	5,0
62	225	5	23272,12	23272,12	6.6	22997.74	23033.39	10796	**22872.40**	**22942.68**	532771	20,0
63	174	5	20122,05	20325,31	14.9	20031.13	20107.15	7935	**19890.65**	**19955.99**	433335	10,0
64	161	3	17135,16	17135,16	9.3	**17135.16**	17144.87	11354	**17135.16**	**17135.16**	1383167	10,0
65	223	3	13072,08	13097,23	6.0	13015.86	13021.05	9832	**12959.16**	**12986.44**	760971	20,0
66	150	4	12924,55	12924,55	10.9	12849.22	12865.86	11601	**12751.27**	**12779.40**	596361	10,0
67	172	5	10976,12	10976,12	18.2	10872.97	10955.41	7941	**10830.32**	**10875.86**	533061	10,0
68	125	4	9122,76	9135,27	9.5	8946.45	8974.09	17097	**8889.03**	**8901.75**	1108449	10,0
69	152	4	9299,47	9299,47	9.9	9227.69	9255.97	9320	**9140.67**	**9158.53**	555911	10,0
70	78	4	6689,61	6689,61	2.0	**6684.56**	6684.83	30768	**6684.56**	**6684.56**	1174247	5,0
71	186	3	9912,71	9938,77	14.9	9953.37	9979.27	6502	**9870.59**	**9953.06**	584546	10,0
72	186	4	5956,27	5956,27	14.3	5922.99	5941.24	9890	**5887.74**	**5896.20**	599111	10,0
73	137	5	10251,80	10909,9	5.3	10212.96	10222.32	16177	**10202.41**	**10218.20**	1289814	10,0
74	125	5	11701,10	12197,85	5.2	11660.70	**11684.56**	21295	**11602.83**	11689.09	2105688	10,0
75	20	3	452,85	**452,85**	0.0	**452.85**	**452.85**	372685	**452.85**	**452.85**	5190946	2,5
76	152	8	12062,86	**12062,86**	11.1	12100.16	12132.36	11530	**12038.38**	12139.64	1525047	10,0
77	190	3	6930,61	7010,15	9.6	6956.42	6967.56	7921	**6901.44**	**6942.63**	402739	10,0
78	190	4	7077,81	7077,81	9.8	**7041.95**	**7062.04**	7499	7054.81	7082.00	507052	10,0
79	147	4	7278,10	7287,38	4.4	**7257.97**	7262.85	13979	**7257.97**	**7257.97**	1230373	10,0
80	171	3	6841,07	6841,07	13.9	6823.04	6830.37	9733	**6819.07**	**6823.09**	604402	10,0
81	106	4	10700,47	10719,66	2.4	**10591.13**	10607.45	27818	10591.77	**10603.56**	1482693	10,0
82	79	3	4772,94	4774,26	2.4	4742.28	4762.80	23754	**4718.27**	**4734.58**	1084908	5,0
83	124	4	10019,83	**10019,83**	5.1	10038.09	10046.68	19970	**10019.15**	10021.03	2041504	10,0
84	105	4	7262,81	7270,78	4.5	7229.48	7237.95	12592	**7227.88**	**7227.88**	761231	5,0
85	146	4	8893,92	8893,92	3.9	8800.29	8812.36	13733	**8763.90**	**8780.96**	1188393	10,0
86	153	5	9088,64	9094,77	9.0	9056.19	9071.31	14011	**9031.60**	**9051.99**	1450211	10,0
87	108	4	3753,87	3781,07	1.1	**3753.87**	**3753.87**	25572	**3753.87**	**3753.87**	2741016	10,0
88	127	5	12442,48	12568,64	17.4	12440.03	12474.03	17222	**12413.25**	**12435.80**	1683138	10,0
89	134	5	7143,51	7216,53	9.7	7132.71	7139.72	15537	**7080.77**	**7096.09**	870783	10,0
90	102	4	2359,42	2359,42	5.2	2282.77	2293.06	13729	**2278.85**	**2287.20**	770989	5,0
91	196	4	6441,06	6441,06	18.6	6468.59	6483.83	6681	**6402.96**	**6414.11**	986361	10,0
92	35	3	564,39	**564,39**	0.3	**564.39**	**564.39**	87105	**564.39**	**564.39**	4756449	2,5
93	39	6	1036,99	**1036,99**	0.5	**1036.99**	**1036.99**	45999	**1036.99**	**1036.99**	1643735	2,5
94	46	5	1378,25	1378,66	0.3	**1378.25**	**1378.25**	70996	**1378.25**	**1378.25**	2890279	2,5
95	183	2	6245,03	6251,59	8.8	6223.32	6229.75	9138	**6173.10**	**6204.80**	1013863	10,0
Average dev. to BKS (%)				0,54		-0.40	-0.23		-0,74	**-0,51**		

the automotive industry. Instances p07 and p12 were created taking into account the real data. We converted the distance matrix based on traffic distances to a 2D coordinates system in order to make it easier to be tested by other algorithms, still retaining the real case characteristics. The remaining instances were created considering clients and depots randomly located inside on a boundary box from one of the real instances. The demand and frequency of each client were sampled from probability distribution functions created considering the real clients' data. Each type of vehicle has the same capacity and speed obtained from the real case, however, we added a noise for the fix and variable costs. The results are presented in Table 4.14, that shows a similar behaviour to the one observed for the HVRP instances. For the smaller instances, the UHGS heuristic provided better results, whereas AVNR was better for the larger ones. The AVNR heuristics also provides

Table 4.11: Results for the PVRP Instances.

					Literature			UHGS			AVNR			
Instance	$\|U\|$	$\|T\|$	n_{ts}	M_k^r	BKS	VCGLR	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
p01	50	2	50	3	524.61	**524.61**	0.2	**524.61**	**524.61**	78321	**524.61**	**524.61**	4635321	2.5
p02	50	5	104	3	1322.87	**1322.87**	0.4	**1322.87**	**1322.87**	40568	**1322.87**	**1322.87**	2567826	5.0
p03	50	5	50	1	524.61	**524.61**	0.2	**524.61**	**524.61**	214867	**524.61**	**524.61**	6366362	2.5
p04	75	2	75	5	835.26	836.59	1.1	**835.26**	835.28	34243	**835.26**	**835.26**	2623777	2.5
p05	75	5	153	6	2024.96	2033.72	2.3	**2024.96**	2026.79	39121	**2024.96**	**2025.58**	1563915	10.0
p06	75	10	75	1	835.26	842.48	0.9	**835.26**	**835.26**	95302	**835.26**	**835.26**	3357352	2.5
p07	100	2	100	4	826.14	827.02	0.9	**826.14**	**826.14**	22030	**826.14**	**826.14**	2145096	5.0
p08	100	5	202	5	2022.47	2022.85	2.5	**2022.47**	**2022.47**	28958	2022.47	2024.07	1299626	10.0
p09	100	8	100	1	826.14	826.94	1.0	**826.14**	**826.14**	51587	**826.14**	**826.14**	2189370	5.0
p10	100	5	174	4	1593.43	1605.22	1.8	**1593.43**	1594.01	29974	**1593.43**	**1593.43**	1864869	10.0
p11	126	5	191	4	770.89	**775.84**	4.6	774.90	776.90	19255	774.57	775.87	967097	10.0
p12	163	5	185	3	1186.47	1195.29	5.3	1192.56	1194.41	16549	1188.13	**1190.27**	764149	10.0
p13	417	7	457	9	3462.73	3599.86	40.0	3711.93	3730.52	6397	3533.21	**3550.01**	652214	40.0
p14	20	4	40	2	954.81	**954.81**	0.1	**954.81**	**954.81**	215103	**954.81**	**954.81**	3408970	2.5
p15	38	4	72	2	1862.63	**1862.63**	0.2	**1862.63**	**1862.63**	79899	**1862.63**	**1862.63**	4405200	2.5
p16	56	4	104	2	2875.24	**2875.24**	0.3	**2875.24**	**2875.24**	33018	**2875.24**	**2875.24**	2587786	5.0
p17	40	4	80	4	1597.75	**1597.75**	0.3	**1597.75**	**1597.75**	34790	**1597.75**	**1597.75**	2271954	2.5
p18	76	4	144	4	3131.09	**3131.09**	0.9	**3131.09**	**3131.09**	35494	**3131.09**	**3131.09**	2784390	10.0
p19	112	4	208	4	4834.34	4834.50	2.3	**4834.34**	**4834.34**	19282	**4834.34**	**4834.34**	1882963	10.0
p20	184	4	336	4	8367.40	**8367.40**	4.0	**8367.40**	**8367.40**	24281	**8367.40**	**8367.40**	2327398	30.0
p21	60	4	120	6	2170.61	**2170.61**	0.9	**2170.61**	**2170.61**	29019	**2170.61**	**2170.61**	1745844	5.0
p22	114	4	216	6	4193.95	4194.23	4.3	4194.57	4194.57	21926	**4193.95**	**4193.95**	995952	10.0
p23	168	4	312	6	6420.71	6434.10	4.3	**6420.71**	**6420.71**	23386	**6420.71**	**6420.71**	2050897	20.0
p24	51	6	90	3	3687.46	**3687.46**	0.3	**3687.46**	**3687.46**	69100	**3687.46**	**3687.46**	2336271	2.5
p25	51	6	90	3	3777.15	**3777.15**	0.6	**3777.15**	**3777.15**	67305	**3777.15**	**3777.15**	2314061	2.5
p26	51	6	90	3	3795.32	**3795.32**	0.3	**3795.32**	**3795.32**	60633	**3795.32**	**3795.32**	2923755	2.5
p27	102	6	180	6	21833.87	21885.70	3.5	21880.16	21934.83	26124	21846.02	**21852.31**	970549	10.0
p28	102	6	180	6	22242.51	22272.60	4.7	22278.39	22352.22	25054	22252.20	**22270.42**	840591	10.0
p29	102	6	180	6	22543.76	22564.05	3.9	22575.09	22599.19	25780	22544.63	**22549.22**	763353	10.0
p30	153	6	270	9	73875.19	74534.38	10.0	74681.38	75086.14	23146	74134.98	**74288.52**	705313	20.0
p31	153	6	270	9	75957.62	76686.65	10.0	76562.84	76940.29	22805	76317.02	**76528.28**	940853	20.0
p32	153	6	270	9	77591.23	78168.82	10.0	78284.84	78562.04	21533	77880.03	**78055.32**	894567	20.0
pr01	48	4	96	2	2209.02	**2209.02**	0.3	2209.02	2209.02	38669	**2209.02**	**2209.02**	2410228	5.0
pr02	96	4	192	4	3767.50	3768.86	2.5	3767.54	3769.89	19607	**3767.50**	**3767.95**	1384054	10.0
pr03	144	4	288	6	5153.54	5174.80	7.3	5161.36	5172.33	18689	5155.55	**5161.00**	1527242	20.0
pr04	192	4	384	8	5877.37	5936.16	10.0	5941.93	5964.67	15731	5902.41	**5907.77**	1243022	30.0
pr05	240	4	480	10	6581.86	6651.76	20.0	6786.45	6807.53	10313	6605.47	**6619.77**	669514	40.0
pr06	288	4	576	12	8207.21	**8284.94**	20.0	8454.67	8472.92	7367	8294.95	8302.48	491121	50.0
pr07	72	6	216	3	4996.14	**4996.14**	1.5	**4996.14**	**4996.14**	22182	**4996.14**	**4996.14**	1549396	10.0
pr08	144	6	432	6	6970.68	7035.52	10.0	7041.54	7065.51	20156	6982.77	**6998.36**	1798871	30.0
pr09	216	6	648	9	10038.43	10162.22	20.0	10199.84	10248.24	12426	10072.34	**10091.35**	776847	50.0
pr10	288	6	864	12	12897.01	**13091.00**	20.0	13508.17	13551.68	3841	13069.35	13160.66	339528	60.0
Average dev. to BKS (%)						0.43		0.63	0.76		0.18	**0.26**		

a better average deviation.

Table 4.12: Results for the MDVRP instances.

				Literature			UHGS			AVNR			
Instance	$\|U\|$	$\|D\|$	M_k^r	BKS	CB	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
p01	50	4	4	576.87*	**576.87**	1.32	**576.87**	**576.87**	44364	**576.87**	**576.87**	1113981	2.5
p02	50	4	2	473.53*	**473.53**	1.22	**473.53**	**473.53**	53867	**473.53**	**473.53**	3634227	2.5
p03	75	5	3	641.19*	**641.19**	2.54	**641.19**	**641.19**	48201	**641.19**	**641.19**	2726451	5.0
p04	100	2	8	1001.04*	1001.16	3.28	**1001.04**	**1001.04**	26053	**1001.04**	**1001.04**	1409898	5.0
p05	100	2	5	750.03*	750.19	4.72	**750.03**	**750.03**	23124	**750.03**	**750.03**	2139013	5.0
p06	100	3	6	876.50*	**876.50**	3.70	**876.50**	**876.50**	27118	**876.50**	**876.50**	1739615	5.0
p07	100	4	4	881.97*	**881.97**	3.52	**881.97**	**881.97**	25414	**881.97**	**881.97**	1679389	5.0
p08	249	2	14	4372.78	**4381.91**	18.90	4393.03	4407.91	12622	4391.77	4401.60	722953	20.0
p09	249	3	12	3858.66	3870.47	22.39	3859.17	3860.16	18143	**3858.66**	**3860.05**	710329	20.0
p10	249	4	8	3631.11	3647.09	21.35	**3631.11**	3632.44	15545	**3631.11**	**3632.41**	953085	20.0
p11	249	5	6	3546.06	3547.45	23.55	**3546.06**	**3546.06**	15909	**3546.06**	**3546.06**	962796	20.0
p12	80	2	5	1318.95*	**1318.95**	3.17	**1318.95**	**1318.95**	32548	**1318.95**	**1318.95**	2376841	2.5
p13	80	2	5	1318.95*	**1318.95**	3.22	**1318.95**	**1318.95**	28504	**1318.95**	**1318.95**	2544222	2.5
p14	80	2	5	1360.12*	**1360.12**	3.25	**1360.12**	**1360.12**	44468	**1360.12**	**1360.12**	2902776	2.5
p15	160	4	5	2505.42*	**2505.42**	9.99	**2505.42**	**2505.42**	25184	**2505.42**	**2505.42**	1521832	10.0
p16	160	4	5	2572.23*	**2572.23**	12.28	**2572.23**	**2572.23**	19664	**2572.23**	**2572.23**	1848810	10.0
p17	160	4	5	2709.09*	**2709.09**	12.36	**2709.09**	**2709.09**	19468	**2709.09**	**2709.09**	1710508	10.0
p18	240	6	5	3702.85*	**3702.85**	20.39	**3702.85**	**3702.85**	17557	**3702.85**	**3702.85**	1416076	20.0
p19	240	6	5	3827.06*	**3827.06**	28.67	**3827.06**	**3827.06**	13658	**3827.06**	**3827.06**	1927319	20.0
p20	240	6	5	4058.07*	**4058.07**	31.93	**4058.07**	**4058.07**	14071	**4058.07**	**4058.07**	1385869	20.0
p21	360	9	5	5474.84*	5476.61	41.26	**5474.84**	**5474.84**	11404	**5474.84**	**5474.84**	772016	30.0
p22	360	9	5	5702.16	5703.70	65.34	**5702.16**	**5702.16**	11852	**5702.16**	**5702.16**	645603	30.0
p23	360	9	5	6078.75	**6078.75**	72.99	**6078.75**	**6078.75**	8584	**6078.75**	**6078.75**	609879	30.0
pr01	48	4	1	861.32*	**861.32**	1.55	**861.32**	**861.32**	35778	**861.32**	**861.32**	1903972	2.5
pr02	96	4	2	1307.34	**1307.34**	4.68	**1307.34**	**1307.34**	13338	**1307.34**	**1307.34**	1476249	5.0
pr03	144	4	3	1803.80*	**1803.80**	10.02	**1803.80**	**1803.80**	13651	**1803.80**	**1803.80**	1966449	10.0
pr04	192	4	4	2058.31	2058.76	15.82	**2058.31**	**2058.31**	9668	**2058.31**	**2058.31**	477441	10.0
pr05	240	4	5	2331.20	2338.19	21.56	2341.62	2346.54	8718	**2331.20**	**2332.39**	536322	20.0
pr06	288	4	6	2676.30	2686.77	31.61	2681.78	2690.07	7011	**2676.30**	**2679.24**	363737	20.0
pr07	72	6	1	1089.56*	**1089.56**	2.83	**1089.56**	**1089.56**	26286	**1089.56**	**1089.56**	2149324	5.0
pr08	144	6	2	1664.85*	1666.55	8.15	**1664.85**	**1664.85**	17668	**1664.85**	**1664.85**	2047432	10.0
pr09	216	6	3	2133.20	2135.05	18.10	**2133.20**	2133.24	14369	**2133.20**	**2133.20**	1277676	20.0
pr10	288	6	4	2867.46	**2873.80**	28.91	2896.41	2906.99	5587	2868.26	2877.54	355598	20.0
Average dev. to BKS (%)					0.07		0.06	0.10		0.01	**0.04**		

Table 4.13: Results for the SDMTPVRP instances.

					Literature			UHGS			AVNR			
Instance	$\|U\|$	$\|T\|$	n_{ts}	$\|K\|$	BKS	AAB	T(min)	Best	Average	*Itr*	Best	Average	*Itr*	T(min)
p01	50	2	64	2	2690.41	2695.96	0.78	**2690.41**	**2690.41**	32670	**2690.41**	**2690.41**	3006371	2.5
p02	75	4	140	2	6245.17	6245.17	1.35	**6120.99**	6159.15	37507	6139.78	**6146.51**	1923035	10
p03	100	4	200	3	7208.69	7305.18	1.69	7079.96	7113.85	16799	**7049.27**	**7067.72**	1446278	10
p04	150	4	290	4	12499.03	12774.28	2.23	12428.56	12488.27	16377	**12216.64**	**12285.23**	1390540	20
p05	200	4	400	5	15116.86	15589.62	3.44	15094.85	15219.74	16313	**14692.25**	**14744.47**	1208380	40
p06	288	5	648	6	20437.29	20658.37	8.27	20494.48	20629.18	10873	**19810.44**	**19915.31**	1197720	60
p07	350	5	810	8	32592.80	33126.81	8.43	32820.43	33019.79	6579	**31547.90**	**31676.61**	589503	60
p08	500	5	1050	10	101418.52	102329.36	8.15	103498.11	103817.16	3356	**98351.12**	**98734.27**	431832	60
p09	750	6	1750	13	130635.46	131334.72	12.9	132552.19	133206.74	2030	**129220.84**	**129593.30**	103591	60
p10	1000	6	2100	13	118703.09	121479.08	19.74	122165.79	122704.66	460	**117621.84**	**119828.83**	35470	60
Average dev. to BKS (%)						1.34		0.29	0.79		-2.00	**-1.56**		

Table 4.14: Results for the HSDMDMTPVRP instances.

						UHGS			**AVNR**			
Instance	$\|U\|$	$\|T\|$	n_{ts}	$\|D\|$	$\|K\|$	Best	Average	Iterations	Best	Average	Iterations	T(min)
p01	60	2	87	2	5	4870.18	**4876.71**	31221	**4869.78**	4925.71	1895771	5.0
p02	80	3	163	2	5	**9110.49**	**9113.96**	23604	**9110.49**	9234.90	1569455	10.0
p03	120	2	173	2	5	7891.10	**7975.54**	17420	**7875.00**	8064.63	1355003	10.0
p04	80	4	210	2	5	**13519.97**	**13558.22**	37379	13560.13	13632.79	2461616	20.0
p05	99	3	200	3	5	**9815.36**	**9849.66**	35634	9940.17	9975.78	2367039	20.0
p06	75	5	225	2	5	**12428.14**	**12452.22**	42876	12514.54	12612.70	2542886	20.0
p07	95	5	218	2	5	9573.01	9636.16	19650	**9297.00**	**9436.59**	1302913	20.0
p08	160	2	241	2	5	13911.14	13972.04	16192	**13674.72**	**13800.89**	1385730	20.0
p09	150	3	282	2	5	14440.06	14544.85	16524	**14263.66**	**14374.81**	1269747	20.0
p10	140	4	326	2	5	20512.50	20651.22	18593	**20154.65**	**20263.02**	1615274	30.0
p11	250	2	370	3	5	16934.79	16988.92	8842	**16624.89**	**16765.58**	748821	30.0
p12	198	5	370	3	5	20764.34	21037.60	9497	**20174.23**	**20446.11**	728591	30.0
p13	201	3	395	2	5	17931.25	17997.60	15069	**17656.56**	**17792.77**	1308254	40.0
p14	200	4	489	3	5	28016.34	28181.26	13294	**27401.06**	**27835.35**	1193506	50.0
p15	270	4	668	4	5	26532.58	26703.89	9638	**26161.49**	**26458.54**	968488	60.0
p16	293	6	957	4	5	41705.84	41970.52	5686	**40797.98**	**41479.48**	520365	60.0
Average dev. to BKS (%)						1.26	1.79		0.14	**1.25**		

4.5 Final remarks of the chapter

In this chapter, given a real world demand, we designed two metaheuristics to solve the Heterogeneous Site-Dependent Multi-Depot Multi-Trip Periodic Vehicle Routing Problem (HSDMDMTPVRP), which is a combination of multiple well-known vehicle routing problem variants.

The first proposed metaheuristic is an adaptation of the well performing VRP metaheuristic called Unified Hybrid Genetic Search (UHGS) [16]. The second metaheuristic is completely new, shares many search mechanisms with UHGS, and we named it Adaptive Variable Neighborhood Race (AVNR). The results presented in Section 4.4 reveal that both UHGS and AVNR present good results, outperforming several approaches proposed in the literature for the VRP variants. In special, AVNR provided an overall advantage.

Chapter 5

Conclusions, achievements and future works

This work solved three logistics transportation optimization problems: the Traffic Counting Location Problem (TCLP), the Periodic Supply Vessel Planning Problem with Berth-Allocation (PSVPP-BA) and the Heterogeneous Site-Dependent Multi-Depot Multi-Trip Periodic Vehicle Routing Problem (HSDMDMTPVRP).

We proposed a progressive hybrid algorithm based on set covering to solve the TCLP. Greedy, exact and hybrid methods, based on a simple and innovative concept which has not yet been explored in the literature, were developed embedded in a set covering framework to solve 26 real-world instances obtained from the Brazilian states. Our algorithm found better results than the clustering search of GONZÁLEZ *et al.* [1], providing best-known solutions for the hardest instances MG, MT and SP. Regarding the exact methods, the branch-and-cut algorithm of GONZÁLEZ *et al.* [1] solved only four instances to optimality, whereas we were able to find 20 optimal solutions with lower CPU times. Besides, two new optimal solutions were found after evaluation of the results.

We have presented an exact branch-and-cut method and an adaptive large neighborhood search (ALNS) heuristic with multiple starts and spaced local searches to solve the periodic supply vessel planning problem (PSVPP) arising in the upstream offshore petroleum logistics chain. The PSVPP tackled in this work consists of a periodic vehicle routing problem while simultaneously determining the optimal fleet size and a mix of heterogeneous offshore supply vessels, their one week routes and schedules for servicing the offshore oil and gas installations, besides the berth allocations at the supply base. We have extended the previous works of HALVORSEN-WEARE and FAGERHOLT [12], KISIALIOU *et al.* [11] and CRUZ *et al.* [2] since we used a replicable one-week planning horizon both for the offshore units and the vessels. We solved the largest available real-world instances, without dividing them into clusters, and we achieved good solutions relatively fast, performing significantly better than the branch-and-cut algorithm.

Finally, given a real world demand, we designed two metaheuristics to solve a Heterogeneous Site-Dependent Multi-Depot Multi-Trip Periodic Vehicle Routing Problem (HSDMDMTPVRP), that is a combination of multiple well-known Vehicle Routing Problem (VRP) variants. The first metaheuristic is an adaptation of the Unified Hybrid Genetic Search (UHGS) proposed by VIDAL *et al.* [16] which provides good results for the VRP variants. Based on that, we propose the Adaptive Variable Neighborhood Race (AVNR) metaheuristic, which shares many search mechanisms with our adaptation for the UGHS. AVNR has provided good results even for those VRP variants well-explored in the literature.

Achievements

Concerning the PSVPP-BA, during my PhD Visit at the Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), under the supervision of Prof. Gilbert Laporte, we published a scientific paper at the European Journal of Operations Research (EJOR) [39].

We won a conference prize of "ANPET Prize of Scientific Production 2017" for the paper called "A multi-objective assessment of emergency care based on population, number of occurrences and distance traveled by rescue vehicles" [97].

Regarding the TCLP, we won a conference prize of "ANPET Prize of Scientific Production 2019" and published a scientific article at the journal Expert Systems with Applications (ESWA) [18].

For the HSDMDMTPVRP, a scientific paper is under production for submission to a high quality journal.

Future works

Regarding the TCPL, we suggest to solve the case that uses partial coverage to maximize the total number of O-D pairs covered, when considering a pre-established budget for the sensors. New heuristics and metaheuristics could be proposed to generate Pareto curves to analyze the trade-off between maximizing the coverage of the traffic network and minimizing the cost of installing the sensors.

The PSVPP's algorithms can also be adapted to other PSVPP variations, tested on more real-world instances, and can be modified to tackle variations which include weather conditions or other uncertainties like delayed services.

For the HSDMDMTPVRP, we can adapt the AVNR to other routing or optimization problems in general. It also can be easily adapted to run on multiple threads or multiple nodes in clusters, like the work of CORDEAU and MAISCHBERGER [86], since each solution search per step is independent, only requiring a single thread between steps.

Acknowledgements

This work was conducted during financial support of the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES) (grant 88882.425238/2019-01). This work was partially supported by the National Council for Scientific and Technological Development - CNPq, under grants #309661/2019-6 and #307835/2017-0, and by Carlos Chagas Filho Research Support Foundation – FAPERJ, grant #233926. The PSVPP work was conducted during a scholarship supported by the International Cooperation Program CAPES/COFECUB at the Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and financed by CAPES (grant #88881.186960/2018-01) - Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Ministry of Education of Brazil. The support of the Canadian Natural Sciences and Engineering Research Council (grant #2015-06189) is gratefully acknowledged.

References

[1] GONZÁLEZ, P. H., CLÍMACO, G., MAURI, G. R., et al. "New approaches for the traffic counting location problem", *Expert Systems with Applications*, v. 132, pp. 189–198, 2019. ISSN: 0957-4174. doi: https://doi.org/10.1016/j.eswa.2019.04.068.

[2] CRUZ, R., BERGSTEN MENDES, A., BAHIENSE, L., et al. "Integrating berth allocation decisions in a fleet composition and periodic routing problem of platform supply vessels", *European Journal of Operational Research*, v. 275, n. 1, pp. 334–346, 2019. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2018.11.030. Available in: <`https://www.sciencedirect.com/science/article/pii/S037722171830955X`>.

[3] CNT. "Modais de Cargas, Passageiros, Aquaviários, Ferroviários e Autônomos", *Pesquisas, Confederação Nacional dos Transportes*, 2002. Available in: <`https://atlas.cnt.org.br/.`>.

[4] GOMES, G. *Análise comparativa das características de tráfego calculadas pelo HCM-2010 e pelo microssimulador Aimsun para uma área de estudo em Santa Catarina.* Tese de Doutorado, Universidade Federal de Santa Catarina, Santa Catarina, Brazil, 2015. Available in: <`https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/160628/337905.pdf?sequence=1&isAllowed=y`>.

[5] FEI, X., MAHMASSANI, H. S., MURRAY-TUITE, P. "Vehicular network sensor placement optimization under uncertainty", *Transportation Research Part C: Emerging Technologies*, v. 29, pp. 14–31, 2013. DOI doi.org/10.1016/j.trc.2013.01.004.

[6] GARBER, N. J., HOEL, L. A. *Traffic and highway engineering.* Canada, Cengage Learning, Canada, 2014. pp. 1248, ISBN 978-0-495-08250-7.

[7] WANG, N. *Locating Counting Sensors in Traffic Network to Estimate Origin-Destination Volume.* Tese de Doutorado, Arizona State University, Ari-

zona, USA, 2013. Available in: <`https://pdfs.semanticscholar.org/1e88/93911d916e672bcb36fa682635d01aa05dc6.pdf`>.

[8] GENTILI, M., MIRCHANDANI, P. "Locating sensors on traffic networks: Models, challenges and research opportunities", *Transportation research part C: emerging technologies*, v. 24, pp. 227–255, 2012. DOI doi.org/10.1016/j.trc.2012.01.004.

[9] GRIBKOVSKAIA, I., LAPORTE, G., SHLOPAK, A. "A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms", *Journal of the Operational Research Society*, v. 59, n. 11, pp. 1449–1459, 2008. doi: doi.org/10.1057/palgrave.jors.2602469.

[10] AAS, B., HALSKAU SR, Ø., WALLACE, S. W. "The role of supply vessels in offshore logistics", *Maritime Economics & Logistics*, v. 11, n. 3, pp. 302–325, Sep 2009. ISSN: 1479-294X. doi: 10.1057/mel.2009.7.

[11] KISIALIOU, Y., GRIBKOVSKAIA, I., LAPORTE, G. "The periodic supply vessel planning problem with flexible departure times and coupled vessels", *Computers & Operations Research*, v. 94, pp. 52–64, 2018. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2018.02.008.

[12] HALVORSEN-WEARE, E. E., FAGERHOLT, K. "Optimization in offshore supply vessel planning", *Optimization and Engineering*, v. 18, n. 1, pp. 317–341, Mar 2017. ISSN: 1573-2924. doi: 10.1007/s11081-016-9315-4.

[13] DANTZIG, G. B., RAMSER, J. H. "The Truck Dispatching Problem", *Management Science*, v. 6, n. 1, pp. 80–91, 1959. doi: 10.1287/mnsc.6.1.80.

[14] TOTH, P., VIGO, D. *Vehicle Routing: Problems, Methods, and Applications, Second Edition.* MOS-SIAM Series on Optimization. Philadelphia, Pennsylvania, United States, SIAM, 2014. ISBN: 9781611973594. doi: https://doi.org/10.1137/1.9781611973594.

[15] KOÇ, Ç., BEKTAŞ, T., JABALI, O., et al. "Thirty years of heterogeneous vehicle routing", *European Journal of Operational Research*, v. 249, n. 1, pp. 1–21, 2016. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.07.020.

[16] VIDAL, T., CRAINIC, T. G., GENDREAU, M., et al. "A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems", *Operations Research*, v. 60, n. 3, pp. 611–624, 2012. doi: 10.1287/opre.1120.1048.

[17] YANG, H., YANG, C., GAN, L. "Models and algorithms for the screen line-based traffic-counting location problems", *Computers & operations research*, v. 33, n. 3, pp. 836–858, 2006. DOI doi.org/10.1016/j.cor.2004.08.011.

[18] VIEIRA, B. S., FERRARI, T., RIBEIRO, G. M., et al. "A progressive hybrid set covering based algorithm for the traffic counting location problem", *Expert Systems with Applications*, v. 160, pp. 113641, 2020. ISSN: 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113641.

[19] HODGSON, M. J. "A Flow-Capturing Location-Allocation Model", *Geographical Analysis*, v. 22, n. 3, pp. 270–279, 1990. DOI doi.org/10.1111/j.1538-4632.1990.tb00210.x.

[20] KIM, J.-G., KUBY, M. "The deviation-flow refueling location model for optimizing a network of refueling stations", *International Journal of Hydrogen Energy*, v. 37, n. 6, pp. 5406 – 5420, 2012. ISSN: 0360-3199. DOI doi.org/10.1016/j.ijhydene.2011.08.108.

[21] ARSLAN, O., JABALI, O., LAPORTE, G. "Exact Solution of the Evasive Flow Capturing Problem", *Operations Research*, v. 66, n. 6, pp. 1625–1640, 2018. DOI doi.org/10.1287/opre.2018.1756.

[22] FISK, C. "On combining maximum entropy trip matrix estimation with user optimal assignment", *Transportation Research Part B: Methodological*, v. 22, n. 1, pp. 69–73, 1988. DOI doi.org/10.1016/0191-2615(88)90035-5.

[23] MOHAMAD, D., SINHA, K., KUCZEK, T., et al. "Annual average daily traffic prediction model for county roads", *Transportation Research Record: Journal of the Transportation Research Board*, v. 1, n. 1617, pp. 69–77, 1998. DOI doi.org/10.3141/1617-10.

[24] ZHONG, M., LINGRAS, P., SHARMA, S. "Estimation of missing traffic counts using factor, genetic, neural, and regression techniques", *Transportation Research Part C*, v. 12, n. 2, pp. 139–166, 2004. DOI doi.org/10.1016/j.trc.2004.07.006.

[25] WANG, N., GENTILI, M., MIRCHANDANI, P. "Model to locate sensors for estimation of static origin-destination volumes given prior flow information", *Transportation Research Record: Journal of the Transportation Research Board*, v. 1, n. 2283, pp. 67–73, 2012. DOI doi.org/10.3141/2283-07.

[26] LARSSON, T., LUNDGREN, J. T., PETERSON, A. "Allocation of Link Flow Detectors for Origin-Destination Matrix Estimation—A Comparative Study", *Computer-Aided Civil and Infrastructure Engineering*, v. 25, n. 2, pp. 116–131, 2010. DOI doi.org/10.1111/j.1467-8667.2009.00625.x.

[27] YANG, H., IIDA, Y., SASAKI, T. "An analysis of the reliability of an origin-destination trip matrix estimated from traffic counts", *Transportation Research Part B: Methodological*, v. 25, n. 5, pp. 351–363, 1991. DOI doi.org/10.1016/0191-2615(91)90028-H.

[28] YANG, H., ZHOU, J. "Optimal traffic counting locations for origin–destination matrix estimation", *Transportation Research Part B: Methodological*, v. 32, n. 2, pp. 109–126, 1998. DOI doi.org/10.1016/S0191-2615(97)00016-7.

[29] YANG, H., GAN, L., TANG, W. "Determining cordons and screen lines for origin-destination trip studies", *Proceedings of the Eastern Asia Society for Transportation Studies*, v. 3, n. 2, pp. 85–99, 2001. Available in: <`http://hdl.handle.net/1783.1/46043`>.

[30] CHOOTINAN, P., CHEN, A., YANG, H. "A bi-objective traffic counting location problem for origin-destination trip table estimation", *Transportmetrica*, v. 1, n. 1, pp. 65–80, 2005. DOI doi.org/10.1080/18128600508685639.

[31] CHEN, A., PRAVINVONGVUTH, S., CHOOTINAN, P., et al. "Strategies for selecting additional traffic counts for improving OD trip table estimation", *Transportmetrica*, v. 3, n. 3, pp. 191–211, 2007. DOI doi.org/10.1080/18128600708685673.

[32] YELBAY, B., BIRBIL, Ş. İ., BÜLBÜL, K. "The set covering problem revisited: An empirical study of the value of dual information", *Journal of Industrial & Management Optimization*, v. 11, n. 2, pp. 575–594, 2015. DOI doi.org/10.3934/jimo.2015.11.575.

[33] CAPRARA, A., FISCHETTI, M., TOTH, P. "A heuristic method for the set covering problem", *Operations research*, v. 47, n. 5, pp. 730–743, 1999. DOI doi.org/10.1287/opre.47.5.730.

[34] KNUTH, D. E. "A generalization of Dijkstra's algorithm", *Information Processing Letters*, v. 6, n. 1, pp. 1–5, 1977. DOI doi.org/10.1016/0020-0190(77)90002-3.

[35] YANG, C., CHOOTINAN, P., CHEN, A. “Traffic counting location planning using genetic algorithm”, *Journal of the Eastern Asia Society for Transportation Studies*, v. 5, pp. 898–913, 2003.

[36] NEMHAUSER, G. L., WOLSEY, L. A. *Integer programming and combinatorial optimization*, v. 20. New York, NY, Springer, 1988. ISBN: 9781118627372. doi: doi.org/10.1002/9781118627372.

[37] BARNHART, C., JOHNSON, E. L., NEMHAUSER, G. L., et al. “Formulating a Mixed Integer Programming Problem to Improve Solvability”, *Operations Research*, v. 41, n. 6, pp. 1013–1019, 1993. DOI doi.org/10.1287/opre.41.6.1013.

[38] SHERALI, H. D., DRISCOLL, P. J. “Evolution and state-of-the-art in integer programming”, *Journal of Computational and Applied Mathematics*, v. 124, n. 1, pp. 319 – 340, 2000. DOI doi.org/10.1016/S0377-0427(00)00431-3.

[39] VIEIRA, B. S., RIBEIRO, G. M., BAHIENSE, L., et al. “Exact and heuristic algorithms for the fleet composition and periodic routing problem of offshore supply vessels with berth allocation decisions”, *European Journal of Operational Research*, 2021. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.03.022.

[40] BAPTISTA, S., OLIVEIRA, R. C., ZÚQUETE, E. “A period vehicle routing case study”, *European Journal of Operational Research*, v. 139, n. 2, pp. 220–229, 2002. ISSN: 0377-2217. doi: https://doi.org/10.1016/S0377-2217(01)00363-0. EURO XVI: O.R. for Innovation and Quality of Life.

[41] BALDACCI, R., BARTOLINI, E., MINGOZZI, A., et al. “An Exact Algorithm for the Period Routing Problem”, *Operations Research*, v. 59, n. 1, pp. 228–241, 2011. doi: 10.1287/opre.1100.0875.

[42] FAGERHOLT, K., LINDSTAD, H. “Optimal policies for maintaining a supply service in the Norwegian Sea”, *Omega*, v. 28, n. 3, pp. 269–275, 2000. ISSN: 0305-0483. doi: https://doi.org/10.1016/S0305-0483(99)00054-7.

[43] AAS, B., GRIBKOVSKAIA, I., HALSKAU, Ø., et al. “Routing of supply vessels to petroleum installations”, *International Journal of Physical Distribution & Logistics Management*, v. 37, n. 2, pp. 164–179, Jan 2007. ISSN: 0960-0035. doi: 10.1108/09600030710734866.

[44] HALVORSEN-WEARE, E. E., FAGERHOLT, K., NONÅS, L. M., et al. "Optimal fleet composition and periodic routing of offshore supply vessels", *European Journal of Operational Research*, v. 223, n. 2, pp. 508–517, 2012. doi: doi.org/10.1016/j.ejor.2012.06.017.

[45] SHYSHOU, A., GRIBKOVSKAIA, I., LAPORTE, G., et al. "A Large Neighbourhood Search Heuristic for a Periodic Supply Vessel Planning Problem Arising in Offshore Oil and Gas Operations", *INFOR: Information Systems and Operational Research*, v. 50, n. 4, pp. 195–204, 2012. doi: 10.3138/infor.50.4.195.

[46] FERNÁNDEZ CUESTA, E., ANDERSSON, H., FAGERHOLT, K., et al. "Vessel routing with pickups and deliveries: An application to the supply of offshore oil platforms", *Computers & Operations Research*, v. 79, pp. 140–147, 2017. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2016.10.014.

[47] BORTHEN, T., LOENNECHEN, H., WANG, X., et al. "A genetic search-based heuristic for a fleet size and periodic routing problem with application to offshore supply planning", *EURO Journal on Transportation and Logistics*, v. 7, n. 2, pp. 121–150, 2018. ISSN: 2192-4376. doi: https://doi.org/10.1007/s13676-017-0111-x.

[48] BORTHEN, T., LOENNECHEN, H., FAGERHOLT, K., et al. "Bi-objective offshore supply vessel planning with costs and persistence objectives", *Computers & Operations Research*, v. 111, pp. 285–296, 2019. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2019.06.014.

[49] BIERWIRTH, C., MEISEL, F. "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals", *European Journal of Operational Research*, v. 244, n. 3, pp. 675–689, 2015. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2014.12.030.

[50] MAURI, G. R., RIBEIRO, G. M., LORENA, L. A. N., et al. "An adaptive large neighborhood search for the discrete and continuous berth allocation problem", *Computers & Operations Research*, v. 70, pp. 140–154, 2016.

[51] ÇAĞATAY IRIS, PACINO, D., ROPKE, S. "Improved formulations and an Adaptive Large Neighborhood Search heuristic for the integrated berth allocation and quay crane assignment problem", *Transportation Research Part E: Logistics and Transportation Review*, v. 105, pp. 123–147, 2017. ISSN: 1366-5545. doi: https://doi.org/10.1016/j.tre.2017.06.013.

[52] KISIALIOU, Y., GRIBKOVSKAIA, I., LAPORTE, G. “Robust supply vessel routing and scheduling”, *Transportation Research Part C: Emerging Technologies*, v. 90, pp. 366–378, 2018. ISSN: 0968-090X. doi: https://doi.org/10.1016/j.trc.2018.03.012.

[53] APPLEGATE, D. L., BIXBY, R. E., CHVÁTAL, V., et al. *The Traveling Salesman Problem*. Princeton, New Jersey, Princeton University Press, 2011. ISBN: 9780691129938. doi: doi:10.1515/9781400841103. Available in: <`https://doi.org/10.1515/9781400841103`>.

[54] CODATO, G., FISCHETTI, M. “Combinatorial Benders’ Cuts for Mixed-Integer Linear Programming”, *Operations Research*, v. 54, n. 4, pp. 756–766, 2006. doi: 10.1287/opre.1060.0286.

[55] BONDY, J. A., MURTY, U. S. R. *Graph Theory With Applications*, v. 290. London, Macmillan, 1976. ISBN: 9780333177914.

[56] KARP, R. M. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*, pp. 85–103, Boston, MA, Springer US, 1972. ISBN: 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9.

[57] ROPKE, S., PISINGER, D. “An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows”, *Transportation Science*, v. 40, n. 4, pp. 455–472, 2006. doi: 10.1287/trsc.1050.0135.

[58] SHAW, P. “Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems”. In: Maher, M., Puget, J.-F. (Eds.), *Principles and Practice of Constraint Programming – CP98*, pp. 417–431, Berlin Heidelberg, 1998. Springer. ISBN: 978-3-540-49481-2.

[59] HWANG, C.-R. “Simulated annealing: Theory and applications”, *Acta Applicandae Mathematica*, v. 12, n. 1, pp. 108–111, May 1988. ISSN: 1572-9036.

[60] BRÄYSY, O., HASLE, G., DULLAERT, W. “A multi-start local search algorithm for the vehicle routing problem with time windows”, *European Journal of Operational Research*, v. 159, n. 3, pp. 586–605, 2004. ISSN: 0377-2217. doi: https://doi.org/10.1016/S0377-2217(03)00435-1. Available in: <`https://www.sciencedirect.com/science/article/pii/S0377221703004351`>.

[61] PALOMO-MARTTÍNEZ, P. J., SALAZAR-AGUILAR, M. A., LAPORTE, G. “Planning a selective delivery schedule through Adaptive Large Neigh-

borhood Search", *Computers & Industrial Engineering*, v. 112, pp. 368–378, 2017. ISSN: 0360-8352. doi: https://doi.org/10.1016/j.cie.2017.08.037. Available in: <`https://www.sciencedirect.com/science/article/pii/S0360835217303960`>.

[62] KOÇ, Ç., JABALI, O., LAPORTE, G. "Long-haul vehicle routing and scheduling with idling options", *Journal of the Operational Research Society*, Apr 2017. ISSN: 1476-9360. doi: 10.1057/s41274-017-0202-y.

[63] ROPKE, S., PISINGER, D. "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls", *European Journal of Operational Research*, v. 171, n. 3, pp. 750–775, 2006. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2004.09.004. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research.

[64] MATTOS RIBEIRO, G., LAPORTE, G. "An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem", *Computers & Operations Research*, v. 39, n. 3, pp. 728–735, 2012. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2011.05.005.

[65] PARASKEVOPOULOS, D. C., REPOUSSIS, P. P., TARANTILIS, C. D., et al. "A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows", *Journal of Heuristics*, v. 14, n. 5, pp. 425–455, Oct 2008. ISSN: 1572-9397. doi: 10.1007/s10732-007-9045-z.

[66] PISINGER, D., ROPKE, S. "A general heuristic for vehicle routing problems", *Computers & Operations Research*, v. 34, n. 8, pp. 2403–2435, 2007. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2005.09.012.

[67] BELTRAMI, E. J., BODIN, L. D. "Networks and vehicle routing for municipal waste collection", *Networks*, v. 4, n. 1, pp. 65–94, 1974.

[68] CAMPBELL, A. M., WILSON, J. H. "Forty years of periodic vehicle routing", *Networks*, v. 63, n. 1, pp. 2–15, 2014. doi: https://doi.org/10.1002/net.21527.

[69] CORDEAU, J.-F., LAPORTE, G. "A Tabu Search Algorithm For The Site Dependent Vehicle Routing Problem With Time Windows", *Information Systems and Operational Research*, v. 39, n. 3, pp. 292–298, 2001. doi: 10.1080/03155986.2001.11732443.

[70] CORDEAU, J.-F., GENDREAU, M., LAPORTE, G. "A tabu search heuristic for periodic and multi-depot vehicle routing problems", *Networks*, v. 30, n. 2, pp. 105–119, 1997. doi: https://doi.org/10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G.

[71] GOLDEN, B., ASSAD, A., LEVY, L., et al. "The fleet size and mix vehicle routing problem", *Computers & Operations Research*, v. 11, n. 1, pp. 49–66, 1984. ISSN: 0305-0548. doi: https://doi.org/10.1016/0305-0548(84)90007-8.

[72] TAILLARD, É. D., LAPORTE, G., GENDREAU, M. "Vehicle Routeing with Multiple Use of Vehicles", *Journal of the Operational Research Society*, v. 47, n. 8, pp. 1065–1070, 1996. doi: 10.1057/jors.1996.133.

[73] BRANDÃO, J. C. S., MERCER, A. "The multi-trip vehicle routing problem", *Journal of the Operational Research Society*, v. 49, n. 8, pp. 799–805, Aug 1998. ISSN: 1476-9360. doi: 10.1057/palgrave.jors.2600595. Available in: <`https://doi.org/10.1057/palgrave.jors.2600595`>.

[74] COELHO, V., GRASAS, A., RAMALHINHO, H., et al. "An ILS-based algorithm to solve a large-scale real heterogeneous fleet VRP with multi-trips and docking constraints", *European Journal of Operational Research*, v. 250, n. 2, pp. 367–376, 2016. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.09.047.

[75] ALONSO, F., ALVAREZ, M. J., BEASLEY, J. E. "A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions", *Journal of the Operational Research Society*, v. 59, n. 7, pp. 963–976, Jul 2008. ISSN: 1476-9360. doi: 10.1057/palgrave.jors.2602405.

[76] VIDAL, T., CRAINIC, T. G., GENDREAU, M., et al. "A unified solution framework for multi-attribute vehicle routing problems", *European Journal of Operational Research*, v. 234, n. 3, pp. 658–673, 2014. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2013.09.045.

[77] PRINS, C. "A simple and effective evolutionary algorithm for the vehicle routing problem", *Computers & Operations Research*, v. 31, n. 12, pp. 1985–2002, 2004. ISSN: 0305-0548. doi: https://doi.org/10.1016/S0305-0548(03)00158-8.

[78] SAMPSON, J. R. "Adaptation in Natural and Artificial Systems (John H. Holland)", *SIAM Review*, v. 18, n. 3, pp. 529–530, 1976. doi: 10.1137/1018105.

[79] LÓPEZ-IBÁÑEZ, M., DUBOIS-LACOSTE, J., PÉREZ CÁCERES, L., et al. "The irace package: Iterated racing for automatic algorithm configuration", *Operations Research Perspectives*, v. 3, pp. 43–58, 2016. ISSN: 2214-7160. doi: https://doi.org/10.1016/j.orp.2016.09.002.

[80] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P. "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations", *Mathematical Programming*, v. 20, n. 1, pp. 255–282, Dec 1981. ISSN: 1436-4646. doi: 10.1007/BF01589353.

[81] GOLDEN, B. L., WASIL, E. A., KELLY, J. P., et al. "The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results". In: *Fleet Management and Logistics*, pp. 33–56, Boston, MA, Springer US, 1998. ISBN: 978-1-4615-5755-5. doi: 10.1007/978-1-4615-5755-5_2.

[82] NAGATA, Y., BRÄYSY, O. "A powerful route minimization heuristic for the vehicle routing problem with time windows", *Operations Research Letters*, v. 37, n. 5, pp. 333–338, 2009. ISSN: 0167-6377. doi: https://doi.org/10.1016/j.orl.2009.04.006.

[83] NAG, B., GOLDEN, B. L., ASSAD, A. "Vehicle routing with site dependencies", *Vehicle routing: Methods and studies*, pp. 149–159, 1988.

[84] CHAO, I.-M., GOLDEN, B. L., WASIL, E. A. "A New Algorithm for the Site-Dependent Vehicle Routing Problem". In: *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*, cap. 8, pp. 301–312, Boston, MA, Springer US, 1998. ISBN: 978-1-4757-2807-1. doi: 10.1007/978-1-4757-2807-1_12.

[85] PESSOA, A., SADYKOV, R., UCHOA, E. "Enhanced Branch-Cut-and-Price algorithm for heterogeneous fleet vehicle routing problems", *European Journal of Operational Research*, v. 270, n. 2, pp. 530–543, 2018. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2018.04.009.

[86] CORDEAU, J.-F., MAISCHBERGER, M. "A parallel iterated tabu search heuristic for vehicle routing problems", *Computers & Operations Research*,

v. 39, n. 9, pp. 2033–2050, 2012. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2011.09.021.

[87] MINGOZZI, A., ROBERTI, R., TOTH, P. "An Exact Algorithm for the Multi-trip Vehicle Routing Problem", *INFORMS Journal on Computing*, v. 25, n. 2, pp. 193–207, 2013. doi: 10.1287/ijoc.1110.0495.

[88] CATTARUZZA, D., ABSI, N., FEILLET, D., et al. "A memetic algorithm for the Multi Trip Vehicle Routing Problem", *European Journal of Operational Research*, v. 236, n. 3, pp. 833–848, 2014. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2013.06.012. Vehicle Routing and Distribution Logistics.

[89] TAILLARD, E. D. "A heuristic column generation method for the heterogeneous fleet VRP", *RAIRO - Operations Research - Recherche Opérationnelle*, v. 33, n. 1, pp. 1–14, 1999. Available in: <`http://www.numdam.org/item/RO_1999__33_1_1_0/`>.

[90] PENNA, P. H. V., SUBRAMANIAN, A., OCHI, L. S. "An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem", *Journal of Heuristics*, v. 19, n. 2, pp. 201–232, Apr 2013. ISSN: 1572-9397. doi: 10.1007/s10732-011-9186-y.

[91] DUHAMEL, C., LACOMME, P., PRODHON, C. "A hybrid evolutionary local search with depth first search split procedure for the heterogeneous vehicle routing problems", *Engineering Applications of Artificial Intelligence*, v. 25, n. 2, pp. 345–358, 2012. ISSN: 0952-1976. doi: https://doi.org/10.1016/j.engappai.2011.10.002. Special Section: Local Search Algorithms for Real-World Scheduling and Planning.

[92] CHRISTOFIDES, N., BEASLEY, J. E. "The period routing problem", *Networks*, v. 14, n. 2, pp. 237–256, 1984. doi: https://doi.org/10.1002/net.3230140205.

[93] RUSSELL, R., IGO, W. "An assignment routing problem", *Networks*, v. 9, n. 1, pp. 1–17, 1979. doi: https://doi.org/10.1002/net.3230090102.

[94] RUSSELL, R. A., GRIBBIN, D. "A multiphase approach to the period routing problem", *Networks*, v. 21, n. 7, pp. 747–765, 1991. doi: https://doi.org/10.1002/net.3230210704.

[95] CHAO, I.-M., GOLDEN, B. L., WASIL, E. "An improved heuristic for the period vehicle routing problem", *Networks*, v. 26, n. 1, pp. 25–44, 1995. doi: https://doi.org/10.1002/net.3230260104.

[96] CHRISTIAENS, J., VANDEN BERGHE, G. "Slack Induction by String Removals for Vehicle Routing Problems", *Transportation Science*, v. 54, n. 2, pp. 417–433, 2020. doi: 10.1287/trsc.2019.0914.

[97] FERRARI, T., VIEIRA, B. S., CAMARA, M. V. O., et al. "Uma avaliação multiobjetivo de atendimentos de emergência com base na população, no número de ocorrências e na distância percorrida pelos veículos de resgate", *TRANSPORTES*, v. 26, n. 3, pp. 145–158, nov. 2018. doi: 10.14295/transportes.v26i3.1643.

Appendix A

Detailed parameters tuning for the ALNS' PSVPP-BA

According to Section 3.5.2, this appendix presents all details about the tuning phase. So, Tables A.1 to A.7 show the average percentage deviation (Dev (%)) of the solution values from the initial set of parameters caused by the variation of these parameters.

A.1 Impact of each operator

In order to validate the performance of each operator, Table A.1 presents the tuning results for the ALNS without each one of its heuristics. Between the removal heuristics, the ones with higher impact was the Shaw frequency removal (RO3) and Worst removal (RO5) operators and looking to the insertion operators, the ones with higher impact was the Deep greedy insertion (IO1) and Greed insertion (IO2) operators. As the removal of any operator implies a lower algorithmic performance, all proposed operators were kept.

Table A.1: Average percentage solution deviation without each operator compared with no operator removed (Base).

Operator	RO1	RO2	RO3	RO4	RO5	IO1	IO2	IO3	Base
Dev (%)	0.70	0.79	1.74	0.53	1.86	3.10	2.84	1.33	0.00

A.2 Single tested set of parameters

Table A.2 presents the tuning results for the single tested ALNS' parameters segment size ρ, removal level of determinism p, local search interval δ, reaction factor τ_A and adaptive scores $(\sigma_1, \sigma_2, \sigma_3)$. The segment size parameter ρ controls the frequency of updates of the adaptive weights and the penalty weight. It was tested for the values 25, 50, 100, 200 and 300, and the best result was reached at the initial value $\rho = 50$. The removal level of the determinism parameter p was tested in a range from totally random $p = 1$ to extremely deterministic $p = 10$, passing through the values $p = 2$, $p = 4$ and $p = 6$. The worst results were observed at both extremes $p = 1$ and $p = 10$, and the best result was achieved with the initial value $p = 4$. The local search interval parameter δ was tested for the values 10, 25, 50, 200 and 500. The best result was reached at the initial value $\delta = 50$, and both the most spaced and least spaced search intervals had a negative impact at the end of the ALNS search. In addition to the average percentage deviation on the objective function caused by the variation of δ, Table A.2 shows the average CPU times in seconds, spent in the computation considering each of its tested values. Taking into account the behavior of the CPU times, it is expected that more frequent executions of local searches (what is computationally expensive) will increase the total search time, but this was not the case. This behavior is justified by the fact that we have applied the local search to the most promising solution instead of the current one. We tested the impact of having no local searches as well ($\delta = \infty$) and we obtained in average percentage solution deviation of 4.215% within approximately the same CPU time.

Table A.2: Average percentage solution deviation with varying ρ, p, τ_A and δ.

ρ	Values	25	**50**	100	200	300
	Dev (%)	2.39	**0.000**	2.42	2.55	3.15
p	Values	1	2	**4**	6	10
	Dev (%)	1.05	0.93	**0.000**	2.45	2.87
δ	Values	10	25	**50**	200	500
	Dev (%)	2.51	0.96	**0.000**	2.08	3.38
	CPU(s)	51.1	46.7	47.5	48.0	49.2
τ_A	Values	0.01	**0.05**	0.10	0.20	0.30
	Dev (%)	0.284	**-0.131**	0.000	0.261	0.142
$(\sigma_1, \sigma_2, \sigma_3)$	Values	(0,10,5)	(10,0,5)	(10,5,0)	**(5,0,10)**	(0,5,10)
	Dev (%)	0.384	0.415	0.000	**-0.195**	0.541

The adaptive parameters have a marginal overall impact. The reaction factor parameter τ_A was tested for the values $0.01, 0.05, 0.10, 0.20$ and 0.30, and the best performance was obtained for $\tau_A = 0.05$. The adaptive scores $(\sigma_1, \sigma_2, \sigma_3)$ were tested for the values $(0, 10, 5)$, $(10, 0, 5)$, $(10, 5, 0)$, $(5, 0, 10)$ and $(0, 5, 10)$, and the best performance was achieved for $(5, 0, 10)$, which may not be in accordance with the initial proposition of [57], but is in line with the recent works of [62] and [11].

A.3 Feasibility weights updating

Table A.3 presents the tuning results for the pair of parameters (τ^-, τ^+). The infeasible update weight τ^- was tested for the values 1.05, 1.10, 1.15 and 1.25, while the feasible update weight τ^+ was tested for the values 0.60, 0.75, 0.90, and 1.00.

The best result was achieved for the pair of initial values ($\tau^- = 1.15, \tau^+ = 0.90$). It is interesting to emphasize that the last column $\tau^+ = 1.00$ exhibits the worst results for all tested τ^- values. This means that once the best solution is feasible, the penalty weights do not decrease, discouraging the exploration of infeasible solutions.

Table A.3: Average solution deviation (%) with varying feasibility weights update values.

$\tau^- \setminus \tau^+$	0.60	0.75	**0.90**	1.00
1.05	1.39	0.49	0.17	2.93
1.10	1.16	0.66	0.07	2.90
1.15	1.27	0.07	**0.00**	2.07
1.25	1.50	1.40	0.27	2.88

A.4 Removal range

Table A.4 presents the tuning results for the pair of parameters (n_1^-, n_2^-). Besides the average percentage deviation of the solution values caused by the variation of these parameters, Table A.4 shows the average CPU times spent in the computation, considering each pair of values. The lower removal rate n_1^- was tested for the values 0.05, 0.10, 0.15 and 0.20, while the upper removal rate n_2^- was tested for the values 0.20, 0.25, 0.30, and 0.40. The best result was achieved for the values $n_1^- = 0.10$ and $n_2^- = 0.25$. The average CPU times behaved as expected since a larger number of removals and insertions per iteration implied higher CPU times.

Table A.4: Average solution deviation (%) and CPU time, in seconds, with varying removal range.

	Dev (%)	CPU(s)	Dev (%)	CPU(s)	Dev (%)	CPU(s)	Dev (%)	CPU(s)
$n_1^- \setminus n_2^-$	0.20		**0.25**		0.30		0.40	
0.05	0.99	32.4	0.01	36.7	0.27	41.7	0.37	52.8
0.10	-0.12	36.7	**-0.55**	41.5	0.00	46.7	0.43	57.7
0.15	0.53	41.8	0.24	46.7	1.03	51.9	-0.03	62.3
0.20	0.34	48.0	-0.13	52.4	0.30	57.5	0.51	68.0

A.5 Temperature range

Table A.5 presents the tuning results for the pair of parameters (θ'_0, θ'_F) with the average CPU times, spent in the computation considering each pair of values. The initial relative temperature θ'_0 was tested for the values 1, 1/3, 1/10 and 1/50, while the final relative temperature θ'_F was tested for the values 1/50, 1/100, 1/300, and 1/500. The best result was achieved for the pair of values ($\theta'_0 = 1/50, \theta'_F = 1/500$), with an improvement of 1.68% in comparison to baseline tests. This pair of parameters does not have a major impact on the quality of the solutions, but the CPU times grow from the lower right corner (lower temperature values) to the upper right corner (higher temperature values).

Table A.5: Average solution deviation (%) and CPU time, in seconds, with varying temperature range.

	Dev (%)	CPU(s)	Dev (%)	CPU(s)	Dev (%)	CPU(s)	Dev (%)	CPU(s)
$\theta'_0 \setminus \theta'_F$	1/50		1/100		1/300		**1/500**	
1	1.02	48.9	1.39	47.9	0.14	47.2	0.03	47.4
1/3	1.05	48.5	1.41	47.8	0.00	46.7	-0.26	46.3
1/10	0.30	47.3	0.43	46.9	-0.01	45.4	-0.76	45.2
1/50	0.73	45.5	-0.41	44.6	0.43	44.2	**-1.68**	43.6

A.6 Multi-start performance

Tables A.6 and A.7 present the average percentage deviation of the solution values related to the best average solution found for all tested combination of α, β and γ. Starting with Table A.6 we relate the number of starts α and the number of iterations per start β, while keeping the total number of iterations fixed at $\gamma = 150,000$. The parameter α was tested for the values 1, 15, 35, 50, 75 and 100, and β was tested for the values 100, 300, 500, and $1,000$. According to the results of Table A.6, line $\alpha = 1$, i.e. "no-multi-start", yielded the worst results for all values of β, and no other value of α consistently presented better solutions. In addition, discarding the line $\alpha = 1$, columns $\beta = 500$ and $\beta = 1,000$ consistently presented the worst results with the increase of α. Finally, column $\beta = 100$ achieved the best results for all values of α, without, however, showing consistency with the increasing or decreasing of α.

Table A.6: Average solution deviation (%) with $\gamma = 150,000$ and varying α and β.

$\alpha \setminus \beta$	**100**	300	500	1000
1	2.73	1.66	2.14	1.61
15	1.65	0.36	0.91	2.58
35	0.86	0.77	1.31	1.55
50	0.52	0.55	1.30	1.94
75	0.79	1.09	1.10	-
100	0.95	1.01	1.82	-

Based on the results of Table A.6, the number of iterations per start was fixed at $\beta = 100$, and new tests were performed varying the parameters α (number of starts) and γ (total number of iterations). The parameter α was tested for the same values as in Table A.6, and γ was tested for the values $15,000$, $50,000$, $150,000$, and $300,000$. According to the results of Table A.7, the best results are achieved for $\gamma = 300,000$, as the total number of iterations γ increases and once again line

$\alpha = 1$ yielded the worst results for all γ values. Finally, Table A.7 shows that the best result was reached for $\alpha = 50$ $\beta = 100$ and $\gamma = 300,000$.

Table A.7: Average solution deviation (%) and CPU time, in seconds, with $\beta = 100$ and varying α and γ.

$\alpha \setminus \gamma$	15,000	50,000	150,000	**300,000**
1	6.75	3.23	1.43	2.37
15	5.39	3.76	1.65	0.26
35	6.92	2.38	0.86	0.47
50	7.19	1.68	0.52	**0.00**
75	7.47	3.20	0.79	0.38
100	7.93	2.68	1.65	0.43
CPU(s)	10.2	33.6	99.4	201.3

Appendix B

Detailed mechanisms impacts for the AVNR heuristic

According to Section 4.4.1, this appendix details the performance impacts of the removal of each individual mechanism or operator to ensure that they performing as designed after the parameters tuning for the second set of tuning instances.

Table B.1 shows the average percentage deviation (Dev (%)) of the solution values from the final set of parameters caused by the variation of these parameters for each mechanism or operator removed from the search for 10 runs on each instance.

The following mechanisms were removed by: adaptive means $\alpha_A = 0$, diversity means $\epsilon_0^{Elite} = \epsilon_F^{Elite} = 1$, population means $\mu_0^{base} = \mu_F^{base} = n_{steps} = 1$, repair means that Lines 16 and 19 from Algorithm 15 were removed, and local search means that Lines 8−10 from Algorithm 15 were removed, and operator i means that the initial weigh for such operator was set as zero ($\phi_i = 0$).

Looking at the results shown in Table B.1, we can observe that removing any mechanism or operator results in a worst performance for the HSDMDMTPVRP, and that little average improvements in a few cases (negative values) are always marginal (less than 0.09%). So, the mechanisms with the higher impact if removed were Diversity, Population, Repair and Local Search.

Table B.1: Percentile impact of the removal of each search mechanism or operator.

	Instance Type								
Removed Mechanism	CVRP	SDVRP	PVRP	HVRP	MTVRP	MDVRP	SDMTPVRP	HSDMDMTPVRP	Average
Adaptativeness	0.10	0.03	0.19	0.14	0.12	0.10	0.17	0.45	0.16
Diversity	0.02	0.63	0.27	0.74	0.50	0.33	1.43	1.63	0.69
Population	1.45	1.56	1.12	1.25	5.46	1.88	4.64	2.91	2.53
Repair	0.36	0.87	0.45	0.29	2.35	-0.09	1.48	1.99	0.96
Local Search	0.57	0.23	0.42	-0.03	0.98	0.01	1.49	1.35	0.63
Oscilatinng Penalty Weghts	0.07	0.30	0.17	0.01	0.27	0.03	1.07	1.50	0.43
Shaw removal (RO1)	-0.01	0.19	-0.03	0.01	0.35	-0.01	0.03	1.27	0.23
Shaw neighbours removal (RO2)	0.06	0.24	0.02	0.29	0.64	0.06	0.10	2.07	0.44
Shaw load removal (RO3)	-0.08	0.32	0.12	0.19	0.42	0.04	0.10	1.55	0.33
ACUT removal (RO4)	-0.02	0.25	0.05	0.01	0.34	0.01	0.07	0.97	0.21
Worst Removal (RO5)	0.04	0.03	0.11	0.01	0.44	0.16	0.17	2.24	0.40
Random removal (RO6)	0.05	0.35	0.13	0.09	0.47	0.04	-0.08	1.80	0.36
Random Greedy insertion (IO1)	0.18	0.11	0.04	-0.07	0.36	-0.03	0.12	0.05	0.09
Partial greedy insertion (IO2)	0.14	0.13	-0.01	0.03	0.39	0.12	0.09	0.16	0.13
Partial k-regret insertion (IO3)	0.26	-0.02	0.10	0.23	0.37	0.09	-0.05	0.25	0.15