ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

Ingegneria Biomedica, Elettrica e dei Sistemi

Ciclo 36

Settore Concorsuale: 01/A6

Settore Scientifico Disciplinare: MAT/09

Vehicle routing and location routing problems with minimum latency: Algorithms and models

Presentata da: Alan Rodolfo Osorio Mora

Coordinatore Dottorato

Michele Monaci

Supervisore

Daniele Vigo

Co-supervisore

Paolo Toth

Esame finale anno 2024

Abstract

Latency can be defined as the sum of the arrival times at the customers. Minimum latency problems are specially relevant in applications related to humanitarian logistics. This thesis presents algorithms for solving a family of vehicle routing problems with minimum latency.

First the latency location routing problem (LLRP) is considered. It consists of determining the subset of depots to be opened, and the routes that a set of homogeneous capacitated vehicles must perform in order to visit a set of customers such that the sum of the demands of the customers assigned to each vehicle does not exceed the capacity of the vehicle. For solving this problem three metaheuristic algorithms combining simulated annealing and variable neighborhood descent, and an iterated local search (ILS) algorithm, are proposed.

Furthermore, the multi-depot cumulative capacitated vehicle routing problem (MDCCVRP) and the multi-depot k-traveling repairman problem (MDk-TRP) are solved with the proposed ILS algorithm. The MDCCVRP is a special case of the LLRP in which all the depots can be opened, and the MDk-TRP is a special case of the MDCCVRP in which the capacity constraints are relaxed.

Finally, a LLRP with stochastic travel times is studied. A two-stage stochastic programming model and a variable neighborhood search algorithm are proposed for solving the problem. Furthermore a sampling method is developed for tackling instances with an infinite number of scenarios.

Extensive computational experiments show that the proposed methods are effective for solving the problems under study.

Keywords: cumulative routing; location-routing; metaheuristics; stochastic optimization

The present work is dedicated to:
my parents Mary Carmen and Rodolfo,
my sisters Abigail and Vanessa,
my grandparents Irene and Eduardo,
my mentors Paolo, Daniele and Francisco,
my colleagues at the University of Bologna,
my co-authors,
my collaborators from the University of Twente,
and the friends I made in Bologna and Lisbon, who were like my family being away from home.
Thanks for this amazing journey.

Acknowledgments

This work was supported by the National Agency for Research and Development (ANID)/ Scholarship Program/Doctorado Becas Chile/2020-72210275.

Contents

1 Introduction **3**
1.1 Latency routing problems: Concepts and properties . . . 3
1.2 Solution methods . . . 5

2 Effective metaheuristics for the latency location routing problem **11**
2.1 Introduction . . . 11
2.2 Problem statement and literature review . . . 12
2.3 The proposed algorithms . . . 13
2.3.1 Search space . . . 15
2.3.2 Initial solution . . . 16
2.3.3 Local search . . . 17
2.3.4 Variable neighborhood descent strategies . . . 19
2.4 Computational results . . . 20
2.4.1 Parameter tuning . . . 20
2.4.2 Gobal results . . . 21
2.4.3 A comparison with the currently published heuristic algorithms . . . 29
2.4.4 A detailed analysis of the components of the metaheuristics . . . 31
2.4.5 A statistical comparison of the proposed metaheuristics . . . 36
2.5 Conclusions and future research . . . 37

3 An iterated local search algorithm for latency vehicle routing problems with multiple depots **45**
3.1 Introduction . . . 45
3.2 Description of the proposed approach . . . 49
3.2.1 Construction phase: Initial solution . . . 50
3.2.2 Improvement Phase: Iterated local Search algorithm (ILS) . . . 53
3.2.3 Lower bounds . . . 56
3.3 Computational results . . . 59
3.3.1 Parameter tuning . . . 60
3.3.2 An analysis of each ingredient of the M-ILS algorithm . . . 60
3.3.3 The multi-depot cumulative capacitated vehicle routing problem (MDC-CVRP) . . . 64
3.3.4 The multi-depot k-traveling repairman problem . . . 77
3.3.5 The latency location routing problem . . . 81
3.4 Conclusions and future research . . . 88

4 A risk-averse latency location-routing problem with stochastic travel times **95**
4.1 Introduction . . . 95
4.2 Literature review . . . 96
4.2.1 Location routing problems under uncertainty . . . 96
4.2.2 Latency routing problems under uncertainty . . . 103

4.3 Problem description and modeling aspects 104
4.4 Methodology 108
4.4.1 Heuristic algorithm 108
4.4.2 Sampling method 113
4.5 Computational experiments 116
4.5.1 Test instances for the problem 117
4.5.2 Evaluation of the proposed methods 117
4.5.3 The relevance of considering a stochastic modeling framework 121
4.5.4 The effect of considering a risk-averse decision maker 123
4.5.5 Sampling method - tackling instances with continuous probability distributions 126
4.5.6 The deterministic (single-scenario) problem 129
4.6 Conclusions 130

1 Introduction

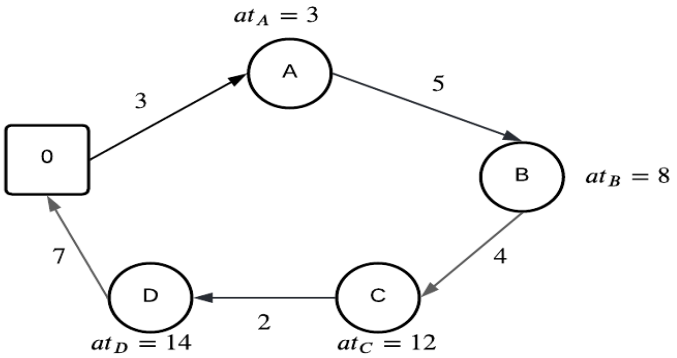
Vehicle routing problems (VRPs) have received a lot of attention from researchers and practitioners due to the many applications and theoretical challenges that these problems represent. They consist of deciding the sequence in which a set of customers is visited by a set of capacitated vehicles that must start from a depot, considering that each customer must be visited exactly once, and the sum of the non-negative demands of the customers associated with each vehicle must not exceed the respective capacity. Despite the family of VRPs is composed by different variants, such as VRPs with time windows, VRPs with simultaneous pickup and delivery, among others (for an overview of VRPs see [1]), we would like to draw your attention to one of the most studied variants of the VRP, which is closely related to all the problems studied in this thesis: the multi-depot vehicle routing problem (MDVRP) [2]. In this version of the problem there is a set of depots from where the vehicles can start the routes to be performed. Several problems in different areas such as logistics, last-mile-delivery, home-health care, and disaster operations management, can be modeled as MDVRPs. Some examples can be the cases in which a company has more than one warehouse from where the products are dispatched, or when more than one shelter is available for supplying affected areas or for receiving people after a natural disaster.

When only a subset of depots can be opened we can talk about a location-routing problem (LRP). The LRP is actually an extension of the MDVRP since the LRP can be reduced to the MDVRP when the maximum number of depots to be opened is equal to the number of available depots. In the LRP not only the location decision must be made, but also the allocation of the vehicles, and the routes that the vehicles must perform. Several reviews regarding LRPs have been published in the last 10 years, the reader is referred to [3], [4], and [5]. From now, when we talk about routing problems, it also considers the LRPs.

In the classical routing problems the objective function is to minimize the global travel time associated with the routes. Despite this approach corresponds to the most widely studied one, several variants have been introduced along the time in order to tackle problems in which the above mentioned approach is not suitable [6]. This is the case for the so-called customer-centric problems, in which customer satisfaction is the key factor. For studying these problems the concept of latency has been usually used as a measure.

1.1 Latency routing problems: Concepts and properties

The latency can be defined as the sum of the arrival times at the customers. The most-simple latency routing problem is the traveling repairman problem (TRP), also known as minimum latency problem, or delivery man problem [7], which is equivalent to the well-known traveling salesman problem, but minimizing the sum of the arrival times at the customers instead of the global travel time.



$global\ travel\ time = c_{0A} + c_{AB} + c_{BC} + c_{CD} + c_{D0} = 3 + 5 + 4 + 2 + 7 = 21$
$latency = at_A + at_B + at_C + at_D = 3 + 8 + 12 + 14 = 37$
or
$latency = 4c_{0A} + 3c_{AB} + 2c_{BC} + 1c_{CD} = 4 * 3 + 3 * 5 + 2 * 4 + 1 * 2 = 37$

Figure 1.1: Differences between the objective functions.

Figure 1.1 presents an example showing the differences between the computation of latency and global travel time of the route of a single vehicle. In the figure, the square represents the depot, the circles represent the customers, above each customer i the respective arrival time (at_i) is presented, and above each edge the associated travel time is presented.

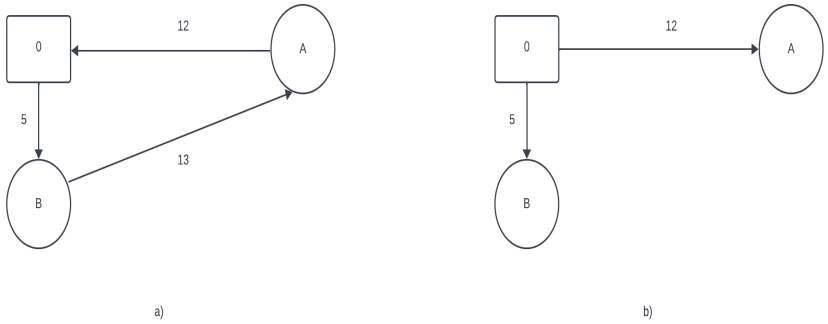


Figure 1.2: Optimal solutions: a) minimizing global travel time, b) minimizing latency.

Latency routing problems have different properties w.r.t. to the classical routing problems. One of the most important property regards the number of vehicles used in the optimal solution. It has been proved that, when the travel times satisfy the triangle inequality, the optimal solution of a latency routing problem uses exactly $\min\{N_c, N_v\}$ vehicles, where N_c is the number of customers, and N_v is the number of available vehicles (see [8] for details). In addition, the last edge of each route, connecting the last customer with the depot, does not affect the objective function, so the latency routing problems can be considered as open routing problems. These properties are not true for the classical routing problems. A trivial example is presented in Figure 1.2 in order to illustrate the differences in the optimal solutions obtained for both objective functions. It considers 2 customers, 2 available vehicles, and one depot.

Due to the above considerations, it is clear that optimal solutions for classical routing problems are sub-optimal solutions for the equivalent latency routing problems as proved empirically in

[9, 10].

For a recent survey paper on latency routing problems the reader is referred to [11].

1.2 Solution methods

When combinatorial optimization problems have to been faced (as is the case in this thesis), three approaches can be adopted: *i*) exact methods, *ii*) approximation methods, and *iii*) heuristic methods.

Despite in an ideal world it is always desirable to obtain the proven optimal solution for the problems through an exact method, it is not always possible, specially in routing problems which have been proved to be NP-hard.

On the other hand; approximation and heuristic algorithms seek for solutions that cannot be proved to be optimal. The difference between the two mentioned methods is that the approximation algorithms ensure a minimum quality of the obtained solutions by using mathematical properties, while the heuristic algorithms produce solutions with no theoretical quality.

Despite that fact, heuristic algorithms have been proven to be very effective for solving combinatorial optimization problems, requiring less computational resources for providing good quality solutions. In fact, in this thesis, the proposed heuristic algorithms are able to outperform the currently published exact methods when middle and large size instances are considered.

According to [12], heuristic and metaheuristic algorithms can be classified into single-solution based heuristics and population-based heuristics.

The single-solution based heuristics modify and optimize a single solution, while those of the second group maintain and improve multiple candidate solutions. Some examples of population-based heuristics include the well-known genetic and ant colony optimization algorithms.

In each chapter of this thesis different single-solution based heuristic algorithms are proposed for solving the respective problems. The proposed algorithms combine different local-search procedures, which can be summarized as follows:

- Simulated annealing (SA)
- Variable neighborhood search (VNS)
- Variable neighborhood descent (VND)
- Iterated local search (ILS)

The SA method, originally proposed in [13], consists of generating random moves for the current solution, which are accepted if the move improves the solution value, otherwise, the move is accepted/rejected according to a certain probability, which depends on a "temperature" parameter. In other words, at the beginning almost all the moves are accepted, while when the temperature decreases, only the good moves (or at least those that are not so bad) are accepted [14]. The stopping condition of SA is usually a final temperature parameter.

The VNS algorithm, introduced in [15], combines local search and shaking procedures (random moves selected from the currently explored neighborhood). The local search procedure is used for achieving local optima, while the shaking procedure is used to escape from the local optima [16]. The VNS algorithm can be summarized as follows: given a set of neighborhoods, and starting from the first neighborhood, *i*) apply the shaking procedure to the current solution (incumbent), *ii*) apply a local search procedure to the solution obtained after the shaking procedure, *iii*) if the

solution obtained after the local search procedure is better than the incumbent update it, and set the current neighborhood equal to the first one, otherwise, explore the next neighborhood. This procedure is repeated until the last neighborhood is explored [15].

The VND can be considered a particular case of the VNS. This approach explores a set of neighborhoods in descending order, under a deterministic criterion. Usually the neighborhoods are sorted according to their complexity; in this way the less complex neighborhoods receive a more intense exploration w.r.t. the more complex ones, avoiding extremely large computing times. The main principle of the VND (also for VNS) approach is that a local optimum for a certain neighborhood is not necessarily a local optimum for another neighborhood [17].

Despite it is not clear who introduced the ILS algorithm, there are several authors that have contributed to its development along the years; some relevant works can be found in [18, 19], and [20]. The ILS schemes combine a local search procedure with a perturbation procedure. The perturbation procedure is used for avoiding local optima, allowing the algorithm to explore new solution spaces. The stopping condition of ILS is usually a maximum number of iterations. One of the main advantages of ILS is its simplicity, furthermore, it leads to better solutions w.r.t. the execution of the corresponding local search procedure from scratch with several random trials. [21].

When uncertainty is come to be addressed, there are mainly two methodologies that can be used: *i*) robust optimization, when the probability distribution of the parameters under uncertainty is not known, and *ii*) stochastic optimization, when the probability distribution of the parameters under uncertainty is known.

A common methodology for approaching stochastic location routing problems is the two-stage optimization [22, 23]. This methodology considers first stage decisions which should be made before uncertainty is disclosed (e.g. the location decisions), and second stage decisions which must be taken after uncertainty is revealed (e.g. the routing decisions).

An important aspect to be considered in a stochastic programming methodology is the attitude of the decision maker to risk. The most common approach is to consider a risk-neutral decision maker, which leads to optimize the expected value of the second stage problem. Nevertheless, in problems like latency routing, which are suitable for modeling post-disaster operations and humanitarian logistics problems, a risk-averse decision maker makes more sense [24].

The risk measure adopted in Chapter 4 is the conditional value at risk (CVaR), which seeks for minimizing the expected value of the $1 - \alpha$ worst case scenarios, where α is a probability.

Capturing the stochasticity of a problem may depend of the number of scenarios that are consider. The complexity of a problem grows when the number of scenarios grows. For some cases it is necessary to consider parameters under uncertainty with continuous probability distributions, which in practice is equivalent to consider a stochastic programming problem with an infinite number of scenarios. This method cannot be used for problems which in their deterministic versions are already NP-hard. In order to tackle problems with continuous probability distributions the so-called sampling methods have been used to approximate the optimal solution [25]. These methods consist of solving samples with a reduced number of independently generated scenarios (Monte-Carlo simulation). This kind of methodologies has been applied to several combinatorial optimization problems, and when the technique used for solving each sample is a heuristic algorithm the method is called simheuristic [26].

Each chapter of this thesis is self-contained. Chapter 2 presents three metaheuristics combining SA and VND for solving the latency location routing problem (LLRP). An ILS scheme for solving the LLRP, the multi-depot cumulative capacitated vehicle routing problem (MDC-CVRP) and the multi-depot k-traveling repairman problem (MDk-TRP) is detailed in Chapter

3. Finally, Chapter 4 studies the latency location routing problem with stochastic travel times (LLRP-STT), presenting a VNS algorithm, a two-stage stochastic programming model, and a sampling method (including also a simheuristic algorithm).

Bibliography

[1] P. Toth and D. Vigo, Vehicle routing: problems, methods, and applications. SIAM, 2014.

[2] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez, and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," Computers & Industrial Engineering, vol. 79, pp. 115–129, 2015.

[3] C. Prodhon and C. Prins, "A survey of recent research on location-routing problems," European Journal of Operational Research, vol. 238, no. 1, pp. 1–17, 2014.

[4] M. Drexl and M. Schneider, "A survey of variants and extensions of the location-routing problem," European Journal of Operational Research, vol. 241, no. 2, pp. 283–308, 2015.

[5] S. T. W. Mara, R. Kuo, and A. M. S. Asih, "Location-routing problem: a classification of recent research," International Transactions in Operational Research, 2021.

[6] T. Vidal, G. Laporte, and P. Matl, "A concise guide to existing and emerging vehicle routing problem variants," European Journal of Operational Research, vol. 286, no. 2, pp. 401–416, 2020.

[7] M. Fischetti, G. Laporte, and S. Martello, "Delivery man problem and cumulative matroids," Operations Research, vol. 41, pp. 1055–1064, dec 1993.

[8] E. Lalla-Ruiz and S. Voß, "A POPMUSIC approach for the Multi-Depot Cumulative Capacitated Vehicle Routing Problem," Optimization Letters, vol. 14, no. 3, pp. 671–691, 2019.

[9] M. Moshref-Javadi and S. Lee, "The latency location-routing problem," European Journal of Operational Research, vol. 255, no. 2, pp. 604–619, 2016.

[10] J. F. Sze, S. Salhi, and N. Wassan, "The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search," Transportation Research Part B: Methodological, vol. 101, pp. 162–184, 2017.

[11] K. Corona-Gutiérrez, S. Nucamendi-Guillén, and E. Lalla-Ruiz, "Vehicle routing with cumulative objectives: A state of the art and analysis," Computers & Industrial Engineering, vol. 169, p. 108054, 2022.

[12] E.-G. Talbi, Metaheuristics: from design to implementation. John Wiley & Sons, 2009.

[13] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," science, vol. 220, no. 4598, pp. 671–680, 1983.

[14] D. Delahaye, S. Chaimatanan, and M. Mongeau, "Simulated annealing: From basics to applications," Handbook of metaheuristics, pp. 1–35, 2019.

[15] N. Mladenović and P. Hansen, "Variable neighborhood search," Computers & operations research, vol. 24, no. 11, pp. 1097–1100, 1997.

[16] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, Variable neighborhood search. Springer, 2019.

[17] A. Duarte, N. Mladenovic, J. Sánchez-Oro, and R. Todosijević, Variable Neighborhood Descent, pp. 341–367. Cham: Springer International Publishing, 2018.

[18] J. Baxter, "Local optima avoidance in depot location," Journal of the Operational Research Society, vol. 32, no. 9, pp. 815–819, 1981.

[19] D. S. Johnson, "Local optimization and the traveling salesman problem," in International colloquium on automata, languages, and programming, pp. 446–461, Springer, 1990.

[20] E. B. Baum, "Towards practical 'neural'computation for combinatorial optimization problems," in AIP Conference Proceedings, vol. 151, pp. 53–58, American Institute of Physics, 1986.

[21] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," Handbook of metaheuristics, pp. 129–168, 2019.

[22] M. Albareda-Sambola, E. Fernández, and G. Laporte, "Heuristic and lower bound for a stochastic location-routing problem," European Journal of Operational Research, vol. 179, no. 3, pp. 940–955, 2007.

[23] C. L. Quintero-Araujo, D. Guimarans, and A. A. Juan, "A simheuristic algorithm for the capacitated location routing problem with stochastic demands," Journal of Simulation, vol. 15, no. 3, pp. 217–234, 2021.

[24] N. Noyan, "Risk-averse two-stage stochastic programming with an application to disaster management," Computers & Operations Research, vol. 39, no. 3, pp. 541–559, 2012.

[25] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," SIAM Journal on optimization, vol. 12, no. 2, pp. 479–502, 2002.

[26] A. A. Juan, J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira, "A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems," Operations Research Perspectives, vol. 2, pp. 62–72, 2015.

2 Effective metaheuristics for the latency location routing problem

Abstract

The latency location routing problem (LLRP), a combination of the facility location problem and the cumulative capacitated vehicle routing problem, is a recently proposed variant of location routing problems. It corresponds to a customer-centric problem, in which the aim is to minimize the sum of the arrival times at the customers. this work proposes three novel metaheuristic algorithms to solve the LLRP. They use a simulated annealing (SA) framework, which after each temperature reduction is intensified through a variable neighborhood descent (VND) procedure. Each algorithm uses a different search strategy as intensification. Results on 76 benchmark instances indicate that the proposed metaheurstics outperform the state-of-the-art algorithms, finding new best solutions for all the large size instances (over 100 customers), or the currently known optimal ones for most of the small and medium size instances, in comparable computing times. Furthermore, in more than 80% of the instances the average value of the solutions found by the proposed algorithms is better than or equal to that of the current best known solution.

keywords: cumulative routing; LLRP; location routing; simulated annealing; variable neighborhood descent

2.1 Introduction

The location routing problem (LRP) corresponds to a problem in which two decisions must be taken simultaneously: the location of logistic facilities and the routing of vehicles departing from these facilities. Therefore, the LRP is a combination of two well know combinatorial optimization problems: the facility location problem (FLP) and the vehicle routing problem (VRP). Since both these problems are NP-hard, the LRP is NP-hard too. The LRP and its extensions have been largely studied in the last decades due to their important applications in transportation and logistics. The traditional approach has been to minimize the total cost, which includes the transportation cost, the fixed costs of set-up the open facilities, and the costs of the used vehicles. Nevertheless, LRP may not be appropriate to model customer-centric problems, in which customer satisfaction, defined as the requirement to attend the customer requests as soon as possible, is the key factor. As an answer to this issue [1] began studying the latency location routing problem (LLRP), which is a combination of the FLP and the cumulative capacitated vehicle routing problem CCVRP (see [2]). The LLRP seeks for minimizing the sum of the arrival times at the customers (latency). As other cumulative routing problems, research on LLRPs is mainly motivated by post-disaster planning operations and humanitarian logistics problems

This chapter is based on the paper: Osorio-Mora A, Rey C, Toth P, Vigo D. Effective metaheuristics for the latency location routing problem. International Transactions in Operational Research. 2023 Apr 11. doi: https://doi.org/10.1111/itor.13294

[2, 1]. Since the LLRPs are closely related to humanitarian logistic, a bad planning may lead to losses of human lives. As pointed out in [3], the LLRP has also important applications in commercial systems where perishable products should be delivered, and in city logistics problems, where deliveries to the customers are performed by means of shared intermediate facilities.

It has been empirically proved that the solutions found by algorithms designed for routing or location-routing problems minimizing the global travel time are not appropriate for latency problems [1, 4]. This is related to the nature of the two objective functions: while the global travel time of a route can be computed as the sum of the travel times of the edges composing it, the latency of a route corresponds to the sum of the travel times of the edges, with each travel time multiplied by the number of customers following the considered edge in the route. It is also to note that for the latency problems the last edge of each route is not involved in the computation of the corresponding term of the objective function. There is also evidence that the sequential approach, that is, solving first the location problem and then the routing problem, leads to sub-optimal solutions [5]. This consideration strongly supports the importance of solving efficiently LLRPs.

In this chapter three versions of a new metaheuristic algorithm called SA-VND are proposed to solve the LLRP. The algorithm is a combination of simulated annealing SA (see [6]) and variable neighborhood descent (VND) procedures [7]. The main difference between the three algorithms is the VND strategy used. The proposed approaches are capable to outperform, in comparable computing times, the state-of-the-art algorithms in terms of solution quality.

The paper is structured as follows. In Section 2.2 a formal description of the problem is provided, and the related literature is analyzed. Section 2.3 describes the proposed metaheuristics. In Section 2.4 the computational results and the comparison with the state-of-the-art exact and heuristic algorithms are presented and discussed. In addition, valid lower bounds are reported for the instances not solved to proved optimality by the previously proposed exact methods. Finally, in Section 2.5, the main conclusions are drawn and future directions are proposed.

2.2 Problem statement and literature review

The LLRP can be defined as follows. Let us consider a complete undirected graph $G = (V, E)$, where V corresponds to the set of nodes, and E is the set of edges. The set V is equal to $V' \cup D$, where V' represents the set of N_c customers, and D is the set of N_d homogeneous uncapacitated depots. Let also K be the set of N_v homogeneous vehicles, each with capacity Q. Each customer $i \in V'$ has a non-negative demand q_i. Each edge $(i, j) \in E$, with $i \neq j$, has an associated non-negative travel time c_{ij}, which satisfies the triangular inequality. The problem is to select at most p (where p is a given value) depots to open, from which the vehicles must perform their routes in order to minimize the sum of the arrival times at the customers. Each customer must be visited once, and each vehicle must start from an open depot. As previously mentioned, an important feature of the cumulative routing problems is that the edges traveled from the last customer of each route to a depot do not affect the objective function [2]. A mixed integer linear programming (MILP) model of the LLRP has been proposed in [1]. Although the MILP model considers a vehicle capacity potentially different for each vehicle, the algorithm proposed in this work and the instances studied were designed for homogeneous fleets of vehicles.

It is to note that, since the travel time matrix satisfies the triangular inequality and the edges connecting the last customer of each route to a depot do not affect the objective function, in an optimal solution of the LLRP $min\{N_v, N_c\}$ vehicles are used. On the other hand, in an optimal solution, the number of open depots can be smaller than p (for example, when the number of

depots "close" to the customers is smaller than p). However, since an open depot is not forced to be used (i.e., to have customers assigned to it), we can assume that the number of depots to be opened is exactly equal to p (although some of them could not be used).

In [1], due to the NP-hardness of the problem, the authors proposed two heuristic algorithms to solve efficiently the LLRP: a memetic algorithm (MA) and a recursive granular algorithm (RGA). According to their computational experiments, MA performs better than RGA for the instances analyzed. More recently, [3] proposed two MILP models, three enumerative algorithms and a GRASP-based iterated local search algorithm (GBILS) for the solution of the LLRP, and report computational experiments on a subset of the LLRP benchmark instances. They were able to provide the optimal solution for several instances with up to 50 customers using the five exact methods, while the metaheuristic algorithm GBILS was able to find globally better quality solutions than those obtained by the algorithms RGA and MA in small computing times. The computational results reported in [3] clearly show that exact methods are not able to find within reasonable computing times good quality solutions for the instances with more than 50 customers. As a consequence, it is necessary to design effective metaheuristic algorithms for tackling large LLRP instances.

In [8] the authors studied the green location routing problem (GLRP), addressing it as a cumulative location-routing problem. Despite the similarities in the names of the problems, the GLRP corresponds to the family of the cumulative vehicle routing problems (CuVRPs), in which the cumulative objective function is not the sum of the arrival times at the customers but the sum of the travel times of the edges traveled weighted by the load inside the vehicle. In the particular case of unitary loads both cumulative functions lead to the same value, nevertheless, the GLRP and the LLRP are not equivalent for several reasons. First, the aim of the GLRP is to minimize the total cost, which is the sum of the costs associated with the fuel consumption and the fixed costs for using the depots. The cumulative (CuVRP) idea is included in the fuel consumption expression but it is not the only component, since also the speed of the vehicle and the technical components (engine) are considered in the computation of this cost. In addition to the nature of the objective functions, the GLRP also considers time windows, which are not included in the LLRP. The differences between the CCVRPs and the CuVRPs have also been discussed recently in [9], where a review of the different cumulative routing problems is presented. It is to note that the LLRP is an extension of the CCVRP since, if $N_d = p = 1$, the problem reduces to a CCVRP.

For comprehensive surveys on the location routing problems the reader is referred to the following works: [10, 11, 12, 13, 14, 15, 16, 17], in chronological order.

2.3 The proposed algorithms

The proposed algorithms combine SA and VND techniques. The SA is a technique used to avoid local optima, in which random moves of a neighborhood are generated, and accepted with a certain probability, which decreases proportionally to a "temperature" parameter which is updated according to a "cooling" procedure. The acceptance of bad moves is used as diversification strategy, nevertheless, at low temperatures the algorithm intensifies the search by accepting almost only moves which improve the objective function value. The VND techniques are local search-based algorithms which use different neighborhoods of a certain solution. The basic principle of the VND algorithms is that a local optimum for a certain neighborhood is not necessarily a local optimum for other neighborhoods. VND applies a deterministic descent

search along each neighborhood until a local optimum w.r.t. all the neighborhoods is reached.

The proposed metaheuristics combine the properties of both algorithms, diversifying the search through the SA random exploration, and intensifying the search on potentially good solutions using a VND approach. Furthermore, the cooling procedure of the SA helps the algorithms to converge.

The proposed algorithms (whose pseudo-code is presented in Algorithm 1) start by setting the current solution s_c and the best feasible solution so far s_{bf} equal to the initial feasible solution s_0 (see Section 2.3.2). Then a simulated annealing framework is used. The current temperature *temp* is set equal to the initial temperature t_0, and the number of iterations without changes NC is set equal to 0. The procedure is applied until the minimum temperature t_f is reached ($temp \leq t_f$) or until the algorithm is unable to escape from a local optimum for a maximum number NC_{max} of temperature updates ($NC \geq NC_{max}$). For each temperature value, it_{SA} random moves are evaluated. The moves are chosen from neighborhoods that are classified in two groups: Group *i*) inter/intra route operators of *insertion*, *swap* and $2-opt$, and Group *ii*) *DepotOpenClose*, *RouteSwap*, and *RouteRelocation*. Neighborhoods in Group *i*) keep the opened depots, and the number of vehicles allocated to each depot unchanged, while neighborhoods in Group *ii*) can change these choices. The probability of selecting moves from Groups *i*) and *ii*) is different and depends on the parameter GP. Further details about the neighborhoods and the selection process are presented in Section 2.3.3. A new solution s_p is generated by applying one of the mentioned random moves to s_c, and is accepted as the new s_c under the classical SA conditions. The new solution s_p replaces the current solution s_c only if one of the two following conditions holds: *i*) if $\Delta f = f(s_c) - f(s_p) > 0$, where $f(s_c)$ and $f(s_p)$ are the objective function values of the current solution and of the current solution with the move applied, respectively, or *ii*) if $\Delta f \leq 0$ and $r < \exp^{(\Delta f/temp)}$ where r is a random number $\in [0, 1]$. The above rule ensures that the algorithms converge to a local optimum after a certain number of cooling steps. If $f(s_c) < f(s_{bf})$, and s_c is feasible (i.e., the result of the procedure $IsFeasible(s_c)$ is *true*), the current solution is saved as the best feasible solution so far. Notice that the algorithms allow infeasible solutions (violating the capacity constraints of the vehicles) in order to extend the search space. Infeasible solutions are penalized with a factor *pen* for each unit of load exceeding the vehicle capacity, see section 2.3.1 for details. If none of the it_{SA} moves generated were accepted, i.e. the current solution s_c remains the same as the solution s_{temp} at the beginning of the current temperature, the counter of non-changes NC is augmented by 1, otherwise it is set equal to 0. After the random phase a VND procedure ($VNDX(s_c)$) is applied to the current solution s_c. The type of VND procedure applied in this phase is the only difference between the three proposed algorithms; these procedures are presented in section 2.3.4. Then, the value of *temp* is reduced according to a cooling factor α. It is to note that the random moves applied before the VND procedure act as a perturbation step for escaping from local optima. Finally, when the stopping condition is reached, the Lin–Kernighan-Helsgaun heuristic (LKH-3) proposed in [18] is applied to each open depot, solving the corresponding CCVRP. The customers and vehicles assigned to each depot are obtained from the best feasible solution s_{bf}. If the solution s_{LKH} obtained by applying the LKH-3 heuristic is better than the current best feasible solution, s_{bf} is updated.

It is to note that the LKH-3 is the most recent update of the heuristic solver LKH, which was originally presented for solving the traveling salesman problem [19]. The LKH solvers are based on the Lin-Kernighan algorithm [20], which essentially consists of using λ-opt moves, calculating the value of λ in an iterative way. The last update of the solver (LKH-3) incorporates different optimization routines and features such as local search, special moves, perturbations, genetic

algorithm, and penalization functions in order to solve effectively a large number of routing problems (for further details see [18]).

Algorithm 1: Main Scheme

Input: t_0, t_f, GP, NC_{max}, it_{SA}, α, pen, s_0
Output: s_{bf} $(Final\ Solution)$

$temp = t_0$, $NC = 0$, $s_c = s_0$, $s_{bf} = s_0$
while $(temp > t_f\ and\ NC < NC_{max})$ **do**
  $it = 0$, $s_{temp} = s_c$
  while $(it < it_{SA})$ **do**
    $s_p = \text{RandomMove}(GP, s_c)$
    if $(AcceptanceCriteria(temp, s_p, s_c))$ **then**
      $s_c = s_p$
      if $(f(s_c) < f(s_{bf})\ and\ IsFeasible(s_c))$ **then**
        $s_{bf} = s_c$
      end
    end
    $it = it + 1$
  end
  if $(f(s_{temp}) = f(s_c))$ **then**
    $NC = NC + 1$
    $temp = \alpha * temp$
  else
    $NC = 0$
    $s_c = VNSX(s_c)$
    if $(f(s_c) < f(s_{bf})\ and\ IsFeasible(s_c))$ **then**
      $s_{bf} = s_c$
    end
    $temp = \alpha * temp$
  end
end
s_{LKH}= solution obtained by applying for each open depot the procedure LKH-3 to the set of customers and the set of vehicles assigned to the considered depot in the solution s_{bf}
if $(f(s_{LKH}) < f(s_{bf}))$ **then**
  $s_{bf} = s_{LKH}$
end
return: s_{bf}

2.3.1 Search space

In order to extend the search space and avoid local optima, all the parts of the algorithms accept infeasible solutions by applying a penalization term. Thus, the value of the objective function $f(s_c)$ of a solution s_c, feasible or not, is given by the following formula:

$$f(s_c) = \bar{f}(s_c) + pen\Delta Q \tag{2.1}$$

Where pen is a penalization coefficient calculated as a percentage of the value of the objective function corresponding to the initial solution, $\bar{f}(s_c)$ is the sum of the arrival times at the customers and ΔQ is the total amount of load violating the capacities of the vehicles. It is to note that for feasible solutions the second term of the formula is equal to zero. The best improvement strategy is used in the local search procedures, in which the penalized objective function is considered. In the same way, the random moves applied may also be infeasible and penalized.

2.3.2 Initial solution

In order to provide a good initial feasible solution s_0 in short computing time, a combination of the $k-means$ clustering algorithm [21] and the LKH-3 is used. Let us consider the instance with $N_c = 19$, $N_d = 5$, $p = 2$, and $N_v = 5$ presented in Figure 2.1a, where the green triangles are the customers and the red squares are the depots. The $k-means$ algorithm is used to define N_v clusters of customers and the corresponding centroids (Figure 2.1b: (the clusters are represented by big circles, and their centroids are represented by blue stars). Then, a scoring process is used to decide which depots are opened. The depots are chosen by considering their closeness to the centroids of the clusters. Each cluster is assigned to its closest depot, and each cluster assigned adds one point to the score of the depot (Figure 2.1c). Then, the p depots with the highest scores are opened (depots D2 and D3 in Figure 2.1d), and the clusters which were assigned to depots not selected, are allocated to the closest one among the open depots (Figure 2.1d). In case of depots with the same score, the selection is random. Let us consider OD as the set of the p opened depots. The next step is to apply the LKH-3 procedure, solving a CCVRP for each depot $j \in OD$, considering all the customers allocated to the depot j (Figure 2.1e). The number of vehicles assigned to each open depot j (NV_j), is calculated rounding up the ratio between the total demand of the customers allocated to j (dem_j), and the vehicle capacity: $NV_j = \lceil dem_j/Q \rceil$. In case $\sum_{j \in OD} NV_j \neq N_v$, a redistribution of the vehicles is carried out. If $\sum_{j \in OD} NV_j < N_v$, a new vehicle is assigned to the depot which has the smallest value of $(NV_jQ - dem_j)$, i.e. the depot which is using most intensively the capacity of its assigned vehicles. On the other hand, if $\sum_{j \in OD} NV_j > N_v$, a vehicle is removed from the depot which has the largest value of $(NV_jQ - dem_j)$, i.e., the depot with the largest unused vehicles capacity. The LKH-3 algorithm is applied considering its default parameter configuration, except for the number of runs which is set equal to 1. Since the obtained LLRP solution may be infeasible in terms of the vehicle capacity, a repair procedure is applied if one or more routes are infeasible. This procedure consists of applying a sequential inter-route local search procedure on the $insertion$, $swap$, and $2-opt$ neighborhoods (further details are given in section 2.3.3). The move which minimizes the sum of the penalizations associated with the infeasibilities, plus the global latency is applied (Figure 2.1f). Other approaches combining clustering and LKH procedures have been successfully used to solve the CLRP in [22]. Nevertheless, the proposed clustering and the location/allocation procedures are totally different from the one presented in this work. The mentioned work proposed an iterative procedure which splits a giant tour into clusters composed of consecutive customers taking into account the capacity of the vehicles, and

then the location/allocation is determined by solving a MILP model.

2.3.3 Local search

The whole framework of the algorithms uses 8 neighborhoods, and the random moves are selected from 6 of them. First, we describe the neighborhoods used in the random phase (Groups *i*) and *ii*)), and then the remaining two neighborhoods, which are used only in the VND procedure. Notice that the neighborhoods in Group *i*) are used also in the VND procedure. It is to note that, since the objective of the LLRP is to minimize the global latency (and not the global travel time) of the routes, each change of the current solution (concerning the positions of the customers within the routes, the assignment of a route to an open depot, the opening or closing of a depot) must be evaluated in an effective way in order to reduce the computing times of the proposed procedures.

The neighborhoods in Group *i*) can be applied in both the intra-route and the inter-route cases. For the inter-route case, they can be applied for routes starting from the same depot or from different depots. The neighborhoods are the following ones:

- *insertion*: This operator selects a customer i and a position j, and relocates customer i in position j.
- *swap*: This operator selects two customers i and j and swap their positions.
- $2-opt$: This operator deletes two edges, (i, j) and (k, l), and creates two new edges, (i, k) and (j, l). When the operator is applied to edges belonging to the same route, the edges (i, j) and (k, l) are deleted, then the edges (i, k) and (j, l) are created, and the path from k to j is reversed. The above is the classical $2-opt$ operator used in the traveling salesman problems. On the other hand, when the operator is related to two different routes, a crossing is applied: the heads of routes 1 and 2 (until nodes i and k, respectively) are merged with the tails of routes 2 and 1 (from customers l and j, respectively).

The mentioned neighborhoods have been largely used in the routing problem literature, and in the context of CCVRPs they were studied in the seminal work of [2], where the reader can find the way to compute them efficiently. The evaluation of a move in these neighborhoods can be performed in constant time by following the procedures proposed by [2].

The neighborhoods in Group *ii*) correspond to operators which change the depot/vehicle relationship. These operators are applied only in the random phase, and are described below:

- *DepotOpenClose*: This operator selects two depots: i (open) and j (closed). All the routes assigned to i are re-assigned to j. Since the current routes may not be good for the new depot, an intra-route local search ($IntraLS$) procedure is applied to each route by considering j as the starting depot. This procedure explores each *insertion*, *swap* and $2-opt$ neighborhood until no improvement is found, without cycles. The pseudo code of procedure $IntraLS$ is presented in Algorithm 2 (r_c and r_p represent, respectively, the input and output routes; $route(ng, r_p)$ denotes the route obtained by exploring the neighborhood ng starting from the route r_p).
- *RouteSwap*: This operator selects two routes r_1 and r_2 allocated to different open depots i and j, respectively. The relation route-depot is swapped, that is, r_1 will start from depot j and r_2 will start from depot i. The $IntraLS$ procedure is applied to the routes r_1 and r_2.

- *RouteRelocation*: This operator selects a route starting from a depot with more than one route assigned. Then, this route is reassigned to a different open depot, and the $IntraLS$ procedure is applied.

Algorithm 2: IntraLS procedure

Input: r_c, $NeighIntra = \{insertion, swap, 2-opt\}$
Output: r_p ($IntraLS$ *route*)

```
1  r_p = r_c
2  for (each ng ∈ NeighIntra) do
3  |   flag = true
4  |   while (flag = true) do
5  |   |   r'_p = route(ng, r_p)
6  |   |   if (f(r'_p) < f(r_p)) then
7  |   |   |   r_p = r'_p
8  |   |   |   flag = true
9  |   |   else
10 |   |   |   flag = false
11 |   |   end
12 |   end
13 end
```

return: r_p

Since the application of the $IntraLS$ procedure inside the neighborhoods of Group ii) implies a larger computational effort compared to the neighborhoods in Group i), a lower (or equal) probability of selecting moves in Group ii) is considered. This probability is defined by an integer parameter GP. The random move selection is detailed in Algorithm 3. Notice that if $GP = 1$ the random move has the same probability to be selected from all the six considered neighborhoods. Similar neighborhoods have been used in other location routing problems, as done in [23] for the CLRP. The main difference between the operators described in the mentioned paper and those presented here is the local search procedure applied at the end of the move. Furthermore, [23] include these operators into the *insertion* and *swap* neighborhoods, therefore they are not allowed to be treated as "special moves" (with different probabilities of being selected and optimized with a local search procedure).

Algorithm 3: Random move selection (RandomMove)

Input: GP, s_c
Output: s_p
$m = Rand[1, 3GP + 3]$
if $(m \leq GP)$ **then**
  $s_p = \text{RandomInsertion}(s_c)$
end
if $(GP < m \leq 2GP)$ **then**
  $s_p = \text{RandomSwap}(s_c)$
end
if $(2GP < m \leq 3GP)$ **then**
  $s_p = \text{RandomTwoOpt}(s_c)$
end
if $(m = 3GP + 1)$ **then**
  $(s_p) = \text{DepotOpenClose}(s_c)$
end
if $(m = 3GP + 2)$ **then**
  $(s_p) = \text{RouteSwap}(s_c)$
end
if $(m = 3GP + 3)$ **then**
  $(s_p) = \text{RouteRelocation}(s_c)$
end
return: s_p

It is to note that the presented simulated annealing scheme is also known in the literature as multi-neighborhood simulated annealing [24, 25].

Finally, the following two neighborhoods are considered only in the VND procedures. In case the operators are applied to different routes, these can start from the same depot or from different depots. These neighborhoods are also well-known in the literature related to the vehicle routing problems, and have been used successfully for the solution of the CCVRPs. Information about how to compute them in constant time can be found in [26]:

- $arc - swap$: Two pairs of consecutive customers (i, j) and (k, l) exchange their position. This operator can be applied both for the intra-route and the inter-route cases.
- $shift_{2-1}$: A pair of consecutive customers (i, j) assigned to route r_1, and a customer k assigned to a different route r_2 exchange their position.

2.3.4 Variable neighborhood descent strategies

Denote by $Neigh = \{insertion,\ swap,\ 2-opt,\ arc-swap,\ shift_{2-1}\}$ the set of the $N_{neigh} = 5$ previously described neighborhoods, and consider $ng \in Neigh$ as the ng^{th} neighborhood of the current solution s_c. The neighborhoods are explored according to the order in which they are listed in the set $Neigh$. The three search strategies VND0, VND1 and VND2 are described in the following:

- VND0: The exploration starts from the first neighborhood, which is explored until no improvement is found. Then the exploration moves to the next neighborhood, and the process is repeated until the last neighborhood does not improve the current solution s_c.

If some improvement was found in any of the neighborhoods, the search is restarted from the first neighborhood. If no improvement is found for all the neighborhoods, the VND0 procedure ends.

- VND1: The exploration starts from the first neighborhood. Each neighborhood is explored until no improvement is found, then the exploration moves to the next neighborhood: if there is an improvement, the search restarts from the first neighborhood. The procedure ends when no neighborhood improves the current solution s_c.

- VND2: This procedure is a combination of the VND0 and VND1 search strategies previously described. When an improvement is found at the ng^{th} neighborhood, the search remains at the current neighborhood until no improvement is found, then the search restarts from the first neighborhood. The procedure ends when no neighborhood improves the current solution s_c.

2.4 Computational results

The proposed metaheuristics were implemented in C++, and the experiments were carried out on an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz with 32 GB RAM, under Linux Ubuntu 18.04 operative system (single thread). The 76 instances belonging to the three available LLRP benchmark data sets were considered to compare the proposed algorithms with the state-of-the-art ones. The travel time matrix for all the instances was calculated with double precision. In order to have a fair comparison with the algorithms proposed in [1], thirty random seeds were created, and each instance was solved with each seed (i.e., 30 runs were executed for each instance).

The 76 instances used in [1] to test the algorithms MA and RGA correspond to the 36 instances of the Tuzun and Burke data set [27], the 30 instances of the Prodhon data set proposed by Prins et al. in [28], and 10 out of the 19 instances of the Barreto data set [29]. Only the results corresponding to the 30 instances of the Prodhon data set and to 8 out of the 10 instances of the Barreto data set are reported in [3].

2.4.1 Parameter tuning

Since the number of combinations of the parameters is too large to test all of them, the *iterated racing for automatic algorithm configuration* (IRACE) method (proposed in [30]) was used. This software applies an elitist procedure, which iteratively takes samples of parameter combinations according to a certain probability, selecting the best ones, and discarding those which lead to low quality results. At each iteration the samples are updated, and the parameter values with the best performance increase their probabilities of being selected.

The IRACE software was trained with a set of instances, which correspond to 1/3 of the instances of each data set, considering different geographical distribution types, sizes and other features. Globally, 26 out of the 76 instances were used: 12 from the Tuzun-Burke data set, 10 from the Prins et al. data set, and 4 from the Barreto data set. The objective of selecting a heterogeneous sample of the instances is to obtain a parameter configuration that fits well for different types of instances.

The output of the software IRACE is a set of parameter configurations, which correspond to the most promising ones according to the training phase. Then, after preliminary experiments, the best configurations were selected considering the obtained solution quality and the required

computing time. It is important to note that each algorithm was calibrated independently of the other algorithms. This implies that the algorithms SA-VND0, SA-VND1 and SA-VND2 have different parameter configurations. The values evaluated for each parameter were the following: t_0={500, 800, 1000, 1200, 1500, 2000}, t_f{1, 5, 10, 50, 100, 150}, α=[0.90 , 0.99], it_{SA}={30, 50, 100, 200, 300, 500}, NC_{max}={1, 2, 3, 4, 5, 6}, GP={1, 2, 3, 4, 5, 6}, and pen={1, 3, 7, 10, 15, 20}. The selected configurations are presented in Table 2.1. The calibration process required 67020.9 s, 118441.5 s, and 122871.7 s for the algorithms SA-VND0, SA-VND1, and SA-VND2, respectively.

Algorithm	t_0	t_f	α	it_{SA}	NC_{max}	GP	pen[1]
SA-VND0	500	1	0.9893	300	6	1	1
SA-VND1	1000	1	0.9888	300	5	2	1
SA-VND2	1500	1	0.9885	500	2	1	15

Table 2.1: Best configuration of parameters for each algorithm

2.4.2 Gobal results

In order to provide an understandable presentation, the results are divided into four subsections, one for each considered data set, and one for the overall data set. In the supplementary materials the reader can find the detailed results of each run.

Tables 2.2 to 2.4 show the results obtained by the algorithms RGA, MA, SA-VND0, SA-VND1 and SA-VND2 by executing 30 runs for each instance of the three data sets. Tables 2.3 and 2.4 also show the results obtained on the second and third data sets by the algorithm GBILS executed 5 times for each instance, and the best solution value found by the two MILP models (solved with the Gurobi 9.0.1 solver) and the three enumerative algorithms (implemented with the algebraic modeling language AIMMS) as reported in [3]. The experiments in [1] were performed on a 3.1 GHz computer with 4GB RAM. The above is the only information available about this computer. Since the differences between the used computers are not clear, the CPU times presented in the tables for the algorithms RGA and MA are those reported in [1]; nevertheless, by considering the ratio between the corresponding values of GHz, it is possible to estimate that our computer is about 1.2 times faster than that used in [1]. On the other hand, the computing times of the exact methods and of the algorithm GBILS are multiplied by a "scaling factor" equal to 0.61, since the computer used in this work is faster than that used in [3]. The value of the "scaling factor" was calculated as the ratio between the single thread scorings of the two computers, which can be obtained in https://www.cpubenchmark.net/.
For each instance, the following values are given;

- *Instance*: Name of the instance.
- N_c: Number of customers.
- N_d: Number of depots.
- LB: Lower bound. It corresponds to the largest value between LB1, LB2 (that are the lower bounds proposed in [1]), the optimal solution value of the linear relaxation found after implementing the first MILP model (Model 1) presented in [3] for the LLRP, and solving it using CPLEX 20.1, and the optimal solution value/best lower bound obtained

[1] As percentage of the value of the initial solution s_0

after implementing and solving (with a time limit of 216000 s) the first MILP model (Model 1) presented in [31] for the multi-depot k-traveling repairman problem (MDk-TRP). The optimal solution value of the MDk-TRP is indeed a valid lower bound for the LLRP, because it is a special case of LLRP in which the capacity constraints of the vehicles are not considered, and all the available depots can be opened. It is to note that new tighter lower bounds are reported in italic in the Tables 2.2 to 2.4. Furthermore, note that the value (331.9) of LB2 reported in [1] for the instance 20-5-1 of the Prodhon data set is larger than the proved optimal solution value 330.0 (see [3]). Hence, we implemented a correct procedure and computed again the value of LB2 for all the instances.

- BKS_0: Best known solution value considering the algorithms RGA, MA, the five exact methods and the algorithm GBILS (i.e., the best known solution value found by the current-state-of-the-art algorithms). Each BKS_0 value proved to be the optimal solution value by [3] is presented in boldface.

- BKS: Best known solution value considering all the algorithms.

- Gap_{LB}: Percentage gap between BKS and LB, computed as $Gap_{LB} = 100\frac{(BKS-LB)}{LB}$.

It is to note that for the Prodhon and Barreto data sets, the columns N_c and N_d are not reported since this information is present in the name of the instance: the first number corresponds to N_c and the second one to N_d.
In addition, for each algorithm and each instance the following values are reported:

- $Best$: Best solution value found.

- Gap_B: Percentage gap between $Best$ and BKS, computed as $Gap_B = 100\frac{(Best-BKS)}{BKS}$.

- Avg: Average solution value (computed over 30 runs) for the RGA, MA, SA-VND0, SA-VND1 and SA-VND2.

- $time$: Global computing time for finding the $Best$ value (expressed in seconds). For all the algorithms, with the exception of the exact methods, it corresponds to the average computing time spent for each run multiplied by the number of runs. The global computing time of the exact methods proposed in [3] is given by the sum of the "scaled" computing times required by the five exact methods for solving the considered instance. In particular, for each instance a time limit of 2 hours was imposed for the execution of each MILP model, while for the execution of each enumerative algorithm the time limit was set to 2000 seconds for the instances with $N_c \leq 50$ and to 2 hours for the remaining instances. This means that if, for an instance with $N_c > 50$, all the five exact methods reach the corresponding time limit, the value of $time$ is given by 0.61*(2*2000+3*7200)=21960.00 seconds.

The values of $Best$ equal to BKS are presented in boldface. The average values Avg better than or equal to BKS_0 are presented in italic. For each column, at the bottom of the corresponding table an average summary (Global avg) is presented, in which the best values are presented in boldface.

The Tuzun–Burke data set

The results for these instances are given in Table 2.2. No results for the exact algorithms and the algorithm GBILS are presented on this data set in [3]. As the table indicates, the proposed metaheuristics outperform the other two algorithms (RGA and MA) in terms of solution quality. Each of the proposed algorithms is able to improve the value BKS_0 for all the 36 instances of this data set. Algorithms SA-VND0, SA-VND1 and SA-VND2 provide new values of BKS for 14, 10 and 13 instances, respectively. The average gap between the best solution value found and the new value of BKS is 8.7% for RGA and 5.0% for MA. For these two algorithms, for some instances the gap reaches 10%. It is to note that for each of the algorithms SA-VND0, SA-VND1, and SA-VND2 the corresponding average values Avg are better than BKS_0 for 35 instances. In terms of computing time, both RGA and MA are faster than the proposed metaheuristics. The fastest one among the three proposed algorithms is SA-VND0. For all but 6 instances in this data set, the value of the solution obtained by solving the MDk-TRP is a lower bound tighter than that proposed in [1]. The new LB values presented for these instances correspond to the optimal solution values of the MDk-TRP, with exception of the instances 121112, 121122, 121222, and 123122 for which the best lower bound is presented. It is to note that these values are very close to the optimal solution ones (the average MILP optimality gap is equal to 0.008%). The average and maximum values of Gap_{LB} are equal to 15.1% and 32.4%, respectively.

The Prodhon data set

The results of this data set are presented in Table 2.3. Also for these instances, the proposed algorithms globally outperform the currently published algorithms in terms of solution quality. Over the 30 instances of this data set, each of the proposed algorithms is able to improve or find the value BKS_0 for 27 instances. Algorithms SA-VND0, SA-VND1 and SA-VND2 provide new values of BKS for 10, 5 and 5 instances, respectively, while for 9 instances the proposed algorithms find solution values equal to BKS_0. It is to note that for these 9 instances the solutions found by the exact methods were proved to be optimal by [3]. The best solution values found by the three proposed metaheuristics are equal to those found by the algorithms GBILS for 5 instances, while are better for 25 (SA-VND0), 24 (SA-VND1) and 24 (SA-VND2) instances. For the four instances with $N_c = 20$ the three proposed algorithms converge always to the optimal solution. Regarding the average values Avg, the proposed algorithms find better values than those found by RGA and MA for all the instances. In addition, for 20 (SA-VND0), 18 (SA-VND1) and 19 (SA-VND2) instances, the average value Avg obtained by the proposed metaheuristics is better than or equal to BKS_0. The overall results are very similar for the three new algorithms, with SA-VND0 showing slightly better results. Also for this data set, in terms of computing time, MA, RGA and GBILS are faster than the proposed metaheuristics, and the fastest one among the latter algorithms is SA-VND0. The three proposed algorithms are globally faster than the exact methods. For 18 (resp. 1) instances in this data set the value of the optimal solution of the MDk-TRP (resp. the linear relaxation of the LLRP) is the tightest lower bound, while for the remaining 11 instances LB1 is the largest lower bound value. The average and maximum values of Gap_{LB} are equal to 8.9% and 25.0%, respectively. However, it is to note that for the 11 instances of this data set whose optimal solution value is known, the average and maximum values of Gap_{LB} are equal to 5.4% and 11.4%, respectively, while the average and maximum values of Gap_B (i.e., of the percentage gap of the $Best$ value with respect to the corresponding optimal solution value) for the proposed metaheuristics are equal

to 0.05% and 0.5%, respectively, which means that the real optimality gaps for these instances are much smaller than the corresponding Gap_{LB} values.

The Barreto data set

For this data set, the results obtained by the proposed metaheuristics and by the state-of-the-art algorithms are given in Table 2.4. It is to point out that for the instance $Christ - 50 - 5$ the best solution values found by the proposed algorithms ($Best = 1661.6$), and also by the algorithm MA ($Best = 1690.8$), are smaller than the optimal solution value ($Best = 1719.9$) reported in [3]. After implementing and executing (using CPLEX 20.1) the first MILP model (Model 1) presented in [3] we proved that the solution found by the proposed metaheuristics is optimal (the details are given in the Appendix 2.5), hence its solution value is presented in boldface in column BKS. For the instance $Christ - 75 - 10$ we found a valid lower bound equal to 2260.1, which is larger than the best solution value ($Best = 2228.4$) reported in [3]. Because of the above, these two instances were not considered in the global average results of the exact methods and of the algorithm GBILS. For the instances Min-134-8 and Or-117-14 no results are reported in [3]. The last line of Table 2.4 gives the average values (Global avg_{NG}) computed by considering only the 6 instances whose values are correctly reported in [3]. The table shows that each of the proposed algorithms is able to improve the value BKS_0 for 5 out of the 10 instances (including the optimal solution found for the instance $Christ - 50 - 5$); for the remaining 5 instances, the three algorithms are able to find the optimal solution value. Algorithms SA-VND0, SA-VND1 and SA-VND2 provide new values of BKS for 4, 2 and 2 instances, respectively. Comparing the average values Avg, for all the instances the proposed algorithms find better results than those obtained by the RGA and MA algorithms. In addition, for all the instances but one, the average value Avg obtained by the proposed metaheuristics is better than or equal to BKS_0. Also, for 4 instances the average value Avg obtained by the three proposed algorithms is equal to the optimal solution value. In terms of computing times there are not big differences between the heuristic algorithms, with the exception of algorithm GBILS, which has much smaller computing times. On the other hand, the exact methods are clearly more time consuming than the heuristic ones. The global results show that the performances of the three proposed algorithms are similar for what concerns the solution quality, while algorithm SA-VND0 has the smallest computing times. For all the instances in this data set the values of the new proposed lower bounds are tighter than those proposed in [1]. For 9 instances the solution value found by solving the MDk-TRP is the best lower bound, and for one instance the value of the linear relaxation of the LLRP is the tightest one. All the 9 instances but the instance $Min - 134 - 8$ were solved to optimally (solving the corresponding MDk-TRP), and the MILP optimality gap obtained for this instance is equal to 0.014%. The average and maximum values of Gap_{LB} are equal to 6.2% and 12.9%, respectively, considering all the 10 instances, and to 5.4% and 12.6%, respectively, considering only the 6 instances correctly solved by [3]. However, it is to note that for the 6 instances of this data set whose optimal solution value is known, the average and maximum values of Gap_{LB} are equal to 5.1% and 12.6%, respectively, while the proposed metaheuristics solve all these instances to optimality.

Overall data set

By considering the three data sets, we can note that the average values of the percentage gaps between Avg and $Best$ computed for each instance and each of the algorithms RGA, MA, SA-VND0, SA-VND1 and SA-VND2 are, respectively, 10.1%, 11.3%, 1.5%, 1.6% and 1.4% for the

Tuzun-Burke data set, 6.5%, 6.6%, 0.9%, 0.9% and 0.9% for the Prodhon data set, and 10.9%, 6.4%, 0.7%, 0.8% and 0.7% for the Barreto data set. This proves that the proposed algorithms not only provide better solutions, but also are much more stable than the current state-of-the-art algorithms. This analysis cannot be performed for the algorithm GBILS, since the average values found by this algorithm are not reported in [3]. Moreover, it can be noted that the algorithm SA-VND0 is able to find or improve the best solution value found by GBILS for all the 36 instances analyzed, while the algorithms SA-VND1 and SA-VND2 do that for all the instances but one. Furthermore, in, respectively, 29, 28 and 29 instances the average value obtained by the algorithms SA-VND0, SA-VND1 and SA-VND2 is better than or equal to the best solution value found by GBILS. Similarly, the proposed algorithms are able to find or improve the best solution value found by the exact methods for all the instances but three. Furthermore, in, respectively, 28, 27 and 27 instances the average value obtained by the algorithms SA-VND0, SA-VND1 and SA-VND2 is better than or equal to the best solution value found by the exact methods. With respect to the lower bounds, by considering all the 76 instances of the three data sets, the value of the optimal solution/best lower bound obtained by solving the MDk-TRP improved the LB value corresponding to LB1 and LB2 for 57 instances, while the value of the linear relaxation of the LLRP did it for 2 instances. The average and maximum values of Gap_{LB} are equal to 11.5% and 32.4%, respectively. Despite these values are large, it is important to remark that the current lower bounds are not good approximations for the optimal solution values since, as previously mentioned, even for the 17 instances for which the optimal solution value is known, the value of Gap_{LB} is large. Indeed, if only the instances solved to optimally are considered, the average value of Gap_{LB} is equal to 5.3%, with a maximum value equal to 12.6%. For 15 out of these 17 instances the proposed metaheuristics are able to find the optimal solution value, and, for the remaining 2 instances, the maximum value of Gap_B for the proposed algorithms is equal to 0.5%. Thus, since the proposed metaheuristics find optimal or near optimal solutions for these 17 instances, it is possible to infer that the algorithms provide good quality solutions also for the remaining 59 LLRP instances.

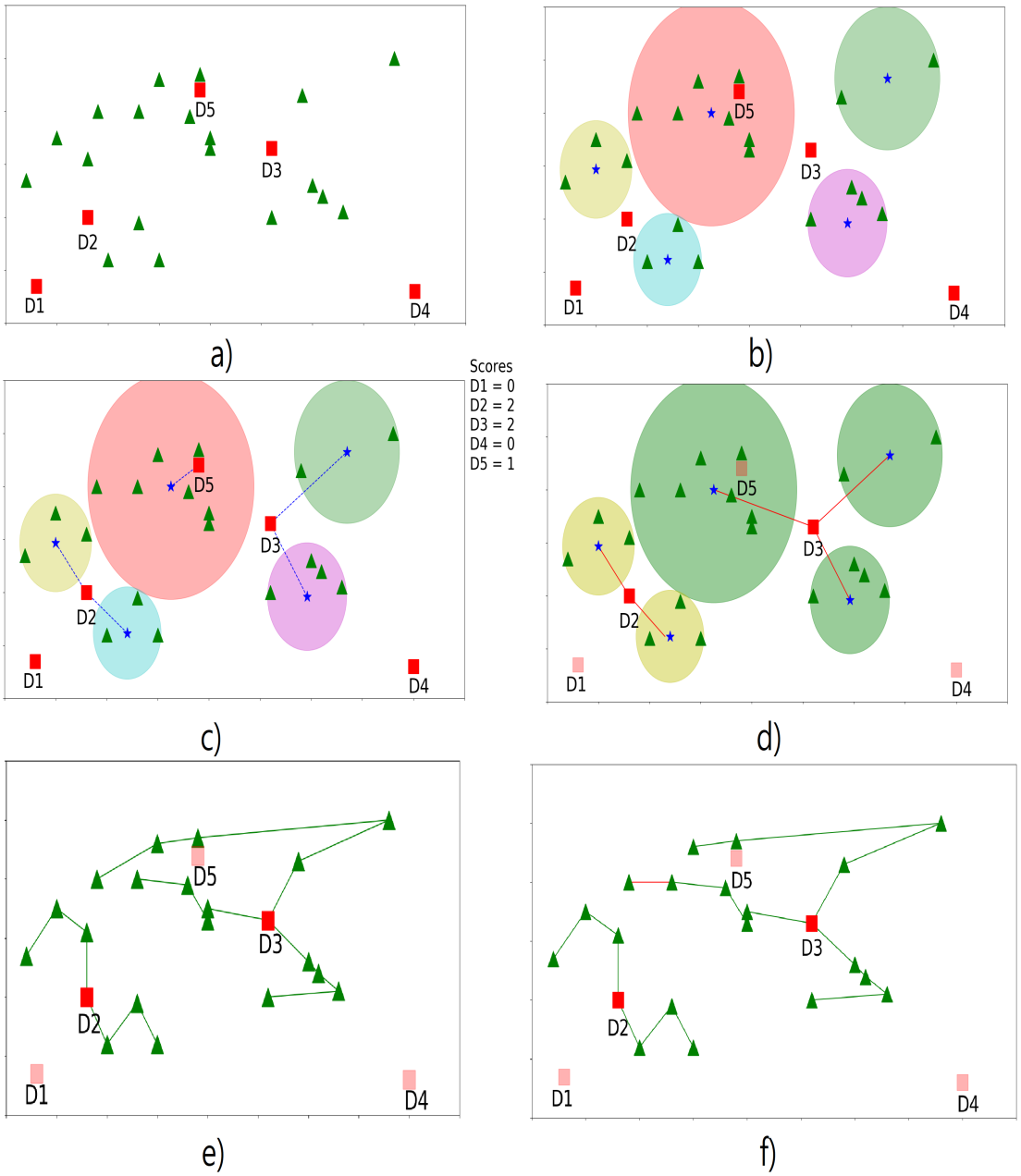


Figure 2.1: Initial Solution Procedure. (**a**) An LLRP instance. (**b**) $K-means$ procedure. (**c**) Scoring. (**d**) Allocation process. (**e**) LKH-3 procedure. (**f**) Repair procedure.

Table 2.2: Detailed results for the first data set (Tuzun-Burke Instances).

Instance	N_c	N_d	LB	BKS_0	BKS	Gap_{LB}	RGA				MA				SA-VND0				SA-VND1				SA-VND2			
							Best	Gap_B	Avg	time	Best	Gap_B	Avg	time	Best	Gap_B	Avg	time	Best	Gap_B	Avg	time	Best	Gap_B	Avg	time
111112	100	10	*3526.1*	4017.9	3862.9	9.6	4208.7	9.0	4544.2	630.0	4017.9	4.0	4270.1	1200.0	**3862.9**	0.0	*3971.5*	1912.8	3893.0	0.8	*3972.9*	2390.6	3882.8	0.5	*3964.1*	2801.2
111122	100	20	*3047.3*	3719.1	3612.4	18.5	3980.5	10.2	4348.2	651.0	3719.1	3.0	4087.5	1200.0	**3612.4**	0.0	*3694.7*	1934.6	3633.6	0.6	*3712.6*	2419.2	3623.7	0.3	*3687.8*	2860.6
111212	100	10	*3416.6*	4264.4	3938.9	15.3	4344.1	10.3	4769.2	870.0	4264.4	8.3	4563.5	1245.0	3960.2	0.5	*4038.2*	1805.4	3988.1	1.2	*4067.9*	2335.5	**3938.9**	0.0	*4066.1*	2692.0
111222	100	20	*3170.1*	4278.3	4068.3	28.3	4492.9	10.4	5025.4	642.0	4278.3	5.2	4557.3	1242.0	4086.7	0.5	*4140.3*	1909.6	4077.9	0.2	*4147.9*	2376.2	**4068.3**	0.0	*4138.8*	2785.2
112112	100	10	*2228.7*	2795.6	2739.2	22.9	2908.5	6.2	3557.3	825.0	2795.6	2.1	3049.7	1242.0	**2739.2**	0.0	*2755.5*	2281.4	2740.2	0.0	*2759.4*	2908.3	2741.8	0.1	*2756.1*	3271.1
112122	100	20	*1955.1*	2097.9	2057.5	5.2	2230.6	8.4	2524.7	792.0	2097.9	2.0	2702.6	1329.0	2060.3	0.1	*2072.6*	2211.6	**2057.5**	0.0	*2078.0*	2797.4	2060.8	0.2	*2071.9*	3205.3
112212	100	10	*1335.3*	1442.2	1397.4	4.7	1485.1	6.3	1615.9	846.0	1442.2	3.2	1604.9	1455.0	1403.0	0.4	*1416.2*	2349.9	1403.6	0.4	*1416.7*	3050.0	**1397.4**	0.0	*1415.0*	3335.3
112222	100	20	1346.6	1659.0	1621.4	20.4	1744.0	7.6	2470.4	822.0	1659.0	2.3	2084.1	1314.0	1623.7	0.1	*1633.0*	2522.2	**1621.4**	0.0	*1633.9*	3137.8	1626.9	0.3	*1633.9*	3562.2
113112	100	10	*2415.5*	2897.7	2835.8	17.4	3001.8	5.9	3271.5	792.0	2897.7	2.2	3162.6	1170.0	2837.5	0.1	*2852.6*	2043.2	**2835.8**	0.0	*2853.6*	2735.6	2839.5	0.1	*2857.5*	3014.0
113122	100	20	*2205.0*	2912.0	2774.4	25.8	3054.0	10.1	3322.8	810.0	2912.0	5.0	3330.0	1248.0	2776.4	0.1	*2782.5*	2063.4	**2774.4**	0.0	*2784.4*	2762.2	2776.4	0.1	*2782.1*	2998.4
113212	100	10	*1750.2*	1832.1	1815.6	3.7	1871.1	3.1	2095.2	654.0	1832.1	0.9	1901.0	1458.0	1817.0	0.1	*1823.2*	2326.4	**1815.6**	0.0	*1822.8*	2922.2	**1815.6**	0.0	*1822.8*	3326.7
113222	100	20	*1699.4*	2017.3	1876.1	10.4	2017.3	7.5	2239.8	654.0	2037.5	8.6	2360.6	1239.0	**1876.1**	0.0	*1888.5*	2117.2	1879.6	0.2	*1891.0*	2816.5	1876.9	0.0	*1888.9*	3063.5
131112	150	10	*4746.3*	5863.8	5448.9	14.8	6135.7	12.6	6528.5	1953.0	5863.8	7.6	6160.7	1797.0	5473.2	0.4	*5582.9*	4060.2	5464.2	0.3	*5570.1*	6108.7	**5448.9**	0.0	*5576.0*	6409.4
131122	150	20	*3982.4*	5310.3	4974.3	24.9	5591.9	12.4	6114.9	1977.0	5310.3	6.8	5649.3	1746.0	4993.4	0.4	*5142.1*	4462.0	5009.3	0.7	*5143.2*	6549.2	**4974.3**	0.0	*5105.7*	6876.6
131212	150	10	*4750.1*	5967.6	5606.3	18.0	6257.7	11.6	6589.8	1920.0	5967.6	6.4	6234.2	1746.0	5679.7	1.3	*5787.3*	4588.6	**5606.3**	0.0	*5785.2*	6765.0	5653.2	0.8	*5771.6*	7051.5
131222	150	20	*4146.1*	5261.0	5127.0	23.7	5634.9	9.9	6006.8	1950.0	5261.0	2.6	5610.4	1767.0	5141.9	0.3	5284.4	4580.2	**5127.0**	0.0	5277.6	6627.3	5134.4	0.1	5296.6	7093.9
132112	150	10	3545.5	4026.7	3868.9	9.1	4164.6	7.6	4342.8	1905.0	4026.7	4.1	4246.2	1746.0	**3868.9**	0.0	*3895.9*	5963.0	3883.4	0.4	*3899.4*	8542.7	3874.4	0.1	*3894.3*	8771.5
132122	150	20	3186.1	3874.0	3740.1	17.4	3978.6	6.4	4348.3	1908.0	3874.0	3.6	4185.5	1785.0	**3740.1**	0.0	*3795.9*	5155.0	3752.8	0.3	*3787.7*	7796.2	3755.5	0.4	*3797.3*	7768.9
132212	150	10	*2788.5*	2906.0	2837.8	1.8	2994.3	5.5	3140.4	1965.0	2906.0	2.4	3129.9	1779.0	2842.1	0.2	*2857.4*	5839.7	**2837.8**	0.0	*2860.7*	8792.6	2843.2	0.2	*2857.0*	8714.5
132222	150	20	*1612.1*	1784.8	1660.9	3.0	1862.0	12.1	1956.4	1992.0	1784.8	7.5	2152.3	1773.0	**1660.9**	0.0	*1692.0*	6340.4	1672.9	0.7	*1697.5*	9145.6	1678.0	1.0	*1695.1*	9288.8
133112	150	10	*4083.2*	4918.9	4588.4	12.4	4918.9	7.2	5579.3	2004.0	5034.0	9.7	5465.9	1737.0	**4588.4**	0.0	*4619.9*	4670.1	4598.2	0.2	*4630.7*	7313.4	4596.4	0.2	*4625.7*	7130.4
133122	150	20	2580.0	3474.0	3223.4	24.9	3552.7	10.2	3921.7	1974.0	3474.0	7.8	3849.1	1767.0	3223.4	0.0	*3259.5*	5317.0	3225.6	0.1	*3271.0*	8052.6	**3223.4**	0.0	*3248.4*	8012.0
133212	150	10	*2853.7*	3008.0	2907.0	1.9	3066.1	5.5	3199.3	1980.0	3008.0	3.5	3284.5	1770.0	2911.6	0.2	*2938.0*	5887.3	2911.4	0.2	*2938.0*	8572.0	**2907.0**	0.0	*2935.0*	8754.9
133222	150	20	2144.9	2617.4	2501.0	16.6	2732.7	9.3	3466.2	1926.0	2617.4	4.7	3016.9	1755.0	2503.0	0.1	*2550.0*	5692.6	2502.7	0.1	*2558.3*	8274.9	**2501.0**	0.0	*2531.6*	8664.8
121112	200	10	*5486.9*	7008.7	6608.5	20.4	7184.0	8.7	7753.5	2652.0	7008.7	6.1	7371.1	2502.0	**6608.5**	0.0	*6821.2*	8392.7	6621.6	0.2	*6881.4*	13919.9	6683.2	1.1	*6848.6*	14006.4
121122	200	20	*4526.2*	6039.6	5730.5	26.6	6181.5	7.9	6951.9	2685.0	6039.6	5.4	6501.6	2490.0	**5730.5**	0.0	*5954.2*	9991.8	5788.7	1.0	*5966.4*	15672.5	5784.7	0.9	*5952.6*	16313.2
121212	200	10	*5221.2*	6744.3	6429.6	23.1	7019.7	9.2	7515.9	2661.0	6744.3	4.9	7318.6	2559.0	6503.4	1.1	*6613.8*	8391.2	**6429.6**	0.0	*6608.9*	13778.5	6502.7	1.1	*6610.5*	13910.1
121222	200	20	*4947.6*	6828.5	6551.7	32.4	7330.3	11.9	7873.2	2694.0	6828.5	4.2	7567.2	2553.0	**6551.7**	0.0	*6759.2*	8391.4	6562.1	0.2	*6796.7*	14023.1	6648.2	1.5	*6776.4*	13851.2
122112	200	10	*5295.2*	6643.8	6154.6	16.2	6793.1	10.4	7106.1	2676.0	6643.8	7.9	7106.9	2571.0	**6154.6**	0.0	*6255.1*	9463.2	6184.7	0.5	*6280.3*	18340.7	6168.3	0.2	*6268.4*	14679.6
122122	200	20	3228.9	4012.9	3744.7	16.0	4160.9	11.1	4486.4	2721.0	4012.9	7.2	4915.7	2547.0	3757.4	0.3	*3782.5*	10555.0	3757.3	0.3	*3792.7*	17314.9	**3744.7**	0.0	*3785.5*	16701.0
122212	200	10	*3692.3*	4227.5	4043.5	9.5	4352.4	7.6	4449.8	2868.0	4227.5	4.5	4448.1	2535.0	4046.8	0.1	*4075.8*	9845.3	4046.4	0.1	*4078.6*	17116.3	**4043.5**	0.0	*4077.4*	15490.2
122222	200	20	*1947.7*	2127.9	2052.2	5.4	2251.7	9.7	2487.1	2739.0	2127.9	3.7	2345.5	2511.0	2054.3	0.1	*2083.6*	11266.8	2052.2	0.0	*2084.1*	18533.2	**2052.2**	0.0	*2079.7*	17652.2
123112	200	10	*4311.4*	5099.0	4917.0	14.0	5231.2	6.4	5537.9	2694.0	5099.0	3.7	5527.4	2559.0	**4917.0**	0.0	*5024.1*	10869.1	4967.1	1.0	*5047.9*	17275.1	4940.8	0.5	*5029.6*	17259.7
123122	200	20	*4019.4*	5046.4	4707.6	17.1	5046.4	7.2	5498.2	2628.0	5188.7	10.2	5862.9	2544.0	4725.9	0.4	*4771.9*	10590.0	**4707.6**	0.0	*4785.9*	16689.1	4719.9	0.3	*4777.1*	17021.6
123212	200	10	*4831.9*	5363.0	5170.8	7.0	5674.4	9.7	5865.3	2877.0	5363.0	3.7	5678.5	2544.0	**5170.8**	0.0	*5218.4*	10206.0	5178.0	0.1	*5225.0*	17737.1	5195.5	0.5	*5247.2*	16205.9
123222	200	20	*2433.8*	2657.5	2553.2	4.9	2726.8	6.8	3096.7	2643.0	2657.5	4.1	3917.6	2577.0	2567.2	0.5	*2629.5*	10676.9	2555.2	0.1	*2633.9*	18065.4	**2553.2**	0.0	*2606.5*	17096.2
Global avg.			3290.5	4020.7	3826.3	15.1	4171.7	8.7	4544.5	**1805.0**	4028.4	5.0	4422.8	1861.2	**3835.3**	**0.2**	**3901.8**	5740.9	3837.8	0.3	3909.5	8990.5	3841.0	0.3	3902.2	8934.4

Table 2.3: Detailed results for the second data set (Prodhon Instances).

Instance	LB	BKS_0	BKS	Gap_{LB}	RGA				MA				GBILS			Exact methods			SA-VND0				SA-VND1				SA-VND2			
					Best	Gap_B	Avg	time	Best	Gap_B	Avg	time	Best	Gap_B	time	Best	Gap_B	time	Best	Gap_B	Avg	time	Best	Gap_B	Avg	time	Best	Gap_B	Avg	time
20-5-1	*310.4*	**330.0**	330.0	6.3	331.9	0.6	366.7	261.0	337.3	2.2	378.0	387.0	**330.0**	0.0	1.0	**330.0**	0.0	36.9	**330.0**	0.0	*330.0*	125.2	**330.0**	0.0	*330.0*	117.8	**330.0**	0.0	*330.0*	166.4
20-5-1b	*573.7*	**608.1**	608.1	6.0	620.0	2.0	662.8	255.0	**608.1**	0.0	636.8	375.0	608.1	0.0	0.5	**608.1**	0.0	25.1	**608.1**	0.0	*608.1*	145.7	**608.1**	0.0	*608.1*	122.3	**608.1**	0.0	*608.1*	244.9
20-5-2	*279.5*	**302.0**	302.0	8.1	314.7	4.2	352.9	264.0	304.8	0.9	354.4	381.0	**302.0**	0.0	0.7	**302.0**	0.0	35.7	**302.0**	0.0	*302.0*	106.3	**302.0**	0.0	*302.0*	98.5	**302.0**	0.0	*302.0*	150.5
20-5-2b	*471.5*	**486.5**	486.5	3.2	**486.5**	0.0	540.1	246.0	**486.5**	0.0	511.2	381.0	**486.5**	0.0	0.8	**486.5**	0.0	15.1	**486.5**	0.0	*486.5*	158.7	**486.5**	0.0	*486.5*	132.8	**486.5**	0.0	*486.5*	266.6
50-5-1	*806.9*	**843.9**	843.9	4.6	876.6	3.9	938.4	732.0	859.9	1.9	917.3	546.0	846.9	0.3	153.5	**843.9**	0.0	2100.3	846.2	0.3	849.8	709.2	846.5	0.3	850.1	815.3	843.9	0.0	850.4	859.5
50-5-1b	*1161.4*	**1293.5**	1293.5	11.4	1343.6	3.9	1477.3	678.0	1330.2	2.8	1379.8	522.0	1293.9	0.0	65.5	**1293.5**	0.0	5857.0	**1293.5**	0.0	1293.7	619.7	**1293.5**	0.0	1293.5	602.2	**1293.5**	0.0	1293.6	933.8
50-5-2	*616.8*	**684.1**	684.1	10.9	752.5	10.0	801.0	720.0	723.4	5.7	786.2	552.0	691.7	1.1	117.4	**684.1**	0.0	7155.9	**684.1**	0.0	694.4	624.0	**684.1**	0.0	692.4	756.3	**684.1**	0.0	694.7	752.8
50-5-2b	*926.2*	**953.2**	953.2	2.9	990.8	3.9	1077.6	681.0	965.7	1.3	1009.4	573.0	954.9	0.2	68.4	**953.2**	0.0	2428.4	**953.2**	0.0	953.5	534.8	**953.2**	0.0	953.4	534.2	**953.2**	0.0	953.3	799.3
50-5-2BIS	*935.9*	**945.5**	945.5	1.0	957.3	1.3	974.2	711.0	955.2	1.0	981.5	537.0	952.6	0.8	120.7	**945.5**	0.0	5838.4	949.1	0.4	950.8	883.4	949.6	0.4	951.1	1081.1	950.1	0.5	950.9	1024.7
50-5-2bBIS	*801.2*	**803.9**	803.9	0.3	825.4	2.7	877.8	699.0	811.8	1.0	884.9	534.0	**803.9**	0.0	96.6	**803.9**	0.0	327.9	**803.9**	0.0	*803.9*	626.9	**803.9**	0.0	*803.9*	649.9	**803.9**	0.0	*803.9*	883.7
50-5-3	733.1	831.6	831.6	13.4	897.5	7.9	943.2	714.0	848.1	2.0	928.9	612.0	832.2	0.1	119.2	**831.6**	0.0	12444.0	832.0	0.0	835.1	712.3	833.0	0.2	834.9	810.3	833.6	0.2	835.2	863.8
50-5-3b	*1047.5*	**1101.6**	1101.6	5.2	1245.6	13.1	1331.5	681.0	1163.9	5.7	1198.8	531.0	1106.6	0.5	69.4	**1101.6**	0.0	6347.1	**1101.6**	0.0	1103.2	538.4	**1101.6**	0.0	1103.9	541.3	**1101.6**	0.0	1103.5	794.5
100-5-1	1896.3	2030.9	2004.3	5.7	2062.9	2.9	2096.5	1752.0	2030.9	1.3	2044.3	891.0	2035.6	1.6	64.8	2030.9	1.3	21960.0	**2004.3**	0.0	*2023.4*	3039.4	2010.5	0.3	*2023.8*	4791.3	2009.0	0.2	*2026.4*	3777.3
100-5-1b	*2210.8*	2357.9	2311.8	4.6	2437.9	5.5	2582.5	1602.0	2374.9	2.7	2507.8	744.0	2357.9	2.0	102.8	2374.9	2.7	21960.0	**2311.8**	0.0	*2336.6*	2221.6	2312.5	0.0	*2337.3*	2871.7	2313.7	0.1	*2342.8*	3126.5
100-5-2	1032.5	1144.7	1129.8	9.4	1206.5	6.8	1243.7	1725.0	1226.1	8.5	1500.9	852.0	1144.7	1.3	81.1	1206.5	6.8	21960.0	1132.4	0.2	*1136.0*	2760.8	**1129.8**	0.0	*1135.5*	3976.1	1131.3	0.1	*1135.7*	3582.7
100-5-2b	*1371.8*	1567.4	1507.9	9.9	1706.6	13.2	1865.2	1563.0	1622.9	7.6	1701.0	855.0	1567.4	3.9	96.8	1622.9	7.6	21960.0	**1507.9**	0.0	*1517.1*	2330.4	1510.6	0.2	*1519.0*	3141.3	1510.6	0.2	*1519.6*	3448.8
100-5-3	1465.4	1596.8	1581.9	8.0	1641.2	3.7	1654.1	1890.0	1710.4	8.1	1726.2	903.0	1596.8	0.9	49.2	1641.2	3.7	21960.0	**1581.9**	0.0	*1587.2*	2784.3	**1581.9**	0.0	*1586.5*	4072.3	**1581.9**	0.0	*1587.2*	3595.7
100-5-3b	*1547.0*	2032.1	1933.0	25.0	2087.1	8.0	2201.5	1569.0	2054.8	6.3	2190.5	870.0	2032.1	5.1	114.9	2054.8	6.3	21960.0	1933.7	0.0	*1950.9*	2315.8	1935.7	0.1	*1953.8*	2932.6	**1933.0**	0.0	*1955.0*	3247.9
100-10-1	1358	1481.6	1470.7	8.3	1573.0	7.0	1676.0	2124.0	1524.1	3.6	1589.9	1215.0	1481.6	0.7	80.3	1524.1	3.6	21960.0	1472.9	0.1	1511.0	2994.7	**1470.7**	0.0	1503.9	4143.4	1478.0	0.5	1511.0	3908.8
100-10-1b	*1645.3*	1960.7	1901.3	15.6	2152.0	13.2	2285.4	1935.0	1960.7	3.1	2138.6	1185.0	1984.9	4.4	100.8	1960.7	3.1	21960.0	**1901.3**	0.0	*1954.0*	2120.3	1915.8	0.8	1972.3	2723.1	1908.6	0.4	1963.5	2999.3
100-10-2	1011.5	1175.0	1142.3	12.9	1211.9	6.1	1326.1	2214.0	1175.0	2.9	1236.3	1326.0	1287.5	12.7	75.8	1175.0	2.9	21960.0	1143.3	0.1	*1152.8*	2899.7	**1142.3**	0.0	*1155.5*	4132.2	1145.9	0.3	*1155.3*	3739.8
100-10-2b	*1457.9*	1625.8	1564.0	7.3	1675.1	7.1	1857.0	2043.0	1625.8	4.0	1724.2	1230.0	1645.1	5.2	139.1	1625.8	4.0	21960.0	1566.5	0.2	*1585.7*	2208.4	1566.5	0.2	*1588.2*	2889.5	**1564.0**	0.0	*1578.9*	3115.3
100-10-3	1107.2	1216.2	1209.2	9.2	1280.6	5.9	1308.4	2214.0	1246.8	3.1	1288.3	1332.0	1216.2	0.6	57.2	1246.8	3.1	21960.0	**1209.2**	0.0	1221.5	3145.2	1209.9	0.1	1226.0	4290.2	1211.5	0.2	1224.7	4060.6
100-10-3b	*1437.6*	1745.1	1662.4	15.6	1860.8	11.9	2031.9	1848.0	1799.0	8.2	1890.2	1287.0	1745.1	5.0	70.1	1799.0	8.2	21960.0	**1662.4**	0.0	*1705.6*	2146.0	1665.7	0.2	*1706.7*	2831.1	1670.2	0.5	*1707.1*	3058.5
200-10-1	2688.4	2861.9	2797.9	4.1	2953.2	5.6	3125.1	5757.0	2920.7	4.4	3092.4	3414.0	2861.9	2.3	278.2	2920.7	4.4	21960.0	2798.6	0.0	*2854.1*	16846.7	**2797.9**	0.0	2863.3	26673.1	2803.6	0.2	*2860.6*	23819.4
200-10-1b	2688.4	3532.2	3327.1	23.8	3626.0	9.0	3992.6	4896.0	3532.2	6.2	3809.3	3240.0	3558.0	6.9	304.1	3532.2	6.2	21960.0	3368.7	1.3	*3477.1*	11341.6	3355.7	0.9	*3478.9*	17983.6	**3327.1**	0.0	*3452.8*	17334.9
200-10-2	1894.1	1997.0	1985.0	4.8	2049.2	3.2	2095.1	5703.0	2064.2	4.0	2153.3	3411.0	1997.0	0.6	342.5	2049.2	3.2	21960.0	**1985.0**	0.0	2002.0	17482.5	1986.6	0.1	2004.5	29590.3	1988.3	0.2	2002.5	24959.6
200-10-2b	*2036.9*	2473.2	2336.1	14.7	2598.9	11.2	2729.9	5382.0	2516.4	7.7	2684.5	3141.0	2473.2	5.9	273.7	2516.4	7.7	21960.0	**2336.1**	0.0	*2379.0*	12347.0	2355.2	0.8	*2378.2*	19720.6	2368.9	1.4	*2380.1*	18699.6
200-10-3	2620.2	2783.2	2741.2	4.6	2820.1	2.9	2885.4	5334.0	2805.9	2.4	3066.1	3084.0	2783.2	1.5	323.8	2805.9	2.4	21960.0	**2741.2**	0.0	*2758.1*	14786.4	2744.7	0.1	*2757.4*	27900.5	2751.2	0.4	*2763.8*	21623.8
200-10-3b	*2891.6*	3347.0	3225.8	11.6	3474.5	7.7	3651.9	4437.0	3347.0	3.8	3454.6	3267.0	3413.3	5.8	281.4	3347.0	3.8	21960.0	3242.2	0.5	*3274.6*	9511.1	3233.9	0.3	*3267.8*	16534.2	**3225.8**	0.0	*3273.7*	14933.5
Global avg.	1367.5	1537.1	1500.5	8.9	1602.0	6.1	1684.7	2021.0	1564.4	3.7	1610.3	1272.6	1546.3	2.3	**121.7**	1553.9	2.7	14596.4	**1503.0**	**0.1**	**1521.2**	3975.5	1503.9	0.2	1522.3	6248.6	1503.8	0.2	1521.8	5692.4

2.4.3 A comparison with the currently published heuristic algorithms

In order to compare the efficiency of the proposed metaheuristics with respect to that of the published ones, experiments to determine the computing time required to reach target values were carried out. Let us consider for each instance a target value given by the minimum value between the solution values provided by the algorithms RGA, MA and GBILS. These experiments considered a time-limit for each instance equal to the minimum between 3000 s and the global scaled computing time required by the corresponding best algorithm for executing its required number of runs. The SA-VND0 algorithm is able to find or improve the target value for all the instances, while both the SA-VND1 and SA-VND2 algorithms are not able to reach the target value for the instance 50-5-3, finding solution values with gaps equal to 0.1% and 0.17%, respectively. Furthermore, the SA-VND1 algorithm is not able to reach the target value within the time limit for the instance 200-10-2, obtaining a gap equal to 0.39%. Those are the only cases for which the proposed algorithms can not reach the target value within the time limit. The computing times for each instance and each algorithm are drawn in Figure 2.2, where Best heuristic corresponds to the best algorithm among RGA, MA and GBILS for the considered instance. For space reasons the horizontal axis presents only the associated data set and not the name of all the instances. The data sets are sorted from the less complex to the most complex (in terms of size) i.e. Barreto, Prodhon, and Tuzun-Burke. The summary of the average results regarding the computing times, and the number of instances for which each of the proposed algorithms is faster than the Best heuristic is presented in Table 2.6. According to the results, the global average computing time required by the Best heuristic to reach the target value is larger than that required by each of the three proposed algorithms (considering all the instances). Furthermore, the computing time required by the algorithms SA-VND0, SA-VND1, and SA-VND2 to reach the target value is smaller than the computing time required by the Best heuristic in 66, 61, and 59 instances, respectively (columns # faster than BH in the Table). Analyzing separately each data set, it is possible to note that the three proposed metaheuristics require considerably less computing time than the Best heuristic for finding the target value in the Tuzun-Burke and the Barreto data sets, while for the Prodhon data set the algorithm SA-VND0 is slightly faster than the Best heuristic, while the algorithms SA-VND1 and SA-VND2 are slightly slower than it. As we already proved in the previous sections, running the proposed metaheuristics for a longer time leads to much better solutions compared to those obtained by the state-of-the-art heuristic algorithms. Nevertheless, we also proved that, in general, the proposed metaheuristics are able to find similar quality solutions in shorter computing times compared to those of the published heuristic methods.

It is to note that, in order to reduce the global computing times of the proposed metaheuristics it is also possible to reduce the number of runs executed for each instance. The average results obtained when the number of runs is reduced to 10 and 5 runs are analyzed in Appendix 2.5.

Table 2.4: Detailed results for the third data set (Barreto Instances).

Instance	LB	BKS_0	BKS	Gap_{LB}	RGA				MA				GBILS			Exact methods			SA-VND0				SA-VND1				SA-VND2			
					Best	Gap_B	Avg	time	Best	Gap_B	Avg	time	Best	Gap_B	time	Best	Gap_B	time	Best	Gap_B	Avg	time	Best	Gap_B	Avg	time	Best	Gap_B	Avg	time
Christ-50-5	*1523.5*	1690.8	**1661.6**	9.1	1841.7	10.8	1972.3	738.0	1690.8	1.8	1782.4	591.0	-	-	-	-	-	-	**1661.6**	0.0	*1662.1*	541.5	**1661.6**	0.0	*1662.1*	528.7	**1661.6**	0.0	*1662.3*	813.7
Christ-75-10	*2260.1*	2590.3	2383.0	5.4	2703.4	13.4	2893.9	1023.0	2590.3	8.7	2689.8	873.0	-	-	-	-	-	-	2403.8	0.9	2459.0	1159.1	**2383.0**	0.0	2457.4	1384.4	2408.9	1.1	2454.9	1588.4
Christ-100-10	*3551.8*	3984.1	3792.0	6.8	4116.9	8.6	4450.8	1482.0	4058.2	7.0	4194.9	1023.0	3984.1	5.1	451.9	4043.1	6.6	21960.0	**3792.0**	0.0	*3831.2*	1958.7	3806.4	0.4	*3838.9*	2293.6	3795.2	0.1	*3825.4*	2876.3
Gaskell-21-5	*622.7*	653.5	653.5	4.9	699.8	7.1	810.2	324.0	658.4	0.8	741.1	441.0	**653.5**	0.0	0.9	**653.5**	0.0	14.9	**653.5**	0.0	*653.5*	116.4	**653.5**	0.0	*653.5*	102.2	**653.5**	0.0	*653.5*	169.5
Gaskell-29-5	*1064.9*	1199.3	1199.3	12.6	1238.4	3.3	1366.1	366.0	1224.5	2.1	1296.3	468.0	**1199.3**	0.0	5.2	**1199.3**	0.0	1618.1	**1199.3**	0.0	*1199.3*	311.0	**1199.3**	0.0	*1199.3*	255.4	**1199.3**	0.0	*1199.3*	485.9
Gaskell-32-5b	*1522.8*	1552.8	1552.8	2.0	1622.3	4.5	1786.2	372.0	1571.0	1.2	1668.4	483.0	**1552.8**	0.0	4.9	**1552.8**	0.0	612.9	**1552.8**	0.0	1553.3	417.8	**1552.8**	0.0	1553.3	337.6	**1552.8**	0.0	1553.3	650.7
Gaskell-36-5	*1596.3*	1627.2	1627.2	1.9	1646.1	1.2	1694.2	408.0	1642.4	0.9	1647.0	522.0	**1627.2**	0.0	3.2	**1627.2**	0.0	49.4	**1627.2**	0.0	*1627.2*	308.3	**1627.2**	0.0	*1627.2*	274.0	**1627.2**	0.0	*1627.2*	465.8
Min-27-5	*5382.0*	5387.6	5387.6	0.1	**5387.6**	0.0	6343.8	300.0	**5387.6**	0.0	5697.0	834.0	**5387.6**	0.0	84.0	**5387.6**	0.0	28.9	**5387.6**	0.0	*5387.6*	176.3	**5387.6**	0.0	*5387.6*	147.4	**5387.6**	0.0	*5387.6*	267.0
Min-134-8	*19355.7*	23387.0	21852.4	12.9	25496.0	16.7	28872.8	2406.0	23387.0	7.0	26012.5	2220.0	-	-	-	-	-	-	**21852.4**	0.0	*22307.3*	2225.7	21910.5	0.3	*22309.2*	2624.0	21881.8	0.1	*22278.1*	3250.7
Or-117-14	*50526.1*	56209.0	53798.5	6.5	60580.0	12.6	70626.3	1374.0	56209.0	4.5	61396.2	1545.0	-	-	-	-	-	-	**53798.5**	0.0	*54866.7*	1263.1	53859.1	0.1	*54805.9*	1265.5	**53798.5**	0.0	*54905.8*	1958.4
Global avg.	8740.6	9828.2	9390.8	6.2	10533.2	7.8	12081.7	879.3	9841.9	3.4	10712.6	900.0	-	-	-	-	-	-	**9392.9**	**0.1**	9554.7	**847.8**	9404.1	**0.1**	**9549.4**	921.3	9396.6	**0.1**	9554.7	1252.6
Global avg NG	2190.5	2335.7	2282.1	5.4	2451.8	4.1	2741.9	542.0	2423.7	2.0	2540.8	628.5	2400.7	0.8	**91.7**	2410.6	1.1	4047.4	**2368.7**	**0.0**	2375.3	548.1	2371.1	0.1	2376.6	568.4	2369.3	**0.0**	**2374.4**	819.2

Table 2.5: Average solution values and gaps provided by each part of the algorithms.

Data set	Initial		SA-VND0				SA-VND1				SA-VND2			
	Avg	gap	Avg SA	gap SA	Avg LKH	gap LKH	Avg SA	gap SA	Avg LKH	gap LKH	Avg SA	gap SA	Avg LKH	gap LKH
Tuzun-Burke	4513.2	20.4	**3931.5**	**2.4**	**3910.6**	**2.0**	3942.5	2.6	3916.9	2.2	3933.3	**2.4**	**3910.6**	**2.0**
Prodhon	1641.7	9.2	**1527.1**	**1.3**	**1523.6**	**1.2**	1528.2	1.4	1524.6	**1.2**	1527.5	1.4	1523.9	**1.2**
Barreto	12133.5	11.7	**9574.1**	**1.0**	9556.5	0.9	9575.2	**1.0**	**9551.1**	0.9	9577.6	**1.0**	9556.4	**0.8**
Global avg	4382.4	14.8	**3724.8**	**1.8**	**3711.3**	**1.5**	3730.6	1.9	3713.9	1.6	3726.3	**1.8**	**3711.3**	**1.5**

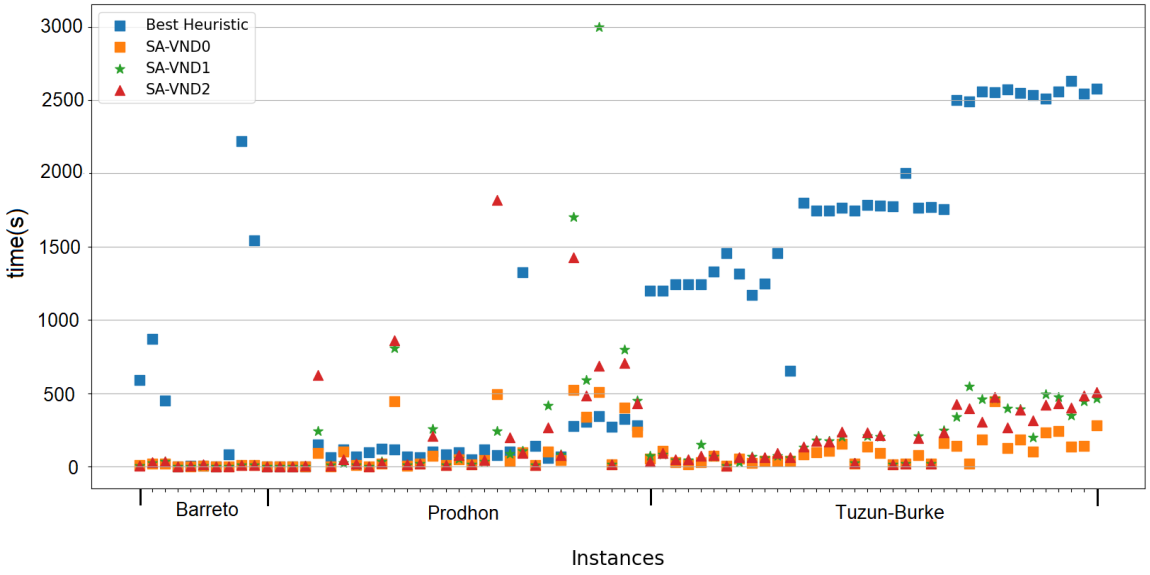


Figure 2.2: Computing times (s) required by each metaheuristic to reach the target values for the considered instances.

2.4.4 A detailed analysis of the components of the metaheuristics

Consider s_0 as the initial solution, s_{SA} as the best solution found by the proposed algorithm after finishing the simulated annealing frame, and s_{LKH} as the solution found by applying the LKH-3 procedure to s_{SA}. Table 2.5 presents, for each data set, the average values of the initial solution s_0 and of the best solutions s_{SA} and s_{LKH} found for each instance by executing 30 runs, and the averages of the corresponding gaps computed with respect to the best known solution value for each considered instance. It is to note that the initial solution is shared by the three proposed metaheuristics. The largest improvement is achieved by the simulated annealing frame, while the LKH-3 procedure further improves the s_{SA} solution value. This behavior is similar for the three metaheuristics. The largest improvements produced by the LKH-3 procedure (on average 0.4%) are found in the first data set, while for the second and the third data sets the average improvements are around 0.2% and 0.1%, respectively. Although the LKH-3 procedure does not improve substantially the solution values, it has an important effect on the stability of the algorithm; without including the LKH-3 procedure, the SA-VND0 and SA-VND2 algorithms are able to improve or find the value BKS_0 for, respectively, 70 and 72 instances, instead of the 73 instances they are able to find by including the LKH-3 procedure. On the other hand the algorithm SA-VND1 is able to improve or find the value BKS_0 for 73 instances with or without including the LKH-3 procedure. Furthermore, if the LKH-3 procedure is not executed, the number of instances for which the average values Avg obtained by the metaheuristics are better than or equal to BKS_0 is reduced by 4, 4 and 3, for the algorithms SA-VND0, SA-VND1, and SA-VND2, respectively.

In Figure 2.3 it is possible to analyze the percentage of the global computing time required by each stage of the algorithms. The simulated annealing frame (SA in the figure) is the one which requires the largest computing time. The computing time required by the LKH-3 procedure is close to 6% of the total computing time for the algorithms SA-VND1 and SA-VND2, and around 9% for the algorithm SA-VND0.

Table 2.6: Average computing times required by each metaheuristic to reach the target values for each data set.

data set	Best heuristic (BH)	SA-VND0		SA-VND1		SA-VND2	
	time	time	# faster than BH	time	# faster than BH	time	# faster than BH
Tuzun-Burke	1854.7	103.9	36	206.5	36	201.0	36
Prodhon	163.3	124.1	21	302.0	16	274.1	15
Barreto	577.9	7.9	9	9.2	9	11.5	8
Total	1019.0	99.3	66	218.3	61	204.9	59

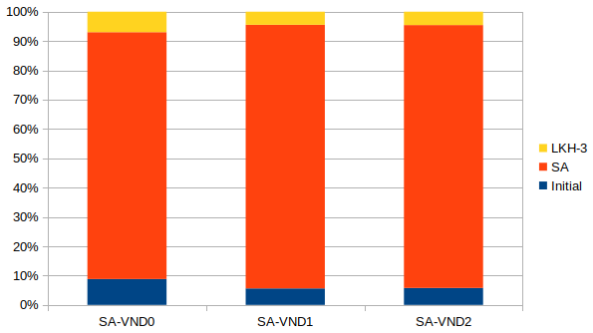


Figure 2.3: Relative computing time required by each part of the proposed metaheuristics.

Table 2.7: Average results obtained by each part of the combined phase (SA-VND).

Criteria	SA-VND0	pure SA0	pure VND0	SA-VND1	pure SA1	pure VND1	SA-VND2	pure SA2	pure VND2
Gap_B	0.1	9.6	4.9	0.2	10.6	4.7	0.2	9.0	6.1
Gap_{A-B}	1.3	11.8	10.4	1.4	11.9	10.2	1.3	11.7	11.6
Avg. time	146.7	38.0	25.1	228.2	34.7	25.6	221.5	47.8	25.2

It is possible to conclude that the proposed metaheuristics produce high quality results even without the LKH-3 procedure; nevertheless, by paying a relative low cost in terms of computing time, the stability of the algorithms improves.

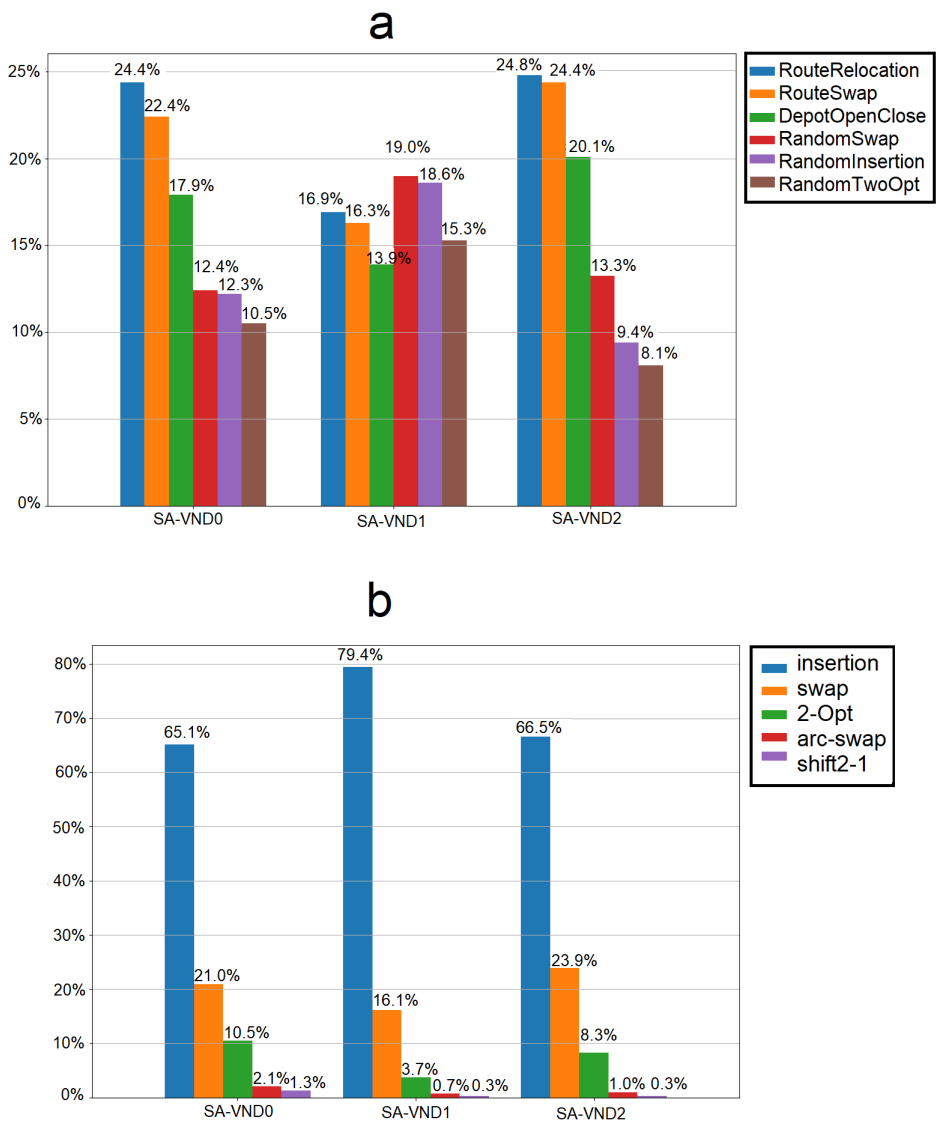


Figure 2.4: Percentage average contribution of the neighborhoods. (**a**) SA. (**b**) VND.

As it was previously shown, the combined part of the algorithm (SA-VND) is the one which impacts the most on the final solution. In order to understand the importance and the interaction for each component of the combined phase, that is, the SA and the VND procedures, experiments considering independently each procedure were carried out. Table 2.7 presents the average results, regarding the gap between the best solution value found and the updated BKS (Gap_B), the gap between the average solution value and the updated BKS (Gap_{A-B}), and the average computing time in seconds (Avg. time). The columns pure SA0, pure SA1, and pure SA2 correspond to the results obtained by deleting the VND0, VND1, and VND2 procedures,

respectively. It is to note that for the cases of pure VND0, pure VND1, and pure VND2, the parameter it_{SA} was set equal to 1 in order to re-start the VND procedures by evaluating/applying a single random move. According to the results, the computing times can be significantly reduced by applying each heuristic by itself. Nevertheless, the quality of the best and average solutions found is considerably affected. The values of Gap_B are close to 10% (5%) when only the SA (VND) is applied, while the original metaheuristics present Gap_B values close to 0.0%. Furthermore, the Gap_{A-B} values are always above 10% when each heuristic is independently applied, which means that it is necessary to run several times the algorithms in order to obtain good quality solutions. This behavior is totally the opposite to that observed in the original metaheuristics, which are very stable (as we proved in the previous experiments). These results highlight the importance of combining both heuristic approaches.
Another interesting analysis regards the importance of each neighborhood both in the SA and the VND phases. Figure 2.4 presents the percentage average contribution of each neighborhood in terms of the number of times it is applied over the total number of moves applied, for each of the proposed algorithms. It is to note that the contribution of the neighborhoods is analyzed independently in the SA and VND phases. For the case of the SA neighborhoods, the analysis considered the number of times a move was accepted (independently if it improved the solution or not) over the total number of SA moves accepted, while for the case of VND the analysis considered the number of times a move was applied improving the current solution over all the VND applied moves. Regarding the SA phase (see Figure 2.4.a), as it is possible to note, the contribution of each neighborhood for the SA-VND0 and the SA-VND2 algorithms is similar, presenting a dominance in the use of neighborhoods of Group *ii*), while for the SA-VND1 algorithm the contribution of all the neighborhoods is similar, with a slightly dominance of the neighborhoods in Group *i*). The above can be explained by considering the values of the parameter GP, which is equal to 1 for the algorithms SA-VND0 and SA-VND2, and equal to 2 for the algorithm SA-VND1. When $GP = 1$ all the neighborhoods have the same probability of being selected, and as it has been mentioned before, the neighborhoods in Group *ii*) include an embedded local search procedure, which implies that the generated move has potentially good quality, and thus higher probability of being accepted. On the other hand, when $GP = 2$, the neighborhoods in Group *i*) have the double of probabilities of being selected in comparison to the neighborhoods in Group *ii*). With respect to the VND phase (see Figure 2.4.b), due to the descent design of the search strategy the neighborhoods explored at the beginning contribute the most, and this behavior is shared by the three proposed algorithms. It is possible to note that in the SA-VND1 algorithm the contribution of the *insertion* neighborhood is larger in comparison to the other two algorithms, which implies a smaller contribution for the other neighborhoods. This situation is explained by the nature of algorithm SA-VND1, in which each time an improvement is found the search is immediately re-started from the *insertion* neighborhood. On the other hand, the algorithms SA-VND0 and SA-VND2 allow to explore in a deeper way the other neighborhoods, which can be a possible explanation for the better results obtained by these two algorithms in comparison to the SA-VND1 algorithm.

According to the previous experiments, all the neighborhoods provide a contribution (whose magnitude depends either on their position in the VND or on the probability of selection in the SA) on the global performance of the algorithm. Thus, an experiment corresponding to the removal of each neighborhood was carried out in order to determine the effects of this action on the performance of the algorithms, both in terms of computing time and solution quality. Table 2.8 presents the global average results of this experiment by considering 5 runs for each

instance. In order to evaluate the effect of removing each neighborhood, the following criteria were considered for the Global case (average results considering the three algorithms SA-VND0, SA-VND1, and SA-VND2):

- *Best*: Best solution value found.
- *Avg*: Average solution value (computed over 5 runs).
- *time* (avg): Average computing time.
- Gap_B: Percentage gap between *Best* and the new best known solution value (reported in columns BKS in Tables (2.2-2.4)), computed as $Gap_B = 100\frac{(Best-BKS)}{BKS}$.
- Gap_{BKS0}: Percentage gap between *Best* and the best known solution value found by the current state-of-the-art algorithms (reported in columns BKS_0 in Tables (2.2-2.4)), computed as $Gap_{BKS0} = 100\frac{(Best-BKS_0)}{BKS_0}$.
- *leq* BKS: The number of instances for which the best solution value *Best* found by the proposed algorithm is better than or equal to BKS.
- *leq* BKS_0: The number of instances for which the best solution value *Best* found by the proposed algorithm is better than or equal to BKS_0.

The mentioned criteria are relevant for analyzing the performance of the algorithms with respect to both the currently published algorithms and the original version of the proposed metaheuristics. According to the results, by removing the neighborhoods from the VND procedure reduces the computing time, but also affects the quality of the average solution. The reduction in computing time depends on the position of the neighborhood in the VND procedure, thus, by removing the *insertion* or the *swap* neighborhoods it is possible to achieve the largest reduction in computing time. By removing the other neighborhoods in the VND procedure the reduction in computing time is marginal. The results regarding the *Avg* values also indicate that the neighborhoods *insertion* and $2-opt$ are essential in the VND procedure in order to consistently provide good quality solutions. By comparing the original algorithms with each of the versions obtained by removing each neighborhood from the VND procedure, it is possible to note that the original version of the algorithms is the best for all the criteria but one. On the other hand, the removal of the neighborhoods in Group i) from the random phase (SA) in general does not affect the performance of the algorithms, while the removal of the neighborhoods in Group ii) affects negatively the performance of the algorithms in terms of solution quality. It is to note also that the computing times are generally not reduced by removing neighborhoods in the SA procedure. this occurs because the number of random moves evaluated is not changed, thus, when a neighborhood is deleted the algorithms evaluate the same number of moves from the rest of the neighborhoods. By considering all the neighborhoods, the one that impacts the most on the performance of the algorithms (in terms of solution quality) is the *DepotOpenClose*. In fact, this neighborhood affects the strategic part of the problem, i.e. the location of the depots, and, when it is not considered, the search avoids an important part of the solution space. By comparing the original algorithms with each of the versions obtained by removing each neighborhood from the SA procedure, it is possible to note that the original version of the algorithms is the best in all the criteria but two. By considering all the possible 12 versions, it is possible to note that the best Global performance is obtained by considering the original algorithms. The original algorithms provide the best values for 5 out of the 7 criteria.

Table 2.8: Global average results by removing each neighborhood.

Removed Neighborhood	Best	Avg	time (avg)	Gap_B	Gap_{BKS0}	leq BKS	leq BKS_0
None (original algorithms)	3667.8	**3702.0**	198.4	**0.57**	**-2.58**	**19.7**	**72.3**
insertion	3709.9	3758.9	**157.5**	1.36	-1.82	12.7	62.7
swap	3674.9	3710.5	178.8	0.75	-2.40	16.7	70.7
2-opt	3701.8	3742.1	196.0	0.94	-2.23	15.0	70.0
arc-swap	3673.0	3709.4	197.1	0.64	-2.52	18.0	71.3
$shift_{2-1}$	3672.5	3707.9	199.3	0.62	-2.53	17.7	69.7
RandomInsertion	**3667.0**	3702.7	205.9	0.61	-2.54	17.3	71.3
RandomSwap	3670.2	3709.5	193.8	0.66	-2.50	16.7	70.7
RandomTwoOpt	3678.0	3709.2	195.8	0.60	-2.56	17.0	71.7
DepotOpenClose	3936.2	3956.2	193.6	9.28	5.79	12.3	37.0
RouteSwap	3671.2	3705.3	204.9	0.65	-2.51	15.3	71.7
RouteRelocation	3721.0	3750.0	215.9	2.15	-1.06	12.7	56.7

According to the results presented, there are no reasons for excluding some of the proposed neighborhoods from the SA-VND framework presented in this work.

As it was mentioned before, the presented neighborhoods have been used in several routing/location-routing problems in the literature, some of them have also been used by previous algorithms for solving the LLRP. Indeed, all the neighborhoods used in the VND phase with exception of the $arc - swap$ and the $shift_{2-1}$, were also included in the RGA algorithm presented in [1], and all those neighborhoods but the $arc - swap$, the $shift_{2-1}$ and the inter-route $2 - opt$ were included in the GBILS algorithm [3]. Furthermore, the solution space of the MAs [1] is similar to the solution space of the three proposed metaheurisrtics. Despite the above, the proposed algorithms clearly outperform all the contenders in the literature in terms of solution quality. Thus, we can conclude that the combination of all the proposed ingredients of the metaheuristics presented in this work, i.e., the initial solution, the exploration (combining multi-neighborhood SA and VNDs), the LKH-3 procedure, and the search space (allowing infeasible solutions), are more effective than the methodologies previously used in the literature.

2.4.5 A statistical comparison of the proposed metaheuristics

According to the results reported in the previous sections, it is evident that, for what concerns the quality of the solutions found, the proposed metaheuristics overcome the state-of-the-art heuristic algorithms MA, RGA, and GBILS. Nevertheless, it is not clear which of the proposed algorithms is the dominant one. In order to obtain statistical information about the means of the algorithms, we conducted hypothesis tests using three t-tests, considering 30 runs for each instance, comparing: SA-VND0 vs SA-VND1, SA-VND0 vs SA-VND2 and SA-VND1 vs SA-VND2. The t-test was chosen in order to perform the same statistical analysis proposed in [1]. The results indicate that for 60 instances, with a 5% of significance level, there are no

Table 2.9: Summary of the results of the hypothesis tests for comparing the means of the algorithms.

Preferred Algorithm(s)	Instances
SA-VND0	111212, 121112, 133112, 100-10-1b
SA-VND2	131122, 133122, 133222, 100-10-2b, Christ-100-10
SA-VND0 and SA-VND1	123212, 200-10-3, Christ-100-10
SA-VND0 and SA-VND2	111122, 111122, 112122, 122122

statistically significant differences among the means of the algorithms. The 16 instances for which there are differences among the means of the algorithms are presented in Table 2.9, with their preferred algorithm.

After analyzing the results, it is possible to conclude that the quality of the solutions provided by the three algorithms is similar for most of the studied instances. SA-VND1 is the algorithm with less preferences, while SA-VND0 and SA-VND2 presented different behaviors for few instances.

2.5 Conclusions and future research

An effective metaheuristic framework for the latency location routing problem (LLRP) was proposed. It combines the well known simulated annealing and variable neighborhood descent techniques.

The three proposed metaheuristics (SA-VND0, SA-VND1 and SA-VND2) were tested on the three classical LLRP benchmark data sets, with a total of 76 instances. Extensive computational experiments show that the proposed metaheuristics outperform the state-of-the-art heuristic algorithms MA, RGA (proposed in [1]), and GBILS (proposed in [3]), and the five exact methods (proposed in [3]) in terms of solution quality. Compared with the currently published algorithms, each of the proposed algorithms is able to improve or reach the best known solution value for 73 out of 76 instances. In addition, by neglecting the instances with a number of customers smaller than 50 (which can be easily solved to optimality by the MILP models), the average solution value found by each of the proposed algorithms is better than the best solution value obtained by algorithm GBILS for 70% of the remaining instances, and by algorithms MA and RGA for all but one of the remaining instances. For the small and medium size instances the proposed metaheuristics find several proved optimal solutions, and in some cases the average value was equal to the optimal one.

Despite the metaheuristics presented in this work are more time consuming than the algorithms MA, RGA and GBILS (considering the average computing time associated with one run), they are much more stable. The proposed metaheuristics are globally able to find target values in smaller computing times than those of the state-of-the-art heuristic algorithms. Thus, the presented metaheuristics can reach solutions with the same or better quality within smaller computing times than those of the currently published algorithms, and are able to achieve even better quality solutions when more computing time is allowed.

Comparing the three proposed algorithms, we can conclude that there are not statistically significant differences between their performances, nevertheless, SA-VND0 is the algorithm which requires the smallest computing time.

Based on the obtained results, it is possible to suggest as future directions to apply the

proposed methodology to other problems related to the LLRP. Some examples are: the multi-depot cumulative capacitated vehicle routing problem, the latency location routing problem with time windows, and other extensions of the cumulative capacitated vehicle routing problem and the facility location problem.

Appendix A: Optimal solution for the instance Christ-50-5

Route 1
D1 - 2 - 24 - 7 - 43 - 3 - 8 - 6 - 26
Route 2
D1 - 23 - 48 - 27 - 46 - 12 - 47 - 18
Route 3
D1 - 14 - 25 - 9 - 5 - 4 - 13 - 41 - 19 - 40
Route 4
D4 - 33 - 45 - 15 - 44 - 37 - 17 - 42
Route 5
D4 - 10 - 49 - 38 - 11 - 32 - 1 - 22 - 28 - 31
Route 6
D4 - 39 - 30 - 34 - 50 - 16 - 21 - 29 - 20 - 35 - 36

Appendix B: The effect of reducing the number of runs

As noted in the Section 2.4.2, the proposed metaheuristics are more time consuming and much more stable than the algorithms RGA and MA. This indicates that, in order to obtain good quality solutions with the proposed algorithms, it is not necessary to execute many runs for each instance. Therefore, it is possible to reduce the global computing time required by the proposed algorithms by reducing the number of runs executed for each instance. In the experiments described in the following, the number of runs for each instance is reduced to 10 and 5, and the results are compared with those obtained by the algorithms RGA and MA by considering 30 runs for each instance, the algorithm GBILS by considering 5 runs for each instance, and the five exact methods. In particular, when the number of runs for each instance is fixed to 10 (resp. to 5), the first 10 (resp. 5) random seeds, among the 30 created ones, are used. The summary of the results is presented in Tables 2.10 and 2.11 by considering the data sets: Tuzun-Burke (36 instances), Prodhon (30 instances), Barreto (10 instances), Barreto-NG (containing the 6 instances reported in [3]), Total (containing the 76 instances of the overall data set), and Total-NG (containing the 36 instances reported in [3]). The rows in Table 2.10 have the following meaning:

- $\#\ BKS$: The number of instances for which the algorithm finds the value BKS. When the number of runs for each instance is reduced, the values of BKS found with a larger number of runs are not considered.
- $\#\ Avg - BKS_0$: The number of instances for which the average solution value Avg of the considered algorithm is better than or equal to BKS_0.
- $\#\ Best - BKS_0$: The number of instances for which the best solution value $Best$ found by the proposed algorithm is better than or equal to BKS_0.

Table 2.10: Summarized results on # BKS, # $Avg-BKS_0$ and # $Best-BKS_0$ for the complete data set considering 30, 10 and 5 runs.

Data set	RGA	MA	GBILS	Exact methods	30 runs			10 runs			5 runs		
Tuzun-Burke (36 instances)					SA-VND0	SA-VND1	SA-VND2	SA-VND0	SA-VND1	SA-VND2	SA-VND0	SA-VND1	SA-VND2
# BKS	0	0	-	-	**14**	10	13	**14**	13	11	**15**	10	12
# $Avg - BKS_0$	0	0	-	-	**35**	**35**	**35**	**35**	**35**	**35**	**36**	**36**	35
# $Best - BKS_0$	-	-	-	-	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
Prodhon (30 instances)													
# BKS	1	2	4	12	**19**	14	14	16	**17**	14	**18**	15	13
# $Avg - BKS_0$	0	0	-	-	**20**	18	19	**20**	19	19	**21**	19	20
# $Best - BKS_0$	-	-	-	-	**27**	**27**	**27**	**27**	**27**	26	**27**	26	26
Barreto (10 instances)													
# BKS	1	1	-	-	**9**	7	7	7	7	**8**	**8**	7	7
# $Avg - BKS_0$	0	0	-	-	**9**	**9**	**9**	**10**	9	**10**	**10**	**10**	**10**
# $Best - BKS_0$	-	-	-	-	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Barreto – NG(6 instances)													
# BKS	1	1	5	5	**5**	3	3	3	3	**4**	**4**	3	3
# $Avg - BKS_0$	0	0	-	-	**5**	**5**	**5**	**6**	5	**6**	**6**	**6**	**6**
# $Best - BKS_0$	-	-	-	-	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**
Total (76 instances)													
# BKS	2	3	-	-	**42**	31	34	**37**	**37**	33	**41**	32	32
# $Avg - BKS_0$	0	0	-	-	**64**	62	63	**65**	63	64	**67**	65	65
# $Best - BKS_0$	-	-	-	-	**73**	**73**	**73**	**73**	**73**	72	**73**	72	72
Total - NG (36 instances)													
# BKS	2	3	9	17	**24**	17	17	**19**	**20**	18	**22**	18	16
# $Avg - BKS_0$	0	0	-	-	**25**	23	24	**26**	24	25	**27**	25	26
# $Best - BKS_0$	-	-	-	-	**33**	**33**	**33**	**33**	**33**	32	**33**	32	32

The columns in Table 2.11 have the same meaning as in Tables 2.2, 2.3 and 2.4. The only new column is *Gap avg*, which represents the percentage gap between Avg and BKS_0 for the considered instance. The columns report the average values computed with respect to all the instances of the considered data set. It is to note that negative values for *Gap avg* indicate that the corresponding average value is better than BKS_0

By reducing the number of runs to 10, the algorithms SA-VND0, SA-VND1 and SA-VND2 are able to improve or find the value BKS_0 for 73, 73 and 72 instances, respectively. The number of instances for which the average value is better than or equal to BKS_0 is almost the same, while the quality of the values of the best solutions is slightly reduced. The computing times are now reduced by 3 times, which implies that the algorithm SA-VND0 performs globally faster than the RGA and MA algorithms, and the other two proposed algorithms are more competitive in terms of computing time. On the other hand, by reducing to 5 the number of runs, the algorithm SA-VND0 is still able to improve or find the value BKS_0 for 73 instances, while the algorithms SA-VND1 and SA-VND2 are able to do that for 72 instances. The quality of the values of the best solutions is slightly reduced, nevertheless, for the three proposed algorithms, in over 80% of the instances the average value is still better than or equal to the value BKS_0. In this case, the computing times are reduced by around 6 times, which leads to conclude that all the proposed algorithms require smaller computing times than the algorithms RGA and MA to find better solutions. Despite the above, the algorithm GBILS is the fastest algorithm among all the analyzed ones, but is outperformed by the proposed algorithms in terms of solution quality. According to the reported results, when the number of runs is reduced, the algorithm SA-VND0 exhibits, among the three proposed metaheuristics, the most stable performance in terms of solution quality and the smallest computing times.

Table 2.11: Average results on solution quality and computing times for the complete data set by considering 30, 10 and 5 runs.

Data set	BKS	RGA				MA				GBILS			Exact methods			SA-VND0					SA-VND1					SA-VND2				
Tuzun-Burke (36 instances)		Best	Avg	time	Gap_B	Best	Avg	time	Gap_B	Best	time	Gap_B	Best	time	Gap_B	Best	Avg	time	Gap_B	Gap avg	Best	Avg	time	Gap_B	Gap avg	Best	Avg	time	Gap_B	Gap avg
30 runs	3826.3	4171.7	4544.5	1805.0	8.7	4028.4	4422.8	1861.2	5.0	-	-	-	-	-	-	**3835.3**	**3901.8**	5740.9	**0.2**	**-2.9**	3837.8	3909.5	8990.5	0.3	-2.7	3841.0	3902.2	8934.4	0.3	**-2.9**
10 runs	3834.9	-	-	-	8.4	-	-	-	4.8	-	-	-	-	-	-	**3842.8**	**3897.5**	1910.8	**0.2**	**-3.0**	3852.1	3908.4	2996.9	0.4	-2.8	3851.1	3900.1	2980.9	0.3	**-3.0**
5 runs	3839.1	-	-	-	8.3	-	-	-	4.7	-	-	-	-	-	-	**3847.8**	**3887.1**	952.5	**0.2**	**-3.2**	3868.4	3906.5	1494.8	0.6	-2.8	3860.0	3898.5	1487.6	0.4	-3.0
Prodhon (30 instances)																														
30 runs	1500.5	1602.0	1684.7	2021.0	6.1	1564.4	1610.3	1272.6	3.7	1546.3	**121.7**	2.3	1553.9	14596.4	2.7	**1503.0**	**1521.2**	3975.5	**0.1**	**-0.6**	1503.9	1522.3	6248.6	0.2	**-0.6**	1503.8	1521.8	5692.4	0.2	**-0.6**
10 runs	1503.1	-	-	-	6.0	-	-	-	3.6	-	-	2.2	-	-	2.6	1506.4	**1520.2**	1325.1	**0.1**	**-0.7**	**1505.8**	1522.0	2078.3	**0.1**	-0.6	1507.3	1522.1	1900.4	0.2	-0.6
5 runs	1504.1	-	-	-	6.0	-	-	-	3.6	-	-	2.1	-	-	2.5	1511.0	**1520.2**	661.0	0.3	**-0.7**	**1509.3**	1522.0	1039.7	**0.2**	-0.6	1509.4	1521.1	949.7	0.3	-0.6
Barreto (10 instances)																														
30 runs	9390.8	10533.2	12081.7	879.3	7.8	9841.9	10712.6	900.0	3.4	-	-	-	-	-	-	**9392.9**	9554.7	847.8	**0.1**	**-1.8**	9404.1	**9549.4**	921.3	**0.1**	**-1.8**	9396.6	9554.7	1252.6	**0.1**	**-1.8**
10 runs	9406.3	-	-	-	7.6	-	-	-	3.2	-	-	-	-	-	-	9457.3	9586.6	282.4	0.2	-1.7	**9415.9**	**9546.5**	306.2	**0.1**	-1.7	9433.8	9551.1	416.4	**0.1**	**-1.8**
5 runs	9406.8	-	-	-	7.6	-	-	-	3.2	-	-	-	-	-	-	9509.1	9592.1	140.7	0.2	-1.7	**9416.1**	**9508.9**	152.2	**0.1**	**-1.9**	9436.5	9522.7	206.3	**0.1**	-1.8
Barreto - NG (6 instances)																														
30 runs	2282.1	2451.8	2741.9	542.0	4.1	2423.7	2540.8	628.5	2.0	2400.7	**91.7**	0.8	2410.6	4047.4	1.1	**2368.7**	2375.3	548.1	**0.0**	-0.6	2371.1	2376.6	568.4	0.1	-0.6	2369.3	**2374.4**	819.2	**0.0**	**-0.7**
10 runs	2285.7	-	-	-	4.1	-	-	-	2.0	-	-	0.8	-	-	1.1	2370.9	2375.1	182.5	**0.0**	-0.6	2373.9	2378.2	189.6	0.1	-0.6	**2370.0**	**2374.6**	272.1	**0.0**	**-0.7**
5 runs	2286.4	-	-	-	4.0	-	-	-	1.9	-	-	0.8	-	-	1.0	**2370.9**	**2375.0**	90.8	**0.0**	**-0.6**	2373.9	2375.9	94.4	0.1	**-0.6**	2374.6	2376.9	134.7	0.1	**-0.6**
Total (76)																														
30 runs	3640.4	3994.4	4443.7	1768.5	7.6	3820.7	4173.9	1502.4	4.3	-	-	-	-	-	-	**3645.9**	**3705.9**	4400.2	**0.1**	**-1.9**	3649.0	3709.3	6846.4	0.2	-1.8	3649.4	3706.3	6643.9	0.2	**-1.9**
10 runs	3647.5	-	-	-	7.4	-	-	-	4.1	-	-	-	-	-	-	3659.3	3707.7	1465.3	**0.2**	**-1.9**	**3658.0**	3708.3	2280.3	0.3	-1.8	3660.5	**3705.0**	2216.9	0.3	**-1.9**
5 runs	3650.0	-	-	-	7.3	-	-	-	4.0	-	-	-	-	-	-	3670.3	3703.5	730.6	**0.3**	**-2.0**	3667.2	3702.4	1138.5	0.4	-1.8	**3665.9**	**3700.1**	1106.7	**0.3**	-1.9
Total – NG (36)																														
30 runs	1665.1	1743.6	1865.9	1774.5	5.8	1707.6	1769.8	1165.3	3.5	1688.7	**116.7**	2.1	1696.7	12838.2	2.4	**1647.3**	**1663.6**	3404.3	**0.1**	**-0.6**	1648.5	1664.7	5301.9	**0.1**	**-0.6**	1648.0	1663.9	4880.2	0.2	**-0.6**
10 runs	1667.9	-	-	-	5.7	-	-	-	3.3	-	-	2.0	-	-	2.3	**1650.5**	**1662.7**	1134.6	**0.1**	**-0.7**	**1650.5**	1664.7	1763.5	**0.1**	-0.6	1651.1	1664.2	1629.0	0.2	-0.6
5 runs	1668.8	-	-	-	5.6	-	-	-	3.3	-	-	1.9	-	-	2.3	1654.3	**1662.7**	566.0	**0.2**	**-0.7**	**1653.4**	1664.3	882.2	**0.2**	-0.6	1653.6	1663.7	813.9	0.3	-0.6

Bibliography

[1] M. Moshref-Javadi and S. Lee, "The latency location-routing problem," European Journal of Operational Research, vol. 255, no. 2, pp. 604–619, 2016.

[2] S. U. Ngueveu, C. Prins, and R. Wolfler Calvo, "An effective memetic algorithm for the cumulative capacitated vehicle routing problem," Computers & Operations Research, vol. 37, no. 11, pp. 1877–1885, 2010.

[3] S. Nucamendi-Guillén, I. Martínez-Salazar, S. Khodaparasti, and M. E. Bruni, "New formulations and solution approaches for the latency location routing problem," Computers & Operations Research, vol. 143, p. 105767, 2022.

[4] J. F. Sze, S. Salhi, and N. Wassan, "The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search," Transportation Research Part B: Methodological, vol. 101, pp. 162–184, 2017.

[5] S. Salhi and G. K. Rand, "The effect of ignoring routes when locating depots," European Journal of Operational Research, vol. 39, no. 2, pp. 150–156, 1989.

[6] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," science, vol. 220, no. 4598, pp. 671–680, 1983.

[7] A. Duarte, N. Mladenovic, J. Sánchez-Oro, and R. Todosijević, Variable Neighborhood Descent, pp. 341–367. Cham: Springer International Publishing, 2018.

[8] O. Dukkanci, B. Y. Kara, and T. Bektaş, "The green location-routing problem," Computers & Operations Research, vol. 105, pp. 187–202, 2019.

[9] K. Corona-Gutiérrez, S. Nucamendi-Guillén, and E. Lalla-Ruiz, "Vehicle routing with cumulative objectives: a state of the art and analysis," Computers & Industrial Engineering, vol. In Press, Journal Pre-proof, p. 108054, 2022.

[10] G. Nagy and S. Salhi, "Location-routing: Issues, models and methods," European Journal of Operational Research, vol. 177, no. 2, pp. 649–672, 2007.

[11] R. B. Lopes, C. Ferreira, B. S. Santos, and S. Barreto, "A taxonomical analysis, current methods and objectives on location-routing problems," International Transactions in Operational Research, vol. 20, no. 6, pp. 795–822, 2013.

[12] C. Prodhon and C. Prins, "A survey of recent research on location-routing problems," European Journal of Operational Research, vol. 238, no. 1, pp. 1–17, 2014.

[13] R. Cuda, G. Guastaroba, and M. G. Speranza, "A survey on two-echelon routing problems," Computers & Operations Research, vol. 55, pp. 185–199, 2015.

[14] M. Drexl and M. Schneider, "A survey of variants and extensions of the location-routing problem," European Journal of Operational Research, vol. 241, no. 2, pp. 283–308, 2015.

[15] M. Schneider and M. Drexl, "A survey of the standard location-routing problem," Annals of Operations Research, vol. 259, no. 1, pp. 389–414, 2017.

[16] M. Albareda-Sambola and J. Rodríguez-Pereira, "Location-routing and location-arc routing," in Location science, Springer, 2019.

[17] S. T. W. Mara, R. Kuo, and A. M. S. Asih, "Location-routing problem: a classification of recent research," International Transactions in Operational Research, 2021.

[18] K. Helsgaun, "An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems," Roskilde University, 2017.

[19] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," European journal of operational research, vol. 126, no. 1, pp. 106–130, 2000.

[20] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," Operations research, vol. 21, no. 2, pp. 498–516, 1973.

[21] J. MacQueen et al., "Some methods for classification and analysis of multivariate observations," in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[22] J. W. Escobar, R. Linfati, and P. Toth, "A two-phase hybrid heuristic algorithm for the capacitated location-routing problem," Computers & Operations Research, vol. 40, no. 1, pp. 70–79, 2013.

[23] F. Y. Vincent, S.-W. Lin, W. Lee, and C.-J. Ting, "A simulated annealing heuristic for the capacitated location routing problem," Computers & Industrial Engineering, vol. 58, no. 2, pp. 288–299, 2010.

[24] R. Bellio, S. Ceschia, L. Di Gaspero, and A. Schaerf, "Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling," Computers & Operations Research, vol. 132, p. 105300, 2021.

[25] R. M. Rosati, M. Petris, L. Di Gaspero, and A. Schaerf, "Multi-neighborhood simulated annealing for the sports timetabling competition itc2021," Journal of Scheduling, vol. 25, no. 3, pp. 301–319, 2022.

[26] N. A. Kyriakakis, M. Marinaki, and Y. Marinakis, "A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem," Computers & Operations Research, vol. 134, p. 105397, 2021.

[27] D. Tuzun and L. I. Burke, "A two-phase tabu search approach to the location routing problem," European Journal of Operational Research, vol. 116, no. 1, pp. 87–99, 1999.

[28] C. Prins, C. Prodhon, and R. Wolfler Calvo, "Nouveaux algorithmes pour le problème de localisation et routage avec contraintes de capacité," in MOSIM'04 (4ème Conf. Francophone de Modélisation et Simulation), (Nantes, France), 2004.

[29] S. Barreto, "Análise e modelização de problemas de localização-distribuição [analysis and modelling of location-routing problems," in Ph.D. Thesis, University of Aveiro, (Aveiro, Portugal), 2004.

[30] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," Operations Research Perspectives, vol. 3, pp. 43–58, 2016.

[31] M. E. Bruni, S. Khodaparasti, I. Martínez-Salazar, and S. Nucamendi-Guillén, "The multi-depot k-traveling repairman problem," Optimization Letters, vol. 16, pp. 2681–2709, 2022.

3 An iterated local search algorithm for latency vehicle routing problems with multiple depots

Abstract

The Multi-depot Cumulative Capacitated Vehicle Routing Problem (MDCCVRP) extends the recently proposed Cumulative Capacitated Vehicle Routing Problem (CCVRP). The aim is to minimize the sum of the arrival times at the customers considering a fleet of N_v capacitated vehicles and a set of N_d uncapacitated depots. This chapter proposes valid lower bounds and a novel metaheuristic algorithm for the solution of the MDCCVRP. The initial solution is obtained by combining different heuristic approaches, while the improving phase consists of an iterated local search algorithm (ILS). Computational experiments on 78 MDCCVRP benchmark instances show that the proposed algorithm is able to find, within reasonable computing times, solution values globally better than those obtained by the state-of-the-art heuristic algorithms. For challenging instances (having a large number of customers and a small fleet size), the algorithm can find, within short computing times, solutions globally better than those obtained by the published exact algorithms. The proposed algorithm has also been applied to the recently introduced Multi-depot k-traveling Repairman Problem (MDk-TRP) and the Latency Location Routing Problem (LLRP). The MDk-TRP is a particular case of the MDCCVRP arising when the vehicles are uncapacitated, while the LLRP is a generalization of the MDCCVRP in which, at most, p of the N_d available depots can be used. The computational experiments performed on 87 MDk-TRP benchmark instances and 76 LLRP benchmark instances show that the proposed algorithm globally outperforms the state-of-the-art metaheuristic algorithms for what concerns both the solution quality and the computing time. For large-size instances, the computing time required to provide a good quality solution is considerably smaller than that required by the previously published heuristic and exact algorithms. For all the problems, the proposed algorithm is able to find better solution values than those obtained by the respective state-of-the-art metaheuristic algorithms when it is executed for the same computing time as the respective competitor.

Keywords: cumulative routing; ILS; latency; MDCCVRP ;MDk-TRP; LLRP

3.1 Introduction

The Multi-depot Cumulative Capacitated Vehicle Routing Problem (MDCCVRP) is a variant of the well-known Multi-depot Vehicle Routing Problem (MDVRP), in which, instead of minimizing the total travel time of the system, the aim is to minimize its global latency. The latency can be defined as the sum of the arrival times at the customers, and it is a metric used for defining customer satisfaction. Although the MDVRP is a well-known variant of the vehicle routing

This chapter is based on the paper: Osorio-Mora A, Escobar JW, Toth P. An iterated local search algorithm for latency vehicle routing problems with multiple depots. Computers & Operations Research. 2023 Jun. doi: https://doi.org/10.1016/j.cor.2023.106293

problems (a survey on MDVRPs can be found in [1]), the classical objective function is not appropriate for solving real-world cases concerning customer satisfaction. Indeed, it has been proved that the optimal solutions for classical routing problems lead to sub-optimal solutions for cumulative (latency) routing problems [2]. This work suggests that new methods must be developed for solving effectively cumulative routing problems.

Research on cumulative capacitated vehicle routing problems (CCVRP) dates from 2010 with the seminal work of Ngueveu, Prins, and Wolfler Calvo [3]. The CCVRP is a particular case of the MDCCVRP considering only one depot. For a recent survey paper on cumulative-based vehicle routing problems, focusing on the CCVRP, the reader is referred to [4]. Although the CCVRP was relatively recently introduced, the research on latency routing problems started in the early 90s with the traveling repairman problem (TRP) [5], also known as delivery man problem [6] and minimum latency problem (MLP) [7]. The TRP consists of finding the best sequence for visiting a set of customers considering a single vehicle, such that the latency is minimized.

Some natural extensions of the TRP have been studied by different researchers. The k-TRP [8] corresponds to a generalization of the TRP by considering k uncapacitated vehicles. This problem has also been called m-MLP [9]. The MDk-TRP [10] is a generalization of the k-TRP by considering multiple depots. The MDk-TRP is also a particular case of the MDCCVRP in which the vehicles have a capacity large enough to serve the demand of the customers. A generalization of the MDCCVRP is the latency location routing problem (LLRP) [11], arising when at most a fixed number of the available depots can be used. Figure 3.1 shows a diagram pointing out the relations among the previously mentioned problems. All these problems are NP-hard since they can be reduced to the TRP, which has been proved to be NP-hard [8, 3, 11, 12, 10].

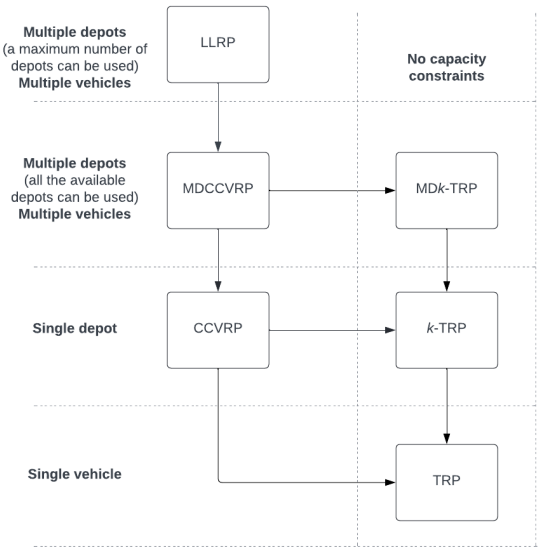


Figure 3.1: Relations between the different latency-based routing problems.

The MDCCVRP can be defined by considering a complete undirected graph $G = (V, E)$, where V corresponds to the set of nodes and E to the set of edges. The set V equals $V' \cup D$, with V' representing the set of N_c customers and D representing the set of N_d uncapacitated depots. Let also K be the set of N_v homogeneous vehicles, each with capacity Q. Each customer $i \in V'$ has a non-negative demand q_i (with $q_i \leq Q$). Each edge $(i, j) \in E$, with $i \neq j$, has an

associated non-negative travel time c_{ij}, which satisfies the triangular inequality. The problem consists of defining the routes to be performed by the vehicles, minimizing the sum of the arrival times at the customers. Each customer must be visited once. Each vehicle can perform one route. Each route starts from a depot and visits a subset of customers whose global demand cannot exceed the vehicle capacity Q. As in the classical MDVRP, the MDCCVRP does not consider the availability of the vehicles at each depot. Note that it is not mandatory to use all the depots. The objective of the cumulative routing problems is to minimize the sum of the arrival times at the customers. Hence the last edge of each route (connecting the last customer of the route with a depot) has not to be considered in evaluating the objective function [3]. Therefore, the routes can be considered as "open routes".

The notation of the problem is summarized in Table 3.1.

Table 3.1: Notation of the multi-depot cumulative capacitated vehicle routing problem.

Sets	
V' :	Set of N_c customers.
D :	Set of N_d identical uncapacitated depots.
V :	Set of nodes, $V = V' \cup D$.
K :	Set of N_v identical vehicles.
Parameters	
Q:	Capacity of the vehicles.
q_i:	Demand of customer i ($i \in V'$).
c_{ij}:	Travel time between nodes i and j ($i, j \in V', i \neq j$).
p:	Maximum number of used depots for the LLRP.

The MDCCVRP was introduced in [12], where a mixed-integer linear programming (MILP) formulation and a POPMUSIC matheuristic algorithm are proposed to solve the problem. The POPMUSIC method begins generating an initial solution through a greedy clustering algorithm. The idea is to improve the solution by partitioning the MDCCVRP into smaller sub-problems. The sub-problems are solved separately with the commercial solver CPLEX, and then included within the global matheuristic solution. Some relevant features of the MDCCVRP have been discussed in that work; in particular, the proof that the number of vehicles in the optimal solution is equal to $min\{N_v, N_c\}$, and the relations of the MDCCVRP with the TRPs.

In [13], a perturb-based local search (PLS) metaheuristic algorithm has been proposed to solve the MDCCVRP. A constructive heuristic based on the k-regrets insertion criterion [14] is used for finding the initial solution. Then, a local search procedure is applied by exploring six different moves under the first improvement criterion. Once no improvement can be reached, 2-opt and 2-exchange moves are applied to perturb the solution and explore a new search space. The efficiency of this approach is compared with the results previously presented in [12].

A branch-cut-and-price algorithm for solving the CCVRP and the MDCCVRP is proposed in [15]. The algorithm is able to provide the optimal solution for many small/medium size instances and high-quality solutions for large-size instances by fixing the maximum number of customers in each route. Two MILP formulations have been proposed in [16] to solve the LLRP. These formulations were also adapted to solve the MDCCVRP, and were able to find some optimal solutions for small/medium-size instances (with up to 50 customers) for both the LLRP and the MDCCVRP. The formulations were able to solve large MDCCVRP instances (with up to 192

customers) when a large number of vehicles is considered (in [12] it has been proved that these instances are easier than the same instances with a relative small number of vehicles).

Although the problem considered in [17] does not correspond to an MDCCVRP, it is important to point out the differences between these two problems. Indeed, the former problem has been defined by its authors as a cumulative multi-depot vehicle routing problem (Cu-MDVRP), considering it as a generalization of the cumulative vehicle routing problem (Cu-VRP) proposed in [18]. The Cu-VRP seeks to minimize the sum of the travel times of the traversed edges weighted by the load on the vehicle at the moment of traversing the edges. Nevertheless, the Cu-MDVRP presented in [17] seeks to minimize the sum of the arrival times at the customers weighted by their demands. Thus, it can be considered as a weighted version of the MDCCVRP. In addition, Cu-MDVRP considers a certain number of vehicles available at each depot, which is another feature that makes it different from the MDCCVRP. For more details about these two families of problems (Cu-VRPs and CCVRPs) the reader is referred to the literature review presented in [4].

The MDk-TRP was recently introduced in [10]. For its solution, the authors proposed two MILP models and a genetic algorithm (GA) under two different configurations. The formulations, which can solve to proven optimally several large-size instances (with over 200 customers), are based on the multi-level network approach. This approach was previously used to solve other related latency routing problems as the MLP [19], the k-TRP/m-MLP [20, 9], and the CCVRP [21]. More recently, as we already mentioned, it was also used to solve the LLRP and the MDCCVRP [16]. On the other hand, the GA is able to solve large-size instances within short computing times.

The LLRP, which is a combination of the facility location problem (FLP) and the CCVRP, was introduced in [11]. The LLRP can be considered as an extension of the MDCCVRP in which, at most, p of the N_d available depots can be used (i.e. opened). Two heuristic algorithms to solve efficiently the LLRP (a memetic algorithm (MA) and a recursive granular algorithm (RGA)) have been proposed in [11]. According to the reported computational experiments, MA performs better than RGA for the complete set of the considered instances. Recently, two MILP models, three enumerative algorithms, and a GRASP-based iterated local search algorithm (GBILS) have been proposed in [16] to solve the LLRP. The authors provide the optimal solution for several instances with up to 50 customers using the five exact methods, while the metaheuristic algorithm GBILS found globally better quality solutions than those obtained by the algorithms RGA and MA within short computing times. The best results on the benchmark instances currently considered for the LLRP have been reported for the three metaheuristic algorithms presented in [22]. These algorithms combine simulated annealing (SA) and variable neighborhood descent (VND) procedures. The main difference between the three proposed algorithms is the VND used strategy. The proposed approaches outperform the state-of-the-art algorithms for the LLRP, being able to find better quality solutions within comparable computing times.

Applications of multi-depot latency routing problems have also been studied, especially in post-disaster and customer-centric contexts. A problem in which the visit to affected areas must be planned after a natural disaster is studied in [23], where also the possibility of restoration of blocked paths is considered. The authors proposed a mixed-integer programming model and two heuristic algorithms based on the cluster-first-route-second approach for solving the problem. A bi-objective location routing problem under uncertainty applied to humanitarian logistics is studied in [24]. The problem considers time windows and a heterogeneous fleet of vehicles, while a risk-averse approach is used for minimizing the total cost and the latency of the system.

The problem was solved with a hybrid genetic algorithm. Bruni et al. [25] propose a VND matheuristic for the Drone Routing Problem in the context of last-mile delivery. The problem is formulated as a deterministic location-routing model and derives its robust counterpart under the travel time uncertainty.

This chapter proposes an iterated local search metaheuristic algorithm called M-ILS to solve effectively the MDCCVRP, the MDk-TRP, and the LLRP. The reported computational experiments on benchmark instances from the literature show that the proposed algorithm finds several current and new best-known solutions within computing times comparable with those required by the state-of-the-art algorithms proposed for the considered problems. Furthermore, the optimal solution values reported in [10] are rectified for several MDk-TRP instances. Finally, valid lower bounds for the MDCCVRP and the MDk-TRP are presented.

The paper is organized as follows. Section 3.2 describes the proposed metaheuristic and presents lower bounds for the MDCCVRP and the MDk-TRP. The computational results are reported and analyzed in Section 3.3. Finally, in Section 3.4, a summary of our findings and future directions are drawn.

3.2 Description of the proposed approach

This section presents an M-ILS metaheuristic approach for solving the MDCCVRP, the MDk-TRP, and the LLRP. Besides, valid lower bounds are proposed for the MDCCVRP and the MDk-TRP. The main body of the proposed approach (M-ILS algorithm) consists of two major phases: the construction phase and the improvement phase. The goal of the construction phase is to build an initial feasible solution s_0 (see Section 3.2.1). In the improvement phase, an Iterated Local Search (ILS) scheme, considering several diversification and local search procedures, is applied to improve the quality of the current solution. The ILS procedure starts by setting the current solution s_c, and the best feasible solution s_{bf} equal to s_0. It consists of three procedures executed for it_{max} iterations: a perturbation procedure, a local search procedure called LS, and a procedure combining the simulated annealing (SA) and the variable neighborhood descent (VND) frameworks (this procedure is called SA-VND). After the execution of the ILS, the well-known Lin–Kernighan-Helsgaun heuristic (LKH-3) [26] is used to solve a CCVRP (for the MDCCVRP and the LLRP) or a k-TRP (for the MDk-TRP) for each open depot, considering the best feasible solution. The local search procedure LS is applied to the solution obtained by the LKH-3 algorithm. Finally, for each route, a procedure called checking is applied. This procedure verifies if each route's first customer is assigned to its closest depot. If this is not the case, the closest depot is assigned to the corresponding route. The details of the ILS procedure are described in Section 3.2.2, and a summarized representation of the overall algorithm is presented in Algorithm 4. Table 3.2 shows the parameters used for the proposed approach.

Table 3.2: Algorithm's parameters

it_{max}	:	Number of iterations of the iterated local search.
it_{soft}	:	Frequency of the $Route - Relocation$ perturbation.
it_{hard}	:	Frequency of the $Configuration - Swap$ perturbation.
a	:	Percentage for the penalization for each unit exceeding the capacity of the vehicles.
t_0	:	Initial temperature in the SA-VND procedure.
t_f	:	Minimum temperature in the SA-VND procedure.
α	:	Cooling factor in the SA-VND procedure.

The key points for the success of the proposed approach are the correct selection of the depots using the heuristic procedure of the construction phase. Besides, the local search and diversification procedures within the improvement phase and the Lin–Kernighan-Helsgaun heuristic (LKH-3) allow an efficient exploration of the search space. Since the most critical decisions of the multi-depot variants of the single-depot vehicle routing problems are initially those concerning the use and assignment of the depots, a correct selection of the depots can reduce the search space for the improvement phase (avoiding local search procedures between depots). Also, starting from a good initial feasible solution allows for improving the current solution by applying the correct local search procedures. The previously mentioned procedures are described in more detail in the following subsections.

Algorithm 4: Main Scheme

Input: A MDCCVRP/MDk-TRP/LLRP instance, Algorithm parameters
Output: s_{bf} ($Best\ feasible\ solution$)
Constructive procedure — **return:** s_0 (initial solution)
Iterated local search:
$it = 0$
while $(it < it_{max})$ **do**
    step 1: Perturbation
    step 2: Local search (LS)
    step 3: SA-VND search procedure
    it=it+1
end
For each used depot solve a CCVRP/k-TRP applying the LKH-3 heuristic
Local search (LS)
Checking
return: s_{bf}

3.2.1 Construction phase: Initial solution

In this phase, we propose an efficient procedure to construct an initial feasible solution. The procedure is based on an approach that combines different heuristic procedures, including the LKH-3 heuristic. Besides, a cluster-based method is considered as starting point within the initial iterative framework. The initial solution s_0 is obtained by the following Constructive procedure, which generally finds good feasible solutions within short computing times:

-*Step 1*: Considering all the customers, construct the corresponding giant Traveling Salesman Problem (TSP) tour using the LKH-3 heuristic. Note that for the giant TSP, the global travel time to visit all the customers is minimized.

-*Step 2*: A good initial solution can be obtained by identifying clusters of customers. To this end, the giant tour is split into N_v clusters, as described below. We apply a clustering procedure by considering each customer as a "starting point", and by splitting the giant tour into groups of consecutive customers. For the MDCCVRP and the MDk-TRP, the splitting aims to balance the solution. The first ($N_c\ mod\ N_v$) clusters are composed of $\lceil \frac{N_c}{N_v} \rceil$ customers, while the remaining clusters are composed by $\lfloor \frac{N_c}{N_v} \rfloor$ customers. The idea of considering balanced solutions is based

on the valid lower bound LB2 presented in Section 3.2.3. For the LLRP, a different clustering procedure is applied. It consists of splitting the giant tour into groups of consecutive customers such that the total load of each cluster does not exceed the capacity of the vehicles. If the number of clusters created is larger than N_v, a repair procedure is applied to delete the least-loaded clusters until the number of clusters equals N_v. The customers are removed from the least-loaded clusters according to the order given in the clusters. Each customer belonging to a least-loaded cluster is removed from its current position and inserted in its best position in a different cluster, so as to minimize the score defined in equation 3.1:

$$ScoreIN_{ik}^{j} = \Delta instime_{j}^{ik} + \theta[max\{0, (dc_j + q_i) - Q\}] \tag{3.1}$$

where: $\Delta instime_{j}^{ik}$ represents the variation of the travel time of cluster j caused by the insertion of customer i in position k, dc_j represents the current load of cluster j, and θ represents a penalization parameter (large positive value). The process is repeated until all the least-loaded clusters are deleted.

-*Step 3*: If the total load of a cluster (say cluster j) exceeds the vehicle capacity, a swapping procedure is applied to two customers (say customers k and i, with $q_i < q_k$) with respect to their current clusters (say clusters j and l, respectively, with $j \neq l$), so as to minimize the following score:

$$\begin{aligned} ScoreSW_{ik}^{jl} = \Delta time_{j}^{ki} + \Delta time_{l}^{ik} + \theta[max\{0, (dc_j - q_k + q_i) - Q\} \\ + max\{0, (dc_l + q_k - q_i) - Q\}] \end{aligned} \tag{3.2}$$

where: $\Delta time_{j}^{ki}$ (resp. $\Delta time_{l}^{ik}$) represents the variation of the travel time of cluster j (resp. cluster l) caused by the exchange of the customers k and i. If no feasible splitting of the customers into N_v clusters is found by this swapping procedure, the exact algorithm MTP proposed in [27] is applied to the Bin Packing Problem (BPP) instance corresponding to the given MDCCVRP instance to obtain a set of at most N_v feasible clusters.

-*Step 4*: Let CL be the clusters set created in the previous step. For each depot $i \in D$ and for each cluster $j \in CL$, we define an allocation cost l_{ij}, which represents the total latency of the route composed by the customers in cluster j and depot i. This allocation cost is obtained by applying an intra-route local search ($IntraLS$) procedure to the path generated for the cluster (more details about this local search procedure are presented in Section 3.2.2).

Step 5: The best assignment of the clusters to the depots for the MDCCVRP and the MDk-TRP is obtained by assigning each cluster $j (j \in CL)$ to the depot i such that $l_{ij} = min\{l_{hj} : h \in D\}$. Thus, the latency of the solution is given by $\sum_{j \in CL} \min_{i \in D}\{l_{ij}\}$. We define a depot $i \in D$ as "used" if at least one cluster is assigned to i. A depot configuration corresponds to a binary vector $Configuration$ of size N_d that indicates if depot $i \in D$ is used ($Configuration_i = 1$) or not ($Configuration_i = 0$). All the previously mentioned information is stored in this *Step*.

For the LLRP, the best assignment of the clusters to the depots is obtained by solving the integer linear programming (ILP) model (3.3)-(3.8). We introduce two sets of binary variables,

where A_{ij} is equal to 1 if cluster j is assigned to depot i ($i \in D, j \in CL$), and y_i is equal to 1 if depot i ($i \in D$) is opened.

$$min \sum_{i \in D} \sum_{j \in CL} l_{ij} A_{ij} \tag{3.3}$$

$$\sum_{i \in D} A_{ij} = 1 \qquad \forall j \in CL \tag{3.4}$$

$$A_{ij} \leq y_i \qquad \forall i \in D, \forall j \in CL \tag{3.5}$$

$$\sum_{i \in D} y_i \leq p \tag{3.6}$$

$$A_{ij} \in \{0, 1\} \qquad \forall i \in D, \forall j \in CL \tag{3.7}$$

$$y_i \in \{0, 1\} \qquad \forall i \in D \tag{3.8}$$

Where l_{ij} is the latency of the route composed by the customers in cluster j and depot i (see *Step 4*), dc_j is the global demand of cluster j, and p is the maximum number of depots to be opened.

The objective function (3.3) seeks to minimize the total latency. Constraints (3.4) ensure that each cluster is allocated to exactly one depot. Constraints (3.5) impose that the clusters can be allocated only to open depots. Equation (3.6) ensures that the maximum number of open depots is at most p. Finally, constraints (3.7-3.8) define the domain of the variables.

-*Step 6*: Note that there are exactly N_c different possibilities to split the giant tour, since in the clustering procedure, the definition of the clusters depends only on the choice of the first customer (starting point), and the clustering procedure is applied N_c times, by considering each of the N_c customers as the starting point. *Steps 2 - 5* are repeated until all the solutions corresponding to the possible splittings of the customers into clusters are evaluated.

A list of promising depot configurations stores all the configurations obtained at *Step 5*, the number of times that each configuration is selected, and the allocation of the clusters to the depots that provides the minimum latency for that configuration. At each of the N_c iterations, if the current $Configuration$ has not yet been stored in the list, it is stored with the respective latency and the allocation of the clusters to the depots. On the other hand, if the current $Configuration$ has been already stored in the list, the number of times that the corresponding depot configuration has been selected is updated, and if the latency associated with the new allocation is smaller than that previously stored, the best latency, and the respective allocation are updated. At the end of the N_c iterations, the solution corresponding to the depot configuration with the minimum latency is selected. Each cluster is considered an open route, which starts from the assigned depot, and the sequence of the customers is as done at *Step 4*. Finally, the list of the promising depot configurations is sorted according to the number of times that each configuration was selected, putting in the first positions those configurations that were selected more times. This list will be used in the perturbation procedure described in Section 3.2.2. For the LLRP, a splitting procedure is applied to add new vehicles if the number of open routes currently created is smaller than N_v. The splitting procedure is performed based on the idea that the total latency decreases by adding new routes for the same or different depots. It consists of a local search procedure with the following steps.

- Select the route r containing the longest edge (i, j) (where i and $j \in V'$).

- Split the route r (starting from depot h) by removing edge (i, j). Two new sub-routes ($r1$ and $r2$) are created. $r1$ is the sub-route starting from depot h and composed of the customers belonging to route r until customer i. $r2$ is the sub-route composed by the customers of route r from customer j to the final customer of route r.
- Assign sub-route $r2$ to the best depot by considering its current or its reverse sequence, so as to minimize the corresponding latency.
- The procedure is performed until the number of routes of the current solution equals N_v.

-*Step 7*: Apply the LKH-3 heuristic for each depot with its assigned routes, solving a CCVRP or a k-TRP depending if the problem to solve is a MDCCVRP/LLRP or a MDk-TRP, respectively.

-*Step 8*: Apply the local search procedure LS until no improvement is found (for more details about the procedure LS, see Section 3.2.2). This procedure allows infeasible solutions in terms of vehicle capacity. These solutions are penalized by using a factor a, defined as a percentage of the solution value provided at the end of *Step 7*.

Although there are similarities between the algorithm proposed in this work and those proposed in [28, 29, 30], substantial differences are pointed out in the following. The major difference is given by the improvement phase of the proposed algorithm (which will be described in the next section) since all the mentioned works presented a tabu-search-based approach, while this work proposes an iterated local search algorithm. Furthermore, due to the differences in the cumulative and classical vehicle routing problems, there are several differences concerning the construction of the initial solution and the diversification/intensification strategies between this work and the mentioned works. Regarding the construction of the initial solution, in *Step 4*, the mentioned works use the LKH heuristic to solve a TSP and to calculate the values l_{ij}, while in this work, the $IntraLS$ procedure was specifically designed for solving a TRP. In preliminary computational experiments, an approach where the TRPs were solved using the LKH-3 heuristic was tested, but it led to extremely high computing times for large-size instances. In addition, the construction of the initial solution presented in the mentioned papers does not ensure that $min\{N_c, N_v\}$ vehicles will be used, while the procedure proposed in this work does it. Another important difference is the inclusion of the binary vector storing the promising depot configurations. It is noted that in the previously mentioned works, the best configuration of the depots is first selected, and then no change of the used depots is performed. This situation may lead to skipping promising parts of the search space.

3.2.2 Improvement Phase: Iterated local Search algorithm (ILS)

In this phase, the algorithm tries to improve the initial solution s_0 by applying an Iterated Local Search (ILS) procedure. The ILS algorithms have been successfully applied to a wide number of combinatorial optimization problems, and the main idea is to explore new regions of the solution space by applying a perturbation when a local optimum is reached. For further details about the ILS algorithms the reader is referred to [31].

For the proposed ILS, the current solution s_c and the best feasible solution s_{bf} are initially equal to the initial solution s_0. The three steps of the ILS are described in this section. Note that the local search procedure LS (step 2) is explained before the perturbation step since the neighborhoods and the local search procedures are used in the three steps of the algorithm.

Local search

In this section, the search space and the neighborhoods are described. The proposed algorithm accepts solutions infeasible with respect to the vehicle capacity to avoid local optima and extend the search space. Thus, the value of the objective function $f(s_c)$ of a solution s_c, feasible or not, is given by the following formula:

$$f(s_c) = \bar{f}(s_c) + af(s_0)\Delta QV \tag{3.9}$$

Where $\bar{f}(s_c)$ is the sum of the arrival times at the customers, $f(s_0)$ is the value of the objective function corresponding to the initial solution s_0 found at *Step 8* of the constructive procedure, and ΔQV is the total amount of load violating the capacities of the vehicles. It is noted that for the feasible solutions the second term of (3.9) is equal to zero, and for the case of the MDk-TRP, this term is always equal to zero. The best improvement strategy is used in the local search-based procedures.

The proposed algorithm executes the following five types of moves: $insertion$, $swap$, $2-opt$, $arc-swap$, and $shift_{2-1}$. All the neighborhoods, except $shift_{2-1}$, can be applied for the intra-route and the inter-route cases. For the inter-route case, the moves can be applied for routes starting from the same or from different depots. The neighborhoods are the following ones:

- $insertion$: A customer i is transferred from its current position to another position just after node j. Note that the selected customer can be moved to a different position in the same or in different routes.

- $swap$: Two customers (i and j) exchange their positions, either in the same route or between different routes.

- $2-opt$: This move is a classical version of the well-known $2-opt$ move for the TSP, in which two non-consecutive edges are removed, and the routes are reconnected differently. Note that if the two selected edges are in the same route, the two opt move is equivalent to that described by Lin and Kernighan [32]. In particular, two edges (i,j) and (k,l) are deleted, and two new edges are created. When the move is applied to edges belonging to the same route, the edges (i,j) and (k,l) are deleted, then the edges (i,k) and (j,l) are created, and the connection from k to j is reversed. On the other hand, when the move is related to two different routes, a crossing is applied: the edges (i,j) and (k,l), with the edge (i,j) in route 1 and the edge (k,l) in route 2, are deleted, then the edges (i,l) and (k,j) are created, and the initial customers of routes 1 and 2 (until nodes i and k, respectively) are merged with the final customers of routes 2 and 1 (from customers l and j, respectively).

- $arc-swap$: Two pairs of consecutive customers (i,j) and (k,l) are swapped with respect to their current positions. The two pairs of customers can belong to the same or to different routes.

- $shift_{2-1}$: Two consecutive customers (i,j) assigned to route $r1$ exchange their current positions with that of the customer k in the route $r2$, with $r1 \neq r2$.

The local search procedure LS (step 2), calls for exploring all the mentioned neighborhoods and applying the move which improves the most the current solution. The procedure stops when no improvement move is found.

Perturbation procedure

Since the ILS procedure can fail in finding a move to improve the current solution, the algorithm tries to escape from a local optimum by perturbing the current solution. The perturbation procedure considers three possible perturbations applied with different frequencies. The "less-aggressive" perturbations are called $Route - Swap$ and $Route - Relocation$, and the "most aggressive" one is called $Configuration - Swap$. $Route - Relocation$ is applied every $iter_{soft}$ iterations, $Configuration - Swap$ every $iter_{hard}$ iterations, while the $Route - Swap$ is applied at each iteration if no other perturbation is applied. The descriptions of the perturbations are given in the following paragraphs (where it is assumed that each random choice is performed by considering the same probability with respect to the possible choices):

- $Route - Swap$: We use an exchange scheme involving two routes. The procedure randomly selects two routes $r1$ and $r2$ belonging to two different depots i and j, respectively. A new solution s is obtained by considering the following move: remove the route $r1$ from the depot i and assign it to the depot j; remove route $r2$ from the depot j and assign it to the depot i. Since the new routes may not generate good solutions, an intra-route local search procedure ($IntraLS$) is applied to $r1$ and $r2$. The $IntraLS$ procedure sequentially explores each *insertion*, *swap*, and $2-opt$ neighborhood for the considered route until no improvement is found for the considered neighborhood.

- $Route - Relocation$: This perturbation procedure randomly selects a depot i with more than one route assigned. Then, the procedure randomly selects a route $r1$ belonging to the depot i. A new solution s is obtained by considering the following move: remove the route $r1$ from the depot i and assign it to a different depot j, which is randomly selected from all the remaining used depots. Then, the $IntraLS$ procedure is applied to the route $r1$.

- $Configuration - Swap$: This perturbation procedure swaps the current depot configuration (called $C1$) with the first one in the list of promising configurations (called $C2$). The list is sorted according to the number of times each configuration has been selected for the initial solution. Each time a configuration is evaluated, it is removed from the list. After picking $C2$, *Steps 7 - 8* of the constructive procedure are applied to construct the new solution. It is to note that, before applying the swapping, the LKH-3 heuristic and the LS procedure are applied to the best solution found during the exploration of the configuration $C1$.

For the exceptional cases in which the depot configuration considers only one available depot (with assigned routes), the perturbations $Route - Swap$ and $Route - Relocation$ are replaced by random moves applied under a simulated annealing framework described in Section 3.2.2. These random moves are applied to the current solution. If there are no more promising configurations to evaluate, $Configuration - Swap$ is skipped, and $Route - Relocation$ is applied to the best feasible solution. In this case, the perturbation procedure is not applied to the current solution. Note that $Route - Relocation$ cannot be applied when all the available depots in the current configuration have only one associated route.

The idea of applying different levels of aggressiveness in the perturbations is based on the fact that if only the "less-aggressive" perturbations are applied, the algorithm stacks into local-optimum solutions. After the application of $Route - Relocation$ or $Configuration - Swap$, the proposed local search operators cannot find the same local-optimum solution found previously. Indeed, these perturbation procedures change the allocation of routes to depots.

The SA-VND search procedure

The SA-VND procedure is presented in Algorithm 5.
This procedure starts by applying a simulated annealing framework (see Steps 1 to 11 of Algorithm 5). RandomMove(s_c) denotes the solution obtained by generating random moves with the same probability using the following neighborhoods: $insertion$, $swap$, and $2-opt$. The current temperature $temp$ is set to an initial temperature t_0. The SA procedure is applied until the minimum temperature t_f is reached ($temp \leq t_f$). A new solution s_p is generated by applying to the current solution s_c one of the mentioned random moves, and it is accepted as the new s_c if one of the following conditions holds (i.e., $AcceptanceCriteria(temp, s_p, s_c)$ is true): i) $\Delta f = f(s_c) - f(s_p) > 0$, where $f(s_c)$ and $f(s_p)$ are the objective function values of the solutions s_c and s_p, respectively; or ii) if $\Delta f \leq 0$ and $r < \exp^{(\Delta f/temp)}$ where r is a uniform random number in the interval $[0, 1]$. If $f(s_c) < f(s_{bf})$, and s_c is feasible (i.e., $IsFeasible(s_c)$ is $true$), the current solution is updated as the best feasible solution s_{bf} found so far. Then, the value of $temp$ is reduced according to a cooling factor α.

After the SA framework, a variable neighborhood descent (VND) procedure is applied to the current solution s_c (see Steps 12 to 30 of Algorithm 5). Denote by $Neigh = \{insertion,\ swap,\ 2-opt,\ arc-swap,\ shift_{2-1}\}$ the set of the five previously described neighborhoods, and consider $ng \in Neigh$ as the ng^{th} neighborhood of the current solution s_c. In addition, $sol(ng, s_c)$ denotes the solution obtained by exploring the neighborhood ng starting from the solution s_c. The neighborhoods are explored according to the order in which they are listed in the set $Neigh$. In the VND procedure, the exploration starts from the first neighborhood, which is explored until no improvement is found. Then, the search moves to the next neighborhood, and the process is repeated until the last neighborhood does not improve the current solution s_c. Otherwise, the search is restarted from the first neighborhood. If no improvement is found for all the neighborhoods, the VND procedure ends. Each time that a move is applied to the current solution s_c, if $f(s_c) < f(s_{bf})$, and s_c is feasible, the current solution s_c is updated as the best feasible solution s_{bf} found so far.

3.2.3 Lower bounds

This section describes two lower bounds, $LB1$ and $LB2$, proposed for both the MDCCVRP and the MDk-TRP. Note that lower bounds for the LLRP have been already proposed in [11]. The lower bounds for the MDCCVRP and the MDk-TRP generalize those proposed in [3] for the CCVRP.

Lower bound LB1: The first lower bound does not restrict the vehicle fleet size and considers one vehicle (i.e., one route) for each customer. The optimal solution for the unrestricted fleet problem is to assign each customer to its closest depot. The value of this solution is a valid lower bound for the considered problem:

$$LB1 = \sum_{i \in V'} \min_{j \in D} \{c_{ij}\} \tag{3.10}$$

Lower bound LB2: The second lower bound assumes a cardinality balanced solution, i.e., a solution where the routes visit an equal number of edges (i.e. of customers) or at most one edge of difference between the route with the largest number of edges and that with the smallest number of edges. Let us define NE_k as the number of edges associated with route $k, \forall k \in K$. The first ($N_c\ mod\ N_v$) routes are composed of $\lceil \frac{N_c}{N_v} \rceil$ edges, while the last $N_v - (N_c\ mod\ N_v)$ routes are composed by $\lfloor \frac{N_c}{N_v} \rfloor$ edges.

Algorithm 5: The SA-VND search procedure.

Input: $t_0,\ t_f,\ \alpha,\ s_c,\ s_{bf},\ Neigh$
Output: s_c $(New\ current\ solution)$, s_{bf} $(New\ current\ best\ feasible\ solution)$
`/* simulated annealing procedure */`
$temp = t_0$
while $(temp > t_f)$ **do**
  $s_p = \text{RandomMove}(s_c)$
  if $(AcceptanceCriteria(temp, s_p, s_c))$ **then**
    $s_c = s_p$
    if $(f(s_c) < f(s_{bf})\ and\ IsFeasible(s_c))$ **then**
      $s_{bf} = s_c$
    end
  end
  $temp = \alpha * temp$
end
`/* variable neighborhood descent procedure */`
$flag_{vnd} = true$
while $(flag_{vnd} = true)$ **do**
  $flag_{vnd} = false$
  for $(each\ ng \in Neigh)$ **do**
    $flag_{neigh} = true$
    while $(flag_{neigh} = true)$ **do**
      $s_{vnd} = sol(ng, s_c)$
      if $(f(s_{vnd}) < f(s_c))$ **then**
        $s_c = s_{vnd}$
        $flag_{vnd} = true$
        if $(f(s_c) < f(s_{bf})\ and\ IsFeasible(s_c))$ **then**
          $s_{bf} = s_c$
        end
      else
        $flag_{neigh} = false$
      end
    end
  end
end
return: $s_c,\ s_{bf}$

For the cumulative (latency) routing problems, the edges from the last customer of a given route to the associated depot do not affect the objective function. Therefore, it is unnecessary to consider them in the solution. Thus, as the total number of edges of a given solution equals the number of customers, $LB2$ considers N_c edges. The first N_v edges must correspond to the shortest edges between depots and customers, while the last $(N_c - N_v)$ edges must correspond to the shortest edges between customers. Let us also define the sets $EDC \subset E$ and $ECC \subset E$ as the sets of the edges between depots and customers and between two different customers, respectively. The vectors $CEDC$ and $CECC$ contain the travel time associated with each edge in EDC and ECC, respectively. The edges in the sets EDC and ECC are sorted according to ascending values of $CEDC$ and $CECC$, respectively.

The proposed lower bound can be computed as described in Algorithm 6. $LB2$ corresponds to the sum of the estimated latencies associated with each route. Due to the nature of the latency functions, the edges at the initial positions of the routes have the largest impact on the value of $LB2$. The procedure for the computation of this lower bound sorts the edges of the graph to include the shortest edges at the first positions of each route. Note that the routes must be sorted in descending order according to the value of NE_k. The shortest edges are included within the routes with the largest NE_k values (i.e., those having the largest impact on the value of the objective function). $LB2$ is divided into two parts: $LB2a$, associated with the edges from the depots to the customers, and $LB2b$, associated with the edges between two different customers.

Algorithm 6: LB2

Input: EDC, ECC, $CEDC$, $CECC$, NE, N_v, N_c
Output: $LB2$
1 $LB2a = 0$, $LB2b = 0$
`/* Computation of LB2a */`
2 **for** $(i = 1\ to\ N_v)$ **do**
3 $k = i$
4 $RE_k = NE_k$
5 $LB2a = LB2a + CEDC_i RE_k$
6 $RE_k = RE_k - 1$
7 **end**
`/* Computation of LB2b */`
8 $k = 1$
9 $i = 1$
10 **for** $(i = 1\ to\ (N_c - N_v))$ **do**
11 **if** $(k > N_v)$ **then**
12 k=1
13 **end**
14 $LB2b = LB2b + CECC_i RE_k$
15 $RE_k = RE_k - 1$
16 $k = k + 1$
17 **end**
18 $LB2 = LB2a + LB2b$
return: $LB2$

It is important to remark that $LB2$ has been previously proposed in [3] and then generalized

Table 3.3: Details of the computers used in each published paper.

Author	[13]	[15]	[10, 16]	**Us**
Computer	Intel Core i5-4210M @ 2.60GHz	Intel Core i7-3770 @ 3.40GHz	Intel Core i5-6300U @ 2.40GHz	Intel Core i7-8700K @ 3.70GHz
Single Thread Rating	1679	2071	1676	2750
Scaling factor	0.61	0.75	0.61	1

in [11]. The definition of $LB2$ presented in [3] and [11] is given by equation (3.11), where W_e and W'_e represent the travel time of the e^{th} shortest edge of the graph between depots and customers, and between two different customers, respectively.

$$LB2 = \sum_{e=1}^{N_v} \left\lceil \frac{N_c + N_v - e - (N_c \; mod \; N_v)}{N_v} \right\rceil W_e + \sum_{e=1}^{N_c - N_v} \left\lceil \frac{N_c - e - (N_c \; mod \; N_v)}{N_v} \right\rceil W'_e \quad (3.11)$$

Considering an instance with $N_c = N_v = 5$, it is possible to show that the equation (3.11) does not define a valid lower bound for this instance. The optimal solution is to visit each customer on a different route. Hence, each edge connecting the depots to the customers impacts the objective function value once, while the edges connecting two customers (corresponding to the second summation of (3.11)) give no contribution; nevertheless, according to the first summation of (3.11), the first 4 (i.e., $N_v - 1$) shortest edges connecting the depots to the customers impact two times on the objective function value.

3.3 Computational results

The overall algorithm (M-ILS) has been implemented in C++, and the computational experiments have been performed on an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz with 32 GB RAM, under Linux Ubuntu 18.04 operative system (single thread). The travel time matrix for all the considered instances was calculated with double precision. The ILP model (3)–(8) has been optimally solved using the ILP solver CPLEX 20.1 [33] under the default parameters configuration (one thread).

Since the previously published papers used different computers, the corresponding computing times are scaled using a "scaling factor," which approximates the original computing times reported in each published paper to the expected computing time of the processor used in our experiments. The "scaling factor" is based on the PassMark performance test (https://www.cpubenchmark.net/), which is focused on evaluating the CPU and memory performance. Higher "Single Thread Rating" values indicate that the corresponding CPU is faster (considering one thread). The value of each "scaling factor" is calculated as the ratio between the "Single Thread Rating" value of each computer and the "Single Thread Rating" value of the computer used in our experiments. The details are presented in Table 3.3.

The tables showing the computational results obtained by the proposed algorithm (M-ILS) and by the state-of-the-art methods for the solution of the MDCCVRP, the MDk-TRP, and of the LLRP are presented in subsections 3.3.3, 3.3.4 and 3.3.5, respectively.

For each instance, the following values are given:

- Instance: Name of the instance.
- N_c: Number of customers.

- N_d: Number of depots.
- N_v: Number of vehicles.
- BKS: Best known solution value considering all the algorithms. The underlined values have not been proved to be optimal.

For each algorithm and for each instance, the following values are reported:

- Best: Best solution value found. When the value of Best is equal to the corresponding BKS, it is presented in boldface.
- gap_B: Percentage gap between Best and BKS, computed as $gap_B = 100\frac{(Best-BKS)}{BKS}$.
- Avg: Average solution value computed over 30 runs for the PLS heuristic algorithm (see [13]), and computed over 30, 10, and 5 runs for the M-ILS algorithm.
- *time*: Global computing time for finding the Best value (expressed in seconds).

3.3.1 Parameter tuning

For selecting the correct parameters of the M-ILS metaheuristic, the *iterated racing for automatic algorithm configuration* IRACE software has been used. IRACE is a well-known calibration tool that has been used successfully for tuning the parameters of different metaheuristic algorithms for several combinatorial optimization problems. Details about the elitist procedures applied by the software can be found in [34].

Because of the significant differences of the instances composing the considered benchmark data sets, the parameter tuning was performed separately for each of the three problems. For each problem, the training set is a sample of 1/3 of the corresponding instances of each data set. The values analyzed for each parameter were the following: it_{max}:{50, 100, 150, 200}, it_{soft}:{5, 10, 15, 18}, it_{hard}:{20, 25, 30, 40} (both it_{soft} and it_{hard} as a percentage of it_{max}), t_0:{100, 200, 300, 400, 500}, t_f:{0.5, 1, 5, 10}, α:{0.90, 0.95, 0.98, 0.99}, and a:{0.1, 0.3, 0.5, 1, 3, 5} as a percentage of the value of the initial solution s_0. The selected configurations for each problem are presented in Table 3.4.

Table 3.4: Best configuration of parameters for each problem

Problem	it_{max}	it_{soft}	it_{hard}	t_0	t_f	α	a
MDCCVRP	200	18	20	500	10	0.98	0.1
MDk-TRP	200	18	30	500	5	0.98	-
LLRP	200	15	20	400	10	0.90	5

3.3.2 An analysis of each ingredient of the M-ILS algorithm

This section presents an analysis regarding the quality of the solution obtained and the computing time required by each ingredient of the proposed algorithm. Furthermore, the efficiency of the local search procedures and the importance of each neighborhood structure are studied. This analysis is performed to evaluate the main contribution of each ingredient of the proposed approach to the quality of the solution concerning the objective function value and the computing time. For each problem, we have considered a set of unique parameters (described in

the previous section) for analyzing the behavior of each ingredient of the proposed algorithm. It is to note that the global contribution of each ingredient remains even if the values of the parameters are changed.

Table 3.5 presents the global average results obtained by removing the different ingredients of the M-ILS algorithm and by executing 5 runs for each instance. The columns of the table correspond to the following values:

- M-ILS: The results obtained by the complete proposed algorithm.
- Initial Solution: The results obtained by the initial Constructive procedure, removing the ILS procedure (see Section 3.2.1).
- M-ILS (wLS): The results obtained by the proposed algorithm when the local search procedure (step 2 of the algorithm) is removed.
- M-ILS (wSA-VND): The results obtained by the proposed algorithm when the SA-VND procedure (step 3 of the algorithm) is removed.
- M-ILS (wLKH-3): The results obtained by the proposed algorithm when the LKH-3 procedure is removed from the first part of the perturbation $Configuration-Swap$ and from the final part of the Improvement phase.

In order to evaluate, for each problem and for each data set, the quality of the initial solution and the effect of removing step 2, step 3, and the LKH-3 procedure from the M-ILS algorithm, the following values (with the averages computed over all the corresponding instances) are considered.

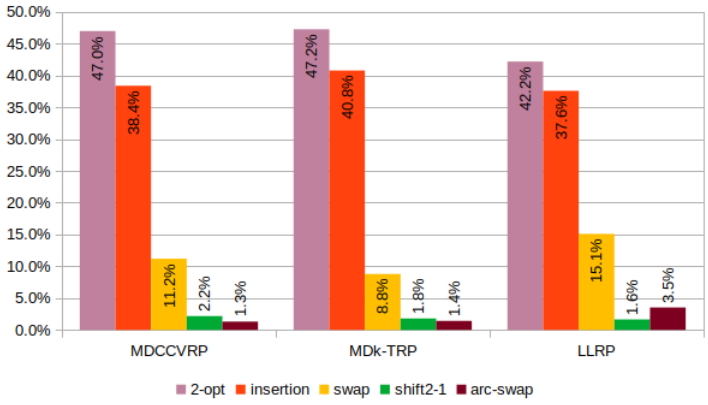
- $A-Best$: Average of the best solution values ($Best$) found by the considered algorithm.
- $A-Avg$: Average of the average solution values found for each run by the considered algorithm.
- $A-time$ (avg): Average of the average computing times required for each run by the considered algorithm.
- $A-gap_{B0}$: Average of the percentage gaps gap_{B0} between the values of $Best$ found by the considered algorithm and BKS_0, where BKS_0 represents the currently published best known solution value for the respective instance, with $gap_{B0} = 100\frac{(Best-BKS_0)}{BKS_0}$.
- $leq\ BKS_0$: The number of instances for which the best solution value $Best$ found by the considered algorithm is better than or equal to BKS_0.

The results reported in Table 3.5 show that the largest reduction of the computing time is achieved when the SA-VND procedure is removed. However, this also implies a considerable reduction of the solution quality. On the other hand, the results indicate that when the local search procedure LS (step 2) is removed, the computing time increases, generally without affecting considerably the quality of the solutions. These results suggest that the LS procedure helps to avoid extensive explorations during the execution of the SA-VND procedure. The results obtained by removing the LKH-3 procedure are worse than those obtained by the complete algorithm for all the data sets but the MDCCVRP data set lr, for which the results obtained by the two versions of the algorithm are similar. In all the cases, there is a reduction of the

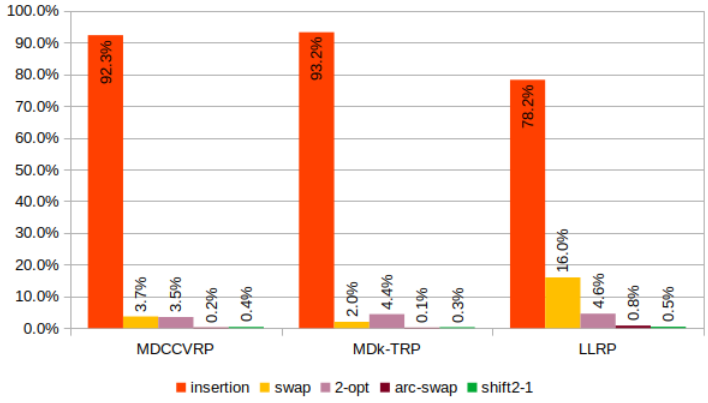
computing times; nevertheless, by considering the most complex data sets it is clear that by not considering the LKH-3 procedure the quality of the solution is negatively affected. Concerning the initial Constructive procedure, it is possible to note that it can provide reasonably good quality solutions in very short computing times; indeed, it can find solution values that are better than or equal to BKS_0 for 5 instance for the MDCCVRP, for 10 instances for the MDk-TRP, and for 13 instances for the LLRP. Globally, the best results are obtained when all the parts of the M-ILS algorithm are considered. This shows that all the ingredients of the proposed algorithm contribute to the final solution and must be considered.

Another interesting analysis regards the importance of each neighborhood in the local search (LS) and VND procedures. Figure 3.2 presents the percentage average contribution of each neighborhood for each problem. The contribution is measured in terms of the ratio of the number of times a move was applied, improving the current solution, over all the applied moves.

Regarding the LS procedure (see Figure 3.2.a), the neighborhoods $2-opt$ and *insertion* correspond to those which produce the largest impact on the effectiveness of the proposed algorithm. On the other hand, due to the descent design of the VND procedure, the neighborhoods explored at the beginning give the largest contribution to the final solution (see Figure 3.2.b), with the neighborhood *insertion* (which is the first neighborhood executed in the VND exploration), being the most applied move. As it is possible to note, all the neighborhoods contribute to the final solution in both procedures, even if some of them are not intensively applied.



(a)



(b)

Figure 3.2: Percentage average contribution of the neighborhoods. (**a**) LS. (**b**) VND.

Table 3.5: Average results obtained, for each problem and data set, when different parts of the M-ILS algorithm are removed.

Data set	# Instances	M-ILS					Initial Solution					M-ILS (wLS)					M-ILS (wSA-VND)					M-ILS (wLKH-3)				
		A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0	A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0	A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0	A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0	A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0
MDCCVRP data sets																										
p-pr	33	**9705.07**	**9762.53**	138.5	**0.27**	**13**	9913.22	10052.91	**15.7**	2.61	2	9714.97	9789.11	154.9	0.43	11	9773.40	9861.13	74.0	0.89	9	9735.35	9808.55	113.8	0.59	9
p-pr with N_v=35	24	**5536.41**	**5547.76**	176.5	**0.17**	4	5564.21	5588.54	**11.9**	0.66	2	5537.26	5549.31	184.0	0.20	**5**	5541.51	5555.14	61.7	0.26	2	5541.23	5553.11	138.4	0.24	3
lr	21	3830.01	3839.59	32.0	**0.08**	14	3869.47	3875.47	**4.1**	1.66	1	3832.21	**3837.97**	31.7	0.11	11	3832.11	3844.69	18.1	0.10	10	**3829.87**	3839.45	17.5	**0.08**	**16**
All the MDCCVRP instances	78	**6840.66**	**6871.04**	121.5	**0.19**	**31**	6947.90	7016.10	**11.4**	1.76	5	6845.70	6882.33	130.7	0.27	27	6871.70	6916.40	55.2	0.48	21	6854.91	6892.12	95.4	0.35	28
MDk-TRP data sets																										
p-pr with reduced fleet	24	**7270.43**	**7300.35**	141.8	**-0.03**	6	7342.18	7418.98	**10.8**	1.27	2	7285.38	7316.95	156.3	0.14	**8**	7290.91	7330.29	55.1	0.30	4	7278.75	7314.82	121.1	0.08	6
p-pr with N_v=35	24	**5529.99**	**5541.86**	188.2	**0.14**	5	5564.08	5584.10	**17.1**	0.69	2	5534.07	5547.23	202.3	0.20	5	5539.96	5553.43	78.0	0.27	3	5535.04	5547.80	151.7	0.20	**7**
lr	21	**3830.41**	**3837.34**	39.3	**0.09**	**14**	3868.50	3875.37	**6.8**	1.65	1	3832.76	3840.14	40.3	0.13	12	3833.68	3845.40	26.6	0.14	12	3831.33	3838.80	21.3	0.10	13
lr with reduced fleet	18	6367.18	**6394.93**	21.4	0.24	10	6526.54	6575.17	**2.7**	2.30	5	**6359.69**	6397.76	21.6	**0.16**	**12**	6405.57	6450.80	14.9	0.76	8	6363.67	6398.89	14.8	0.20	10
All the MDk-TRP instances	87	**5773.08**	**5792.02**	108.1	**0.10**	35	5844.44	5882.87	**9.9**	1.41	10	5777.35	5799.34	113.1	0.16	**37**	5790.21	5816.98	46.2	0.35	27	5776.27	5798.83	83.5	0.14	36
LLRP data sets																										
Tuzun-Burke	36	**3814.16**	**3840.57**	123.4	**-0.22**	22	3855.89	3902.73	**17.3**	0.85	7	3816.82	3840.98	131.0	-0.16	**23**	3823.51	3849.32	106.6	0.01	15	3824.81	3853.05	76.8	0.04	18
Prodhon	30	1497.73	**1503.61**	76.4	**-0.08**	22	1511.46	1527.72	**10.6**	1.09	5	**1497.29**	1503.80	77.1	**-0.08**	**23**	1498.71	1507.14	64.0	0.00	16	1501.09	1508.11	43.0	0.11	15
Barreto	10	9639.26	9815.04	39.3	**0.58**	**6**	9872.22	10432.01	**4.0**	3.54	1	9616.07	**9774.02**	32.5	0.74	5	**9592.40**	9845.78	29.1	0.87	4	9740.23	9916.49	20.9	1.07	**6**
All the LLRP instances	76	3666.24	3704.20	92.8	**-0.06**	50	3722.08	3824.34	**12.9**	1.30	13	**3664.28**	**3699.08**	96.8	-0.01	**51**	3664.89	3713.79	79.6	0.12	35	3685.89	3725.24	56.1	0.20	39

Table 3.6: Average results obtained, for each problem and data set, when different perturbations of the M-ILS algorithm are removed.

Data set	# Instances	M-ILS					P1					P2					P3				
		A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0	A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0	A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0	A-Best	A-Avg	A-time (avg)	A-gap_{B0}	#leq BKS_0
MDCCVRP data sets																					
p-pr	33	9705.07	**9762.53**	138.5	0.27	**13**	9743.62	9832.10	205.0	0.48	11	**9702.28**	9770.17	137.9	**0.25**	10	9731.71	9802.57	**123.3**	0.69	11
p-pr with N_v=35	24	5536.41	**5547.76**	176.5	**0.17**	4	**5534.92**	5548.19	213.8	0.18	**5**	5536.53	5549.04	174.0	0.18	4	5540.44	5555.85	**151.2**	0.25	3
lr	21	3830.01	3839.59	32.0	0.08	**14**	**3829.50**	**3835.28**	31.0	**0.06**	**14**	3832.80	3839.30	31.6	0.15	11	3835.07	3844.56	**20.3**	0.19	10
All the MDCCVRP instances	78	6840.66	**6871.04**	121.5	**0.19**	**31**	6856.37	6899.45	160.9	0.27	30	**6840.27**	6874.59	120.4	0.20	25	6854.53	6891.81	**104.2**	0.42	24
MDk-TRP data sets																					
p-pr with reduced fleet	24	**7270.43**	**7300.35**	141.8	**-0.03**	6	7277.45	7309.53	182.8	0.08	6	7273.97	7300.96	142.0	0.02	**8**	7285.04	7313.76	**129.7**	0.19	7
p-pr with N_v=35	24	**5529.99**	**5541.86**	188.2	**0.14**	**5**	5532.96	5545.72	226.9	0.19	5	5535.82	5546.76	193.5	0.24	4	5531.51	5545.77	**170.0**	0.17	4
lr	21	3830.41	**3837.34**	39.3	0.09	**14**	**3829.80**	3838.10	38.8	**0.06**	**14**	3834.46	3840.50	43.3	0.19	9	3834.07	3842.76	**25.6**	0.16	9
lr with reduced fleet	18	6367.18	6394.93	21.4	0.24	10	**6360.22**	**6377.35**	24.1	**0.17**	**11**	6368.72	6398.71	21.7	0.28	**11**	6380.28	6419.18	**13.8**	0.43	9
All the MDk-TRP instances	87	**5773.08**	**5792.02**	108.1	**0.10**	35	5774.25	5792.17	127.4	0.12	**36**	5776.96	5795.09	107.5	0.17	32	5781.12	5803.13	**91.7**	0.22	29
LLRP data sets																					
Tuzun-Burke	36	**3814.16**	**3840.57**	123.4	**-0.22**	22	3818.79	3843.25	142.1	-0.13	19	3815.69	3842.29	121.6	-0.20	**24**	3826.03	3865.87	**63.8**	0.02	17
Prodhon	30	**1497.73**	**1503.61**	76.4	**-0.08**	**22**	1498.19	1504.66	84.0	-0.05	19	1499.70	1505.99	76.3	0.01	21	1501.77	1510.83	**41.0**	0.24	14
Barreto	10	9639.26	9815.04	39.3	0.58	6	9605.93	**9695.45**	34.3	0.61	5	**9524.81**	9846.84	32.0	**0.56**	**7**	9649.27	9809.65	**12.9**	2.05	3
All the LLRP instances	76	3666.24	3704.20	92.8	**-0.06**	50	3664.23	**3690.15**	105.0	0.00	43	**3652.68**	3710.14	91.9	-0.02	**52**	3674.77	3718.32	**48.1**	0.37	34

The perturbations play a crucial role for the success of M-ILS. In order to evaluate the importance of the criterion based on the level of aggressiveness used in the perturbation step, three different versions of the M-ILS algorithm are compared with the original one. The considered versions are the following:
-P1: $Route - Swap$ is removed and replaced by $Route - Relocation$.
-P2: $Route - Relocation$ is removed and replaced by $Route - Swap$.
-P3: $Configuration - Swap$ is removed and replaced by $Route - Relocation$.
The comparison presented in Table 3.6 considers the same values defined at the beginning of this subsection. The results show that the original algorithm leads to the best global results regarding solution quality for the three problems. The version P1 leads to results similar to those obtained by the original version of M-ILS; nevertheless, P1 is clearly outperformed by the original M-ILS when the most complex instances are analyzed. Furthermore, in general, P1 requires larger computing times than the original version of the algorithm. The reason for the similar results obtained by the two versions is that the solution space does not change when the perturbation $Route - Swap$ is applied. With respect to P2, it is possible to note that also this version obtains results similar to those obtained by the original algorithm for the three problems in similar computing times; nevertheless, for the three problems, the original version is clearly more stable (see columns Avg and gap_{B0}). It is to note that the perturbation $Route - Relocation$ modifies the search space since the number of routes assigned to each used depot is changed; nevertheless, this search space can be potentially explored when new depot configurations are evaluated by applying $Configuration - Swap$, since the allocation of routes to depots is changed. Finally, the results show that P3 leads to the worst results in terms of solution quality among all the analyzed versions. By removing $Configuration - Swap$ it is possible to achieve a considerable reduction in the computing times; however, by avoiding an important part of the solution space associated with the used depots, worse-quality solutions are obtained.

3.3.3 The multi-depot cumulative capacitated vehicle routing problem (MDCCVRP)

There are four papers in the literature for the solution of the MDCCVRP: the POPMUSIC matheuristic [12], the PLS heuristic algorithm [13], the branch-and-cut-and-price algorithm (BCP) [15], and the two MILP formulations presented in [16]. For the MDCCVRP, the M-ILS algorithm is executed for each instance with a number of runs equal to 30, 10 and 5.

Since in [13] the heuristic algorithm PLS has been shown to be more effective than the matheuristic POPMUSIC in terms of both the solution values and the computing times, the latter algorithm is not considered in the following.
In [15], the authors presented computational results for two configurations of the BCP, one obtained by fixing a small value for the maximum number of customers that can be visited in the same route (BCP_{fix}), and the other without fixing this value (BCP_{nf}). According to the results reported in [15], BCP_{fix} dominates the non-fixed version since it can find the optimal solution for all the instances solved to proven optimality by BCP_{nf}, but within shorter computing times. In addition, BCP_{fix} can find feasible solutions for 14 instances for which BCP_{nf} runs out of memory without finding a feasible solution. Of course, the solutions found by BCP $_{fix}$ for these 14 instances are not proved to be optimal.

Table 3.7: Detailed results for the MDCCVRP p-pr data set.

Instance	N_c	N_d	N_v	BKS	PLS				BCP			M-ILS 30 runs						M-ILS 10 runs						M-ILS 5 runs					
					Best	Avg	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	gap_{PLS}	gap_{BCP}	Best	Avg	time	gap_B	gap_{PLS}	gap_{BCP}	Best	Avg	time	gap_B	gap_{PLS}	gap_{BCP}
p01	50	4	11	1055.35	**1055.35**	1095.44	26.7	0.00	**1055.35**	2.1	0.00	**1055.35**	1071.27	451.5	0.00	0.00	0.00	**1055.35**	1070.70	153.9	0.00	0.00	0.00	**1055.35**	1071.85	77.3	0.00	0.00	0.00
p02	50	4	5	2016.18	2055.40	2149.54	58.0	1.95	**2016.18**	44.9	0.00	**2016.18**	2047.63	612.3	0.00	-1.91	0.00	**2016.18**	2047.35	202.8	0.00	-1.91	0.00	2020.17	2046.77	103.9	0.20	-1.71	0.20
p03	75	5	11	1749.29	1758.70	1809.51	59.7	0.54	**1749.29**	25.8	0.00	1762.49	1799.85	1046.3	0.75	0.22	0.75	1762.49	1796.47	362.7	0.75	0.22	0.75	1773.19	1796.09	182.2	1.37	0.82	1.37
p04	100	2	15	2537.59	2618.88	2703.37	153.4	3.20	**2537.59**	748.5	0.00	2552.67	2635.03	2503.9	0.59	-2.53	0.59	2591.08	2640.48	744.9	2.11	-1.06	2.11	2591.08	2644.37	335.9	2.11	-1.06	2.11
p05	100	2	8	3749.92	3766.20	3824.49	181.9	0.43	**3749.92**	939.0	0.00	3754.39	3790.63	1928.3	0.12	-0.31	0.12	3754.39	3783.05	672.8	0.12	-0.31	0.12	3754.39	3776.43	339.0	0.12	-0.31	0.12
p06	100	3	16	2131.61	2160.93	2188.17	94.8	1.38	**2131.61**	80.3	0.00	2132.97	2164.93	2402.3	0.06	-1.29	0.06	2143.65	2164.89	802.9	0.56	-0.80	0.56	2143.65	2166.16	391.6	0.56	-0.80	0.56
p07	100	4	16	2108.89	2142.11	2181.18	109.4	1.58	**2108.89**	118.5	0.00	2140.15	2183.26	2288.7	1.48	-0.09	1.48	2141.06	2180.23	780.4	1.53	-0.05	1.53	2160.59	2177.81	391.8	2.45	0.86	2.45
p08	249	2	25	17272.00	17393.46	17862.13	644.2	0.70	–	–	–	**17272.00**	17516.46	7061.0	0.00	-0.70	–	17302.20	17495.97	2411.7	0.17	-0.52	–	17302.20	17414.82	1219.9	0.17	-0.52	–
p09	249	3	26	14918.00	15041.69	15448.98	492.8	0.83	–	–	–	**14918.00**	15021.94	6719.0	0.00	-0.82	–	**14918.00**	14997.55	2416.8	0.00	-0.82	–	**14918.00**	14976.50	1189.1	0.00	-0.82	–
p10	249	4	26	14024.58	14265.77	14619.35	522.5	1.72	**14024.58**	13887.8	0.00	14089.20	14322.70	6787.8	0.46	-1.24	0.46	14237.00	14366.55	2194.2	1.51	-0.20	1.51	14237.00	14341.92	1141.0	1.51	-0.20	1.51
p11	249	5	26	14161.20	14381.43	14552.90	518.8	1.56	–	–	–	**14161.20**	14404.70	6523.5	0.00	-1.53	–	14269.20	14435.03	2123.2	0.76	-0.78	–	14289.70	14409.52	1060.0	0.91	-0.64	–
p12	80	2	8	5494.36	**5494.36**	5536.40	71.4	0.00	**5494.36**	63.1	0.00	**5494.36**	5495.85	1129.8	0.00	0.00	0.00	**5494.36**	5497.45	377.5	0.00	0.00	0.00	**5494.36**	**5494.36**	196.6	0.00	0.00	0.00
p13	80	2	9	4914.66	**4914.66**	4926.54	74.1	0.00	**4914.66**	48.0	0.00	**4914.66**	4914.83	970.1	0.00	0.00	0.00	**4914.66**	4914.83	314.8	0.00	0.00	0.00	**4914.66**	**4914.66**	145.6	0.00	0.00	0.00
p14	80	2	10	4491.64	4510.12	4512.28	71.0	0.41	**4491.64**	44.6	0.00	**4491.64**	4492.13	894.3	0.00	-0.41	0.00	**4491.64**	4492.59	299.3	0.00	-0.41	0.00	**4491.64**	**4491.64**	144.4	0.00	-0.41	0.00
p15	160	4	16	10590.41	10662.27	10747.26	230.4	0.68	**10590.41**	534.0	0.00	10629.80	10676.81	2653.0	0.37	-0.30	0.37	10646.80	10683.73	885.2	0.53	-0.15	0.53	10668.00	10683.38	435.3	0.73	0.05	0.73
p16	160	4	17	10008.28	10086.50	10122.13	196.0	0.78	**10008.28**	624.8	0.00	10016.10	10070.56	2548.3	0.08	-0.70	0.08	10055.40	10074.89	904.4	0.47	-0.31	0.47	10055.40	10073.96	456.8	0.47	-0.31	0.47
p17	160	4	18	9493.84	9538.69	9573.86	194.0	0.47	**9493.84**	313.5	0.00	9495.91	9518.63	2465.7	0.02	-0.45	0.02	9512.48	9522.32	841.4	0.20	-0.27	0.20	9512.48	9523.62	420.8	0.20	-0.27	0.20
p18	240	6	24	15720.73	15912.27	16072.75	376.2	1.22	**15720.73**	9405.8	0.00	15847.80	15948.26	4978.6	0.81	-0.41	0.81	15847.80	15940.73	1670.6	0.81	-0.41	0.81	15850.30	15920.10	846.7	0.82	-0.39	0.82
p19	240	6	25	15105.26	15255.02	15370.98	346.8	0.99	**15105.26**	1598.3	0.00	15224.80	15301.79	4749.4	0.79	-0.20	0.79	15224.80	15293.53	1522.0	0.79	-0.20	0.79	15268.80	15318.90	766.6	1.08	0.09	1.08
p20	240	6	26	14592.52	14709.23	14786.87	333.8	0.80	**14592.52**	1582.5	0.00	14635.70	14708.73	4736.2	0.30	-0.50	0.30	14636.90	14700.60	1599.3	0.30	-0.49	0.30	14649.00	14699.60	802.0	0.39	-0.41	0.39
p21	360	9	34	25500.80	25770.35	26102.29	725.0	1.06	–	–	–	**25500.80**	25892.53	10982.1	0.00	-1.05	–	25632.20	25943.27	3724.9	0.52	-0.54	–	25632.20	25878.44	1795.3	0.52	-0.54	–
p22	360	9	35	24330.70	24451.01	24816.85	674.5	0.49	–	–	–	**24330.70**	24608.60	10342.0	0.00	-0.49	–	**24330.70**	24606.22	3433.2	0.00	-0.49	–	24448.70	24531.56	1676.4	0.48	-0.01	–
p23	360	9	36	23622.80	23656.13	23925.16	598.0	0.14	–	–	–	**23622.80**	23784.41	9638.3	0.00	-0.14	–	**23622.80**	23750.51	3255.1	0.00	-0.14	–	23636.50	23754.00	1609.1	0.06	-0.08	–
pr01	48	4	4	3748.11	**3748.11**	3773.27	61.7	0.00	**3748.11**	38.8	0.00	3768.69	3768.69	444.5	0.55	0.55	0.55	3768.69	3768.69	149.8	0.55	0.55	0.55	3768.69	3768.69	73.0	0.55	0.55	0.55
pr02	96	4	8	4834.46	4973.36	5000.74	169.8	2.87	**4834.46**	318.0	0.00	**4834.46**	4854.38	983.3	0.00	-2.79	0.00	**4834.46**	4859.87	325.2	0.00	-2.79	0.00	4846.47	4864.04	168.2	0.25	-2.55	0.25
pr03	144	4	11	8353.05	8357.54	8470.56	339.8	0.05	**8353.05**	1401.8	0.00	8357.54	8424.14	1955.7	0.05	0.00	0.05	8383.72	8426.87	657.1	0.37	0.31	0.37	8401.06	8422.54	322.9	0.57	0.52	0.57
pr04	192	4	14	9071.44	9274.00	9585.47	728.2	2.23	**9071.44**	41308.5	0.00	9156.38	9324.65	4431.6	0.94	-1.27	0.94	9161.21	9308.96	1519.6	0.99	-1.22	0.99	9161.21	9256.52	717.8	0.99	-1.22	0.99
pr05	240	4	19	9805.38	10075.01	10283.54	967.3	2.75	–	–	–	**9805.38**	10025.37	7502.8	0.00	-2.68	–	9861.20	10078.00	2434.5	0.57	-2.12	–	9861.20	10051.80	1221.3	0.57	-2.12	–
pr06	288	4	23	10873.00	11071.00	11234.57	1043.1	1.82	–	–	–	**10873.00**	10996.31	8934.9	0.00	-1.79	–	**10873.00**	10971.04	3052.6	0.00	-1.79	–	**10873.00**	10993.04	1659.3	0.00	-1.79	–
pr07	72	6	6	4760.65	4877.86	4906.62	120.4	2.46	**4760.65**	207.0	0.00	**4760.65**	4787.68	731.8	0.00	-2.40	0.00	**4760.65**	4790.07	240.3	0.00	-2.40	0.00	4770.53	4795.37	119.3	0.21	-2.20	0.21
pr08	144	6	12	6997.11	7141.88	7265.17	343.1	2.07	**6997.11**	1395.0	0.00	7049.50	7131.65	2961.0	0.75	-1.29	0.75	7049.50	7116.86	971.2	0.75	-1.29	0.75	7049.50	7102.81	488.6	0.75	-1.29	0.75
pr09	216	6	17	9027.82	9219.95	9350.34	675.8	2.13	**9027.82**	21477.0	0.00	9147.07	9327.58	5598.4	1.32	-0.79	1.32	9191.35	9305.49	1837.2	1.81	-0.31	1.81	9191.35	9295.98	922.4	1.81	-0.31	1.81
pr10	288	6	24	11335.10	11693.45	11811.29	989.1	3.16	–	–	–	**11335.10**	11493.69	9038.6	0.00	-3.06	–	**11335.10**	11531.23	3040.6	0.00	-3.06	–	11476.80	11506.30	1438.4	1.25	-1.85	–
Global avg				9648.39	9758.57	9897.27	**369.4**	1.23	–	–	–	**9671.13**	**9772.90**	4151.4	**0.29**	**-0.92**	–	9691.52	9774.43	1391.6	0.49	-0.72	–	9705.07	9762.53	692.7	0.64	-0.57	–
Global avg BCP				6940.74	7020.80	7107.18	**230.8**	1.17	**6940.74**	4008.6	**0.00**	6972.02	**7031.74**	2510.5	0.39	-0.76	0.39	6986.48	7031.13	834.6	0.59	-0.56	0.59	6992.87	7026.98	416.2	0.71	-0.44	0.71

Table 3.8: Detailed results for the MDCCVRP p-pr data set with N_v=35.

Instance	N_c	N_d	BKS	PLS			BCP			M-ILS 30 runs						M-ILS 10 runs						M-ILS 5 runs					
			Best	Avg	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	gap_{PLS}	gap_{PLS}	Best	Avg	time	gap_B	gap_{PLS}	gap_{PLS}	Best	Avg	time	gap_B	gap_{PLS}	gap_{PLS}
p01	50	4	712.50	713.18	717.44	41.0	0.10	**712.50**	1.9	0.00	**712.50**	714.18	1930.2	0.00	-0.10	**712.50**	713.73	640.6	0.00	-0.10	712.74	714.09	319.3	0.03	-0.06	-0.06
p02	50	4	712.50	713.59	716.78	40.6	0.15	**712.50**	3.8	0.00	**712.50**	713.65	1875.5	0.00	-0.15	712.74	713.75	578.5	0.03	-0.12	712.74	713.73	251.5	0.03	+0.12	+0.12
p03	75	5	950.25	952.17	959.03	61.7	0.20	**950.25**	11.6	0.00	**950.25**	953.90	1859.2	0.00	-0.20	**950.25**	953.76	591.9	0.00	-0.20	**950.25**	953.49	292.3	0.00	+0.00	+0.00
p04	100	2	1955.31	1959.82	1959.14	95.5	0.03	**1955.31**	38.0	0.00	**1955.31**	1955.97	3646.5	0.00	-0.03	1955.48	1955.88	1238.7	0.01	-0.02	1955.48	1956.07	623.0	0.01	+0.02	+0.02
p05	100	2	1982.33	1985.03	1988.46	93.7	0.14	**1982.33**	58.5	0.00	1982.35	1984.17	3177.4	0.00	-0.14	1982.35	1983.94	1114.8	0.00	-0.14	1982.35	1983.57	600.1	0.00	+0.44	+0.44
p06	100	3	1552.13	1553.88	1563.22	83.6	0.11	**1552.13**	28.0	0.00	**1552.13**	1553.88	3164.8	0.00	-0.11	**1552.13**	1554.15	1122.5	0.00	-0.11	1553.64	1555.39	550.8	0.10	+0.02	+0.02
p07	100	4	1520.46	1522.68	1528.43	97.0	0.15	**1520.46**	26.6	0.00	1520.97	1524.03	3161.4	0.03	-0.11	1520.97	1522.96	1103.9	0.03	-0.11	1520.97	1521.73	523.4	0.03	+0.01	+0.01
p08	249	2	15372.60	15410.92	15458.51	374.4	0.25	**15372.60**	937.5	0.00	**15372.60**	15400.51	7956.0	0.00	-0.25	**15372.60**	15394.63	2640.7	0.00	-0.25	15372.80	15397.68	1422.1	0.00	+0.25	+0.25
p09	249	3	13070.74	13136.24	13335.06	340.4	0.50	**13070.74**	812.3	0.00	13071.60	13105.79	7521.8	0.01	-0.49	13078.60	13112.53	2446.3	0.06	-0.44	13078.60	13108.02	1172.1	0.06	+0.44	+0.44
p10	249	4	12052.56	12096.90	12432.72	341.1	0.37	**12052.56**	735.0	0.00	12070.50	12155.06	7577.5	0.15	-0.22	12071.50	12186.16	2581.9	0.16	-0.21	12180.00	12211.26	1315.3	1.06	+1.69	+1.69
p11	249	5	11955.58	12033.48	12228.64	317.3	0.65	**11955.58**	935.3	0.00	11995.90	12051.88	7297.0	0.34	-0.31	12007.20	12052.14	2456.6	0.43	-0.22	12007.20	12057.88	1247.2	0.43	-0.22	-0.22
p12	80	2	2897.06	**2897.06**	2897.06	37.0	0.00	**2897.06**	17.3	0.00	**2897.06**	**2897.06**	1426.2	0.00	0.00	**2897.06**	**2897.06**	471.6	0.00	0.00	**2897.06**	**2897.06**	238.3	0.00	0.00	0.00
p15	160	4	5794.11	**5794.11**	5794.11	95.2	0.00	**5794.11**	171.8	0.00	**5794.11**	**5794.11**	2987.3	0.00	0.00	**5794.11**	**5794.11**	990.4	0.00	0.00	**5794.11**	**5794.11**	484.8	0.00	0.00	0.00
p18	240	6	11433.91	11469.49	11546.91	225.6	0.31	**11433.91**	895.5	0.00	11453.50	11476.90	5023.9	0.17	-0.14	11454.00	11476.24	1674.0	0.18	-0.14	11454.00	11471.48	829.0	0.18	+0.14	+0.14
pr01	48	4	1261.53	1261.81	1264.74	37.5	0.02	**1261.53**	3.1	0.00	1262.43	1266.58	1315.9	0.07	0.05	1263.67	1266.62	445.9	0.17	0.15	1263.67	1267.06	221.1	0.17	0.15	0.15
pr02	96	4	2572.84	**2572.84**	2580.94	88.2	0.00	**2572.84**	37.7	0.00	**2572.84**	2574.88	2739.4	0.00	0.00	**2572.84**	2573.91	985.4	0.00	0.00	**2572.84**	2574.64	443.8	0.00	0.00	0.00
pr03	144	4	4462.50	4466.10	4511.37	137.6	0.08	**4462.50**	122.3	0.00	4464.91	4475.65	4596.5	0.05	-0.03	4464.91	4473.63	1539.2	0.05	-0.03	4464.91	4474.82	765.4	0.05	+0.03	+0.03
pr04	192	4	5804.15	5813.87	5863.56	233.7	0.17	**5804.15**	405.8	0.00	5813.33	5825.20	7706.8	0.16	-0.01	5813.87	5825.33	2618.1	0.17	0.00	5813.87	5826.20	1309.2	0.17	+0.00	+0.00
pr05	240	4	7120.22	7157.06	7225.66	372.0	0.52	**7120.22**	946.5	0.00	7122.06	7146.94	11480.7	0.03	-0.49	7123.63	7149.97	3848.5	0.05	-0.47	7123.63	7149.70	1954.0	0.05	+0.47	+0.47
pr06	288	4	8603.85	8685.68	8874.53	499.4	0.95	**8603.85**	1732.5	0.00	8607.46	8657.63	13211.5	0.04	-0.90	8616.21	8666.15	4299.8	0.14	-0.80	8659.23	8674.06	2166.5	0.64	-0.30	-0.30
pr07	72	6	1723.63	1727.25	1736.14	65.1	0.21	**1723.63**	13.1	0.00	1725.55	1727.74	2162.5	0.11	-0.10	1725.55	1728.32	746.6	0.11	-0.10	1725.61	1729.78	402.3	0.11	+0.09	+0.09
pr08	144	6	4004.11	4023.21	4046.68	150.1	0.48	**4004.11**	118.5	0.00	**4004.11**	4015.78	4411.3	0.00	-0.47	**4004.11**	4015.01	1487.3	0.00	-0.47	4008.65	4015.09	760.2	0.11	+0.36	+0.36
pr09	216	6	5889.02	5937.19	6043.29	268.6	0.82	**5889.02**	512.3	0.00	5899.64	5932.25	8134.3	0.18	-0.63	5904.08	5923.84	2741.1	0.26	-0.56	5904.08	5920.67	1352.2	0.26	+0.56	+0.56
pr10	288	6	9113.49	9166.57	9336.73	523.4	0.58	**9113.49**	1555.5	0.00	9135.86	9177.17	11912.7	0.25	-0.34	9162.80	9183.27	4035.7	0.54	-0.04	9165.34	9178.64	1932.6	0.57	+0.01	+0.01
Global avg			5521.56	5543.59	5608.71	**192.5**	0.28	**5521.56**	421.7	**0.00**	5527.06	5545.04	5261.5	0.07	-0.22	5529.71	5546.55	1766.7	0.10	-0.18	5536.41	5547.76	882.4	0.17	+0.11	+0.11

Table 3.9: Detailed results for the MDCCVRP lr data set.

Instance	N_c	N_d	N_v	BKS	PLS				BCP			Formulations			M-ILS 30 runs					M-ILS 10 runs					M-ILS 5 runs				
					Best	Avg	time	gap_B	Best	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	gap_{PLS}	Best	Avg	time	gap_B	gap_{PLS}	Best	Avg	time	gap_B	gap_{PLS}
lr1	10	4	5	545.69	**545.69**	554.97	2.7	0.00	**545.69**	0.0	0.00	**545.69**	0.0	0.00	**545.69**	546.13	13.4	0.00	0.00	**545.69**	**545.69**	3.4	0.00	0.00	**545.69**	**545.69**	2.0	0.00	0.00
lr2	10	4	5	832.69	**832.69**	847.80	2.6	0.00	**832.69**	0.0	0.00	**832.69**	0.0	0.00	**832.69**	**832.69**	38.2	0.00	0.00	**832.69**	**832.69**	12.8	0.00	0.00	**832.69**	**832.69**	6.4	0.00	0.00
lr3	10	4	5	832.78	**832.78**	841.47	2.6	0.00	**832.78**	0.0	0.00	**832.78**	0.0	0.00	**832.78**	**832.78**	21.8	0.00	0.00	**832.78**	**832.78**	7.5	0.00	0.00	**832.78**	**832.78**	3.3	0.00	0.00
lr4	25	4	10	2082.28	**2082.28**	2099.58	11.5	0.00	**2082.28**	0.8	0.00	**2082.28**	0.3	0.00	**2082.28**	2082.57	377.7	0.00	0.00	**2082.28**	2083.14	124.0	0.00	0.00	**2082.28**	2083.43	60.7	0.00	0.00
lr5	25	4	10	1827.41	**1827.41**	1870.90	11.9	0.00	**1827.41**	0.8	0.00	**1827.41**	0.3	0.00	**1827.41**	1837.30	335.0	0.00	0.00	**1827.41**	1836.14	111.3	0.00	0.00	**1827.41**	1834.39	57.9	0.00	0.00
lr6	25	4	10	1786.95	**1786.95**	1808.86	11.9	0.00	**1786.95**	0.7	0.00	**1786.95**	0.3	0.00	**1786.95**	**1786.95**	229.1	0.00	0.00	**1786.95**	**1786.95**	79.2	0.00	0.00	**1786.95**	**1786.95**	41.3	0.00	0.00
lr7	50	4	20	5424.57	**5424.57**	5440.37	49.8	0.00	**5424.57**	14.2	0.00	**5424.57**	3.0	0.00	**5424.57**	**5424.57**	1382.5	0.00	0.00	**5424.57**	**5424.57**	450.6	0.00	0.00	**5424.57**	**5424.57**	242.7	0.00	0.00
lr8	50	4	20	3737.38	**3737.38**	3759.67	45.6	0.00	**3737.38**	12.5	0.00	**3737.38**	4.1	0.00	**3737.38**	3743.96	713.0	0.00	0.00	**3737.38**	3741.14	196.9	0.00	0.00	**3737.38**	3742.68	96.6	0.00	0.00
lr9	50	4	20	3802.88	**3802.88**	3811.65	49.0	0.00	**3802.88**	11.3	0.00	**3802.88**	3.7	0.00	**3802.88**	3808.52	838.8	0.00	0.00	**3802.88**	3808.80	288.5	0.00	0.00	**3802.88**	3811.90	129.7	0.00	0.00
lr10-25V	50	6	25	2866.73	2868.39	2883.50	111.6	0.06	**2866.73**	8.9	0.00	**2866.73**	3.4	0.00	2867.28	2874.88	1009.5	0.02	-0.04	2869.43	2874.55	311.0	0.09	0.04	2870.00	2876.04	147.2	0.11	0.06
lr11-25V	50	6	25	2978.78	2987.75	3008.88	116.6	0.30	**2978.78**	10.1	0.00	**2978.78**	4.0	0.00	2979.46	2992.10	803.0	0.02	-0.28	2979.46	2993.20	281.9	0.02	-0.28	2985.06	2995.79	118.3	0.21	-0.09
lr12-25V	50	6	25	3090.38	3095.52	3112.60	112.5	0.17	**3090.38**	9.8	0.00	**3090.38**	3.8	0.00	**3090.38**	3095.64	896.8	0.00	-0.17	**3090.38**	3095.38	322.1	0.00	-0.17	3093.25	3096.55	161.4	0.09	-0.07
lr10-20V	50	6	20	2969.83	–	–	–	–	**2969.83**	9.6	0.00	–	–	–	**2969.83**	2992.04	1000.9	0.00	–	**2969.83**	2993.71	337.6	0.00	–	**2969.83**	2993.19	167.8	0.00	–
lr11-20V	50	6	20	3095.22	–	–	–	–	**3095.22**	14.9	0.00	–	–	–	**3095.22**	3120.25	545.4	0.00	–	3113.23	3126.52	170.8	0.58	–	3113.81	3129.93	88.4	0.60	–
lr12-20V	50	6	20	3171.75	–	–	–	–	**3171.75**	11.0	0.00	–	–	–	3174.98	3192.62	814.4	0.10	–	3184.43	3194.44	271.1	0.40	–	3188.09	3193.38	132.3	0.52	–
lr13	100	4	25	8288.43	8293.42	8331.27	111.8	0.06	**8288.43**	155.3	0.00	**8288.43**	78.8	0.00	**8288.43**	8324.16	1892.3	0.00	-0.06	**8288.43**	8331.18	629.4	0.00	-0.06	**8288.43**	8332.12	343.5	0.00	-0.06
lr14	100	4	25	7257.31	7273.03	7353.68	100.7	0.22	**7257.31**	158.3	0.00	**7257.31**	71.3	0.00	**7257.31**	7272.96	1925.0	0.00	-0.22	**7257.31**	7272.84	708.8	0.00	-0.22	**7257.31**	7275.67	346.4	0.00	-0.22
lr15	100	4	25	8626.13	8626.13	8644.64	133.0	0.00	**8626.13**	204.0	0.00	**8626.13**	49.5	0.00	**8626.13**	8643.77	2584.7	0.00	0.00	**8626.13**	8643.36	873.4	0.00	0.00	**8626.13**	8640.30	447.1	0.00	0.00
lr16	100	6	25	5265.30	5306.50	5483.36	128.8	0.78	**5265.30**	115.5	0.00	**5265.30**	66.1	0.00	**5265.30**	5281.53	1407.9	0.00	-0.78	**5265.30**	5278.95	475.4	0.00	-0.78	5268.68	5289.54	241.4	0.06	-0.71
lr17	100	6	25	6107.32	6141.07	6265.57	138.2	0.55	**6107.32**	132.0	0.00	**6107.32**	57.8	0.00	**6107.32**	6112.13	1638.5	0.00	-0.55	**6107.32**	**6107.32**	562.7	0.00	-0.55	**6107.32**	**6107.32**	274.7	0.00	-0.55
lr18	100	6	25	5788.73	5804.19	5905.71	143.3	0.27	**5788.73**	140.3	0.00	**5788.73**	66.3	0.00	**5788.73**	5811.51	1562.1	0.00	-0.27	5789.77	5811.17	499.3	0.02	-0.25	5789.77	5806.50	252.0	0.02	-0.25
Global avg				3827.55	–	–	–	–	**3827.55**	**48.1**	**0.00**	–	–	–	3827.76	3838.53	953.8	0.01	–	3829.22	3838.79	319.9	0.05	–	3830.01	3839.59	160.1	0.08	–
Global avg PLS/Formulations				3952.32	3959.37	4001.36	71.3	0.13	**3952.32**	54.1	**0.00**	**3952.32**	**22.9**	**0.00**	3952.39	3961.34	981.6	0.00	-0.13	3952.56	3961.10	329.9	0.01	-0.13	3953.25	3961.94	165.1	0.03	-0.11

In order to present a fair comparison, the global computing times reported in Tables 3.7, 3.8 and 3.9 for each instance correspond to: *i*) for PLS to the average computing time reported in [13] multiplied by 30 (number of runs) and by the scaling factor (0.61); *ii*) for BCP to the scaled computing times (considering a scaling factor equal to 0.75) reported for BCP$_{fix}$ in [15], and *iii*) for the two formulations to the scaled computing time (considering a scaling factor equal to 0.61) of the fastest between the two MILP models, as the computing time of each separate model was not reported in [16]. The computing times reported for M-ILS correspond to the average computing times multiplied by the respective number of runs. Furthermore, for each number of runs of M-ILS the following values are reported:

- gap_{PLS}: Percentage gap between the Best solution value found by M-ILS ($Best$) and the Best solution value found by PLS (Best$_{PLS}$), computed as $gap_{PLS} = 100\frac{(Best - Best_{PLS})}{Best_{PLS}}$.
- gap_{BCP}: Percentage gap between the Best solution value found by M-ILS and the Best solution value found by BCP (Best$_{BCP}$), computed as $gap_{BCP} = 100\frac{(Best - Best_{BCP})}{Best_{BCP}}$.

The p-pr data set

The computational results corresponding to the 33 instances of the data set p-pr are reported in Table 3.7. This data set contains the most challenging MDCCVRP instances due to the large number N_c of customers and the small number N_v of vehicles (generally, the smaller N_v, the more difficult is the instance [12, 15]). According to the results reported in Table 3.7, only the metaheuristic algorithms, i.e., PLS and M-ILS, can find a feasible solution for all the instances in this data set. BCP$_{fix}$ can find the proven optimal solution for 18 instances and the best-known feasible solution for 6 instances (no feasible solution is found for the remaining 9 instances).

For the 9 instances for which BCP$_{fix}$ runs out of memory without a feasible solution, M-ILS provides the new best-known solution value (outperforming the solution value provided by PLS), independently of the number of runs. The corresponding values of LB and gap_{LB} (where gap_{LB} is the percentage gap between BKS and LB, computed as $gap_{LB} = 100\frac{(BKS - LB)}{LB}$) for these instances are the following: p08: 14444.3 (19.58%), p09: 11742.3 (27.045%), p11: 9883.96 (43.27%), p21: 21397.1 (19.18%), p22: 20822.2 (16.85%), p23: 20247.4 (16.67%), pr05: 6436.69 (52.34%), pr06: 7434.05 (46.26%), and pr10: 7645.9 (48.25%). Furthermore, considering the 24 instances for which BCP$_{fix}$ provides the best-known solution value, M-ILS (executed for 30 and 10 runs) finds the optimal solution value for 7 instances, and the average percentage gap between the best solution value provided by M-ILS and BKS is equal to 0.29% when M-ILS is executed for 30 runs. The global average computing time required by BCP$_{fix}$ for solving these 24 instances is 1.6 times larger than that required by M-ILS (executed for 30 runs). No computing time has been reported in [15] for the 9 instances for which BCP$_{fix}$ runs out of memory without finding a feasible solution. Therefore, it is impossible to compute the global average computing time (considering all the 33 instances) associated with this algorithm. The corresponding values of LB and gap_{LB} for the 6 instances for which BCP provides the best known feasible solution value are the following: p10: 10478.5 (33.84%), p18: 13535.5 (16.14%), p19: 13097.9 (15.33%), p20: 12672.8 (15.15%), pr04: 5559.22 (63.18%), and pr09: 5586.49 (61.60%). On the other hand, the MILP formulations can solve optimally only two small-size instances (with up to 50 customers) and provide, within the time limit, feasible solutions for 7 additional instances with up to 100 customers, with an average percentage value of gap_B equal to 2.48%. For the 9 instances for which the MILP formulations are able to provide a solution, the average percentage value of gap_F (where gap_F is the percentage gap between the Best solution value found by M-ILS and the

Best solution value found by the formulations ($Best_F$), computed as $gap_F = 100\frac{(Best - Best_F)}{Best_F}$) is equal to -2.18%, -2.13%, and -1.99% when M-ILS is executed for 30, 10 and 5 runs, respectively. Since the MILP formulations are dominated by BCP, the results associated with them are not reported in Table 3.7.

By comparing the best results provided by the heuristic algorithms PLS and M-ILS (both executed for 30 runs) on the 33 instances of this data set, it is possible to see that M-ILS provides better solutions than PLS for 27 instances, the same solution value for 4 instances and worse solution values only for 2 instances. The final average percentage value of gap_{PLS} equals -0.92%. However, it is to note that, although M-ILS provides better quality solutions than PLS, the computing times required by PLS are clearly smaller than those required by M-ILS.
For all the instances but one, the Avg. solution value provided by M-ILS is better than the Avg. solution value provided by PLS. Furthermore, for 12 instances, the Avg. solution value provided by M-ILS is better than the best solution value reported for PLS. The global average percentage gap between the average solution value provided by M-ILS and the best solution value provided by PLS is equal to 0.16%. This indicates that M-ILS is more stable than PLS, hence it needs fewer runs to provide good-quality solutions. Indeed, reducing the number of runs to 10 and 5, the number of instances for which M-ILS provides better solution values than those found by PLS equals 27 and 24, respectively. For 4 (resp. 3) instances, both heuristic algorithms found the same solution value when M-ILS is executed for 30 (resp. 10 and 5) runs. Globally, the gap_{PLS} value is equal to -0.72% and to -0.57% by considering 10 and 5 runs, respectively; this means that independently of the number of runs, M-ILS overcomes PLS in terms of solution quality. For 11 and 13 instances, the average solution value provided by M-ILS is better than the best value found by PLS when, respectively, 10 and 5 runs are considered for M-ILS. In addition, when M-ILS is executed for 5 runs, the average solution value provided by M-ILS is equal to the best solution value found by PLS for two instances. Thus, the reduction in the number of runs does not significantly affect the quality of the solutions provided by the proposed algorithm. In contrast, the global computing time of M-ILS is drastically reduced to very competitive ones with respect to those of PLS.

The p-pr data set with N_v=35

The p-pr data set with N_v=35 is composed of 24 instances, and the corresponding computational results are reported in Table 3.8. BCP$_{fix}$ can obtain a proven optimal solution for 16 instances and the best-known feasible solution for the remaining 8 instances. The MILP formulations cannot find a feasible solution for 9 large-size instances, and the largest-size instance that can be solved optimally considers 192 customers. The columns gap$_{BCP}$ and gap$_F$ are not reported in Table 3.8, since for all the instances the solution values provided by BCP$_{fix}$ are equal to BKS (so the value of gap_{BCP} is always equal to gap_B), and all the instances solved by the MILP formulations were solved to proven optimally (so the value of gap_F is equal to gap_B for these instances). Furthermore, by considering only the 15 instances for which the MILP formulations can provide the optimal solution, the global computing time required by the formulations is larger than that required by BCP$_{fix}$ (134.4 s > 70.5 s). Since the MILP formulations are dominated by BCP, the results associated with them are not reported in Table 3.8.

The global average value of gap_B obtained by M-ILS (with 30 runs) is equal to 0.07%. M-ILS finds the optimal solution value for 10 instances and near-optimal solution values (the largest gap_B value is equal to 0.34%) for all the remaining ones. Compared to PLS, M-ILS (with 30 runs) provides a better solution value for 20 instances, the same solution value for 3 instances, and a worse solution value for one instance. The global value of gap_{PLS} for M-ILS (with 30 runs)

is equal to -0.22%, while the global average percentage gap between the Avg value provided by M-ILS and the best solution value obtained by PLS is equal to 0.04%. Regarding the global computing time, M-ILS presents large computing times when 30 runs are considered. However, when the number of runs is reduced to 10 or to 5, the quality of the solutions is not affected significantly, and the computing times decrease considerably. The global values of gap_B are equal to 0.10% and 0.17% when the number of runs is reduced to 10 and 5, respectively. M-ILS provides better solution values than those obtained by PLS for 19 and 18 instances when 10 and 5 runs are considered, respectively.

The average solution value provided by M-ILS is better than (for 21 instances) or equal to (for two instances) the average solution value provided by PLS, independently of the number of runs. Furthermore, for 8 and 7 instances, the average solution value provided by M-ILS considering 30 (or 5) and 10 runs, respectively, is better than the best value provided by PLS. For two instances, these values are equal, independently of the number of runs. The global average percentage gap between the value Avg provided by M-ILS and the best solution value obtained by PLS equals 0.05% and 0.07% when 10 and 5 runs are considered, respectively.

For two instances, the average solution value provided by M-ILS is equal to the optimal solution value independently of the number of runs. In addition, the global percentage gap between the average solution value provided by M-ILS and BKS is equal to 0.33% and 0.35%, when M-ILS is executed with 30 (or 10) and 5 runs, respectively. A single run may provide solution values close to BKS, making M-ILS competitive with respect to the global computing time.

The lr data set

The lr data set is composed of 21 instances and corresponds to the simplest data set due to the small number of customers and the relatively large size of the fleet. The corresponding computational results are reported in Table 3.9. This data set was proposed in [12] as a small data set which allows the exact methods to find optimal solutions. Indeed, all the instances of this data set were solved to proven optimality in [15]. Besides, all the instances but three were also solved optimally in [16]. For the same reasons given in the previous Section (3.3.3), the columns gap_{BCP} and gap_F are not reported in Table 3.9.

By comparing the best results provided by the metaheuristic algorithms, it is possible to note that M-ILS (with 30 runs) provides solutions better than those found by PLS for 8 instances, and the same solution value for 10 instances. The value of gap_{PLS} is equal to -0.13%. Regarding the computing times, PLS is globally much faster than M-ILS. For all the instances, the Avg. solution value provided by M-ILS is better than the Avg. solution value provided by PLS. Furthermore, for 3 instances, the Avg. solution value provided by M-ILS is better than the best solution value reported for PLS. For 4 instances, these values are the same. The global average percentage gap between the average solution value provided by M-ILS and the best solution value provided by PLS is equal to 0.06%.

Also for this data set, the results indicate that M-ILS is more stable than PLS; hence, it needs fewer runs to provide good-quality solutions. Indeed, the number of instances for which M-ILS provides better solution values than those obtained by PLS is equal to 7 when the number of runs is reduced to 10 or to 5. Furthermore, for 10 instances, both algorithms found the same solution value independently of the number of runs executed by M-ILS.

Globally, the gap_{PLS} value is equal to -0.13% and -0.11% by considering 10 and 5 runs, respectively, which means that independently of the number of runs, M-ILS overcomes PLS in terms of solution quality. Furthermore, when the number of runs is equal to 10 (resp. 5), the

Table 3.10: Average results obtained by each metaheuristic considering time limits and target values for each MDCCVRP data set.

Data set	# Instances	PLS		M-ILS			
		TV	**TL**	**Best$_{TL}$**	*gap$_{TL}$*	**#TV**	**time$_{TL}$**
p-pr	33	9758.57	369.4	**9718.85**	**-0.32**	**21**	**205.5**
p-pr with N_v=35	24	**5543.59**	192.5	5545.27	0.02	**15**	**155.5**
lr	18	**3959.37**	71.3	3960.42	0.00	**12**	**39.1**
All the instances	75	7017.97	241.3	**7001.28**	**-0.13**	**48**	**149.6**

number of instances for which the average solution value provided by M-ILS is better than the best solution value found by PLS is equal to 4 (resp. 2), and for 5 instances these values are equal when 10 or 5 runs are considered.

Comparing the results obtained by M-ILS versus the optimal solution values, it is possible to see that M-ILS can find the optimal solution value for 18, 16, and 14 instances, with a global value of gap_B equal to 0.01%, 0.05%, and 0.08%, when respectively, 30, 10, and 5 runs are executed. Furthermore, for 4 and 6 instances the average solution value obtained by M-ILS is equal to the optimal solution value by considering 30, and 10 (or 5) runs, respectively. In addition, the global percentage gap between the average solution value provided by M-ILS and the optimal solution value is equal to 0.27%, 0.28%, and 0.30% when the algorithm is executed with 30, 10, and 5 runs, respectively. A single run may provide near-optimal solution values, making M-ILS competitive in terms of computing time.

Overall results on the MDCCVRP

Analyzing the results, it is possible to state that the proposed algorithm M-ILS overcomes PLS in terms of solution quality for all the studied data sets by executing 30, 10, or 5 runs. The global average percentage gap_{PLS} value is equal to -0.51%, -0.41%, and -0.31% when M-ILS is executed with 30, 10, and 5 runs, respectively. As it is possible to note, the current MILP formulations [16], and the BCP algorithm [15] can manage small-size instances in reasonable computing times. Indeed, BCP$_{fix}$ can solve medium and large-size instances with large fleet sizes. Nevertheless, for the most challenging instances, the proposed M-ILS proved to be the most effective algorithm for what concerns the solution quality, providing the best results within competitive computing times with respect to PLS, and much shorter computing times with respect to BCP. As a consequence of the stability in the performance shown by M-ILS, the number of runs needed for obtaining good quality solutions is not large.

Regarding the proposed lower bounds, we found that the average value of gap_{LB} for the 23 instances not solved to proven optimally is equal to 26.48%. Despite this value is large, it does not mean that the BKS values for these instances correspond to bad quality solutions. Indeed, the average value of gap_{LB} obtained by considering only the instances solved to proven optimally is equal to 14.31%, which means that the proposed lower bounds are not tight. It is possible to note that the proposed lower bounds provide reasonable good approximations of the optimal solution value for instances with a relative large number of vehicles. The average value of gap_{LB} obtained by considering only the instances in the data sets p-pr with $N_v = 35$ and lr is equal to 6.40%.

In order to compare the efficiency of the algorithm M-ILS with that of the algorithm PLS, new experiments were carried out to compare the quality of the solutions obtained by both algorithms

within the same global computing time, and to determine the computing time required by M-ILS to reach for each instance a given target value. Let us consider, for each instance, a target value (TV) given by the best solution value obtained by the algorithm PLS and a time limit (TL) given by the global computing time required by PLS to find the target value.

The summary of the average results of this experiment is presented in Table 3.10 for each data set and for the overall set of instances. The reported columns correspond to: the averages of the target values (TV) and of the time limits (TL), the average of the best solution value found by M-ILS within the time limit (Best_{TL}), the average of the percentage gap between Best_{TL} and TV (gap_{TV}), the number of instances for which the target value is found by M-ILS within the time limit (#TV), and the average of the computing times required by M-ILS to reach the target values (time_{TV}). It is to note that, for the instances for which M-ILS cannot find the target value, time_{TV} is equal to the time limit.

The results show that M-ILS is able to find globally better results than PLS by considering the same global computing time for both algorithms (considering all the instances). It can be noted that M-ILS can find the target value (for 11 instances) or improve it (for 37 instances) for 48 out of the 75 considered instances within the time limit, obtaining a global average value of gap_{TV} equal to -0.13%. This means that M-ILS can find better quality solutions than those obtained by PLS within computing times slightly larger than half of the times reported for PLS. It is to note that M-ILS clearly dominates PLS for the p-pr data set, which contains the most challenging instances. For the other two data sets, both algorithms provide similar results, nevertheless, for each of the three considered data sets, M-ILS is able to find or improve the target value for more than 60% of the instances within the time limit.

As we proved in the previous sections, running the proposed metaheuristic for a longer time leads to better solutions than those obtained by the PLS algorithm. Nevertheless, this experiment also proved that in general the proposed M-ILS is superior to the current state-of-the-art metaheuristic when both algorithms compete with the same conditions.

Table 3.11: Detailed results for the MDk-TRP p-pr data set with reduced fleet

Instance	N_c	N_d	N_v	BKS	Formulations			GA			M-ILS 10 runs						M-ILS 5 runs						M-ILS 1 run				
					Best	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	gap_F	gap_{GA}	Best	Avg	time	gap_B	gap_F	gap_{GA}	Best	time	gap_B	gap_F	gap_{GA}
p01	50	4	5	1958.95	**1958.95**	77.9	0.00	2852.25	1.0	45.60	**1958.95**	1970.08	196.3	0.00	0.00	-31.32	1959.16	1971.88	98.1	0.01	0.01	-31.31	1971.12	21.3	0.62	0.62	-30.89
p02	50	4	5	1958.95	**1958.95**	77.5	0.00	2117.80	1.2	8.11	**1958.95**	1970.08	196.2	0.00	0.00	-7.50	1959.16	1971.88	98.1	0.01	0.01	-7.49	1971.12	21.2	0.62	0.62	-6.93
p03	75	5	8	2271.22	**2271.22**	1886.1	0.00	2407.24	1864.3	5.99	2291.56	2301.25	454.5	0.90	0.90	-4.81	2291.56	2302.12	226.3	0.90	0.90	-4.81	2296.51	44.1	1.11	1.11	-4.60
p04	100	2	10	3122.13	**3122.13**	973.1	0.00	3271.47	1090.9	4.78	3128.16	3138.33	578.8	0.19	0.19	-4.38	3128.16	3133.11	291.7	0.19	0.19	-4.38	3128.65	56.0	0.21	0.21	-4.37
p05	100	2	10	3103.26	**3103.26**	346.8	0.00	3320.63	252.1	7.00	3110.70	3127.44	619.3	0.24	0.24	-6.32	3110.70	3126.47	316.1	0.24	0.24	-6.32	3113.37	55.9	0.33	0.33	-6.24
p06	100	3	10	2870.05	**2870.05**	501.0	0.00	3126.83	602.8	8.95	2881.00	2895.27	639.4	0.38	0.38	-7.86	2881.00	2893.13	328.6	0.38	0.38	-7.86	2881.00	60.9	0.38	0.38	-7.86
p07	100	4	10	2899.07	**2899.07**	3833.4	0.00	3087.86	2318.1	6.51	2927.51	2958.16	630.0	0.98	0.98	-5.19	2927.51	2956.66	313.7	0.98	0.98	-5.19	2972.98	69.6	2.55	2.55	-3.72
p08	249	2	25	16620.90	**16620.90**	3145.0	0.00	16648.23	226.1	0.16	16623.10	16669.80	2263.9	0.01	0.01	-0.15	16623.10	16651.60	1126.8	0.01	0.01	-0.15	16623.10	207.4	0.01	0.01	-0.15
p09	249	3	25	14732.70	14809.50	8784.0	0.52	14816.49	502.5	0.57	**14732.70**	14786.37	2474.8	0.00	-0.52	-0.57	14736.70	14779.98	1277.9	0.03	-0.49	-0.54	14793.40	225.1	0.41	-0.11	-0.16
p10	249	4	25	14054.90	14116.10	8784.0	0.44	14255.40	538.2	1.43	**14054.90**	14088.27	2342.1	0.00	-0.43	-1.41	**14054.90**	14086.50	1217.8	0.00	-0.43	-1.41	14177.50	275.2	0.87	0.43	-0.55
p11	249	5	25	13996.20	15189.20	8784.0	8.52	15209.67	454.3	8.67	**13996.20**	14084.56	2428.0	0.00	-7.85	-7.98	14009.00	14101.48	1191.6	0.09	-7.77	-7.89	14192.80	227.9	1.40	-6.56	-6.69
p12	80	2	8	5479.51	**5479.51**	3406.8	0.00	5739.15	1100.1	4.74	**5479.51**	5480.73	400.8	0.00	0.00	-4.52	**5479.51**	5480.73	195.6	0.00	0.00	-4.52	**5479.51**	40.1	0.00	0.00	-4.52
p15	160	4	16	10370.70	**10370.70**	8784.0	0.00	15834.43	1271.1	52.68	10593.90	10605.93	883.1	2.15	2.15	-33.10	10593.90	10605.72	455.1	2.15	2.15	-33.10	10593.90	99.0	2.15	2.15	-33.10
p18	240	6	24	15682.10	**15682.10**	8784.0	0.00	16884.47	715.6	7.67	15702.30	15770.61	1794.0	0.13	0.13	-7.00	15702.30	15750.56	912.3	0.13	0.13	-7.00	15702.30	172.3	0.13	0.13	-7.00
pr01	48	4	5	3036.43	**3036.43**	164.7	0.00	3245.12	3.4	6.87	**3036.43**	**3036.43**	185.4	0.00	0.00	-6.43	**3036.43**	**3036.43**	92.9	0.00	0.00	-6.43	**3036.43**	18.0	0.00	0.00	-6.43
pr02	96	4	10	4092.51	**4092.51**	1110.7	0.00	4367.49	596.5	6.72	4110.50	4120.43	463.2	0.44	0.44	-5.88	4112.54	4117.15	249.0	0.49	0.49	-5.84	4112.54	53.1	0.49	0.49	-5.84
pr03	144	4	15	6474.18	**6474.18**	4334.5	0.00	7024.86	1180.4	8.51	6482.60	6524.29	877.6	0.13	0.13	-7.72	6501.53	6531.16	419.7	0.42	0.42	-7.45	6562.09	80.3	1.36	1.36	-6.59
pr04	192	4	20	7102.26	**7102.26**	4252.0	0.00	7687.69	745.9	8.24	7114.70	7143.20	1863.6	0.18	0.18	-7.45	7114.70	7140.08	936.0	0.18	0.18	-7.45	7114.70	185.5	0.18	0.18	-7.45
pr05	240	4	24	8157.22	**8157.22**	8784.0	0.00	9061.75	397.1	11.09	8208.90	8253.24	3134.3	0.63	0.63	-9.41	8208.90	8261.97	1551.8	0.63	0.63	-9.41	8241.27	312.1	1.03	1.03	-9.05
pr06	288	4	29	9366.42	**9366.42**	8784.0	0.00	9385.54	439.7	0.20	9384.92	9418.05	4001.0	0.20	0.20	-0.01	9384.92	9427.40	1983.8	0.20	0.20	-0.01	9417.88	385.9	0.55	0.55	0.34
pr07	72	6	8	3496.68	**3496.68**	246.5	0.00	3781.00	140.3	8.13	**3496.68**	3519.80	292.3	0.00	0.00	-7.52	**3496.68**	3515.14	141.5	0.00	0.00	-7.52	**3496.68**	30.0	0.00	0.00	-7.52
pr08	144	6	15	5906.88	**5906.88**	3695.0	0.00	6319.81	2550.7	6.99	5931.75	5995.08	1108.5	0.42	0.42	-6.14	5931.75	5995.11	560.6	0.42	0.42	-6.14	5931.75	114.7	0.42	0.42	-6.14
pr09	216	6	22	7309.01	**7309.01**	1724.2	0.00	8128.12	351.8	11.21	7311.92	7391.88	2140.3	0.04	0.04	-10.04	7311.92	7400.36	1105.2	0.04	0.04	-10.04	7401.46	201.9	1.26	1.26	-8.94
pr10	288	6	29	9869.74	**9869.74**	8784.0	0.00	9876.42	425.7	0.07	9923.98	9964.28	3756.2	0.55	0.55	0.48	9934.39	9971.67	1924.7	0.66	0.66	0.59	9969.99	373.4	1.02	1.02	0.95
Global avg				7247.17	7302.62	4168.6	0.40	7852.07	740.4	9.62	**7268.4**	7300.6	1405.2	**0.32**	**-0.05**	**-7.59**	7270.4	**7300.3**	708.9	0.34	-0.03	-7.57	7299.3	**138.8**	0.71	0.34	-7.23

Table 3.12: Detailed results for the MDk-TRP p-pr data set with $N_v = 35$.

Instance	N_c	N_d	BKS	Formulations			GA			M-ILS 10 runs					M-ILS 5 runs					M-ILS 1 run				
				Best	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	gap_{GA}	Best	Avg	time	gap_B	gap_{GA}	Best	time	gap_B	gap_{GA}
p01	50	4	712.50	**712.50**	0.7	0.00	959.12	0.0	34.61	**712.50**	714.00	717.3	0.00	-25.71	712.52	713.76	317.4	0.00	-25.71	714.78	77.4	0.32	-25.18
p02	50	4	712.50	**712.50**	0.7	0.00	**712.50**	0.0	0.00	**712.50**	714.00	717.3	0.00	0.00	712.52	713.76	317.4	0.00	0.00	714.78	77.5	0.32	0.00
p03	75	5	950.25	**950.25**	6.0	0.00	958.69	6.9	0.89	**950.25**	954.02	722.9	0.00	-0.88	**950.25**	954.41	353.8	0.00	-0.88	957.51	51.9	0.76	-0.62
p04	100	2	1955.31	**1955.31**	17.5	0.00	1970.90	31.1	0.80	**1955.31**	1955.94	1291.6	0.00	-0.79	**1955.31**	1955.82	649.0	0.00	-0.79	**1955.31**	127.2	0.00	-0.79
p05	100	2	1982.33	**1982.33**	14.4	0.00	2002.56	16.3	1.02	1982.77	1984.31	1230.6	0.02	-0.99	1982.77	1984.12	651.0	0.02	-0.99	1984.80	136.6	0.12	-0.89
p06	100	3	1551.64	**1551.64**	17.8	0.00	1587.70	28.5	2.32	1552.15	1554.31	1041.8	0.03	-2.24	1552.15	1554.40	524.8	0.03	-2.24	1553.88	127.8	0.14	-2.13
p07	100	4	1520.46	**1520.46**	16.0	0.00	1546.16	29.6	1.69	1521.19	1522.41	1281.5	0.05	-1.61	1521.19	1522.28	662.6	0.05	-1.61	1521.60	135.9	0.07	-1.59
p08	249	2	15368.20	**15368.20**	1815.9	0.00	16220.42	268.9	5.55	15378.70	15394.01	3210.7	0.07	-5.19	15387.70	15393.34	1651.1	0.13	-5.13	15392.80	340.4	0.16	-5.10
p09	249	3	13047.60	**13047.60**	1446.3	0.00	14129.02	459.9	8.29	13073.60	13082.89	2895.6	0.20	-7.47	13077.50	13085.88	1457.8	0.23	-7.44	13081.60	256.6	0.26	-7.41
p10	249	4	12037.50	**12037.50**	1661.5	0.00	13087.11	427.7	8.72	**12037.50**	12145.70	2789.5	0.00	-8.02	12067.50	12152.66	1406.6	0.25	-7.79	12211.10	266.3	1.44	-7.59
p11	249	5	11932.70	**11932.70**	1315.8	0.00	13282.14	372.2	11.31	11967.10	12014.93	2720.4	0.29	-9.90	11967.10	12014.54	1359.1	0.29	-9.90	12053.50	271.1	1.01	-9.25
p12	80	2	2897.06	**2897.06**	3.8	0.00	**2897.06**	5.3	0.00	**2897.06**	**2897.06**	574.4	0.00	0.00	**2897.06**	**2897.06**	288.4	0.00	0.00	**2897.06**	59.5	0.00	0.00
p15	160	4	5794.11	**5794.11**	61.5	0.00	8563.86	325.0	47.80	**5794.11**	**5794.11**	1237.5	0.00	-32.34	**5794.11**	**5794.11**	617.4	0.00	-32.34	**5794.11**	122.9	0.00	-32.34
p18	240	6	11433.90	**11433.90**	8784.0	0.00	12033.14	622.6	5.24	11457.10	11479.31	1921.9	0.20	-4.79	11457.10	11473.96	976.9	0.20	-4.79	11478.90	182.3	0.39	-4.61
pr01	48	4	1261.53	**1261.53**	0.8	0.00	–	–	–	1262.43	1267.78	452.4	0.07	–	1262.43	1266.64	203.0	0.07	–	1275.76	20.8	1.13	–
pr02	96	4	2572.84	**2572.84**	12.8	0.00	2576.27	36.9	0.13	**2572.84**	2574.36	998.2	0.00	-0.13	**2572.84**	2574.35	529.0	0.00	-0.13	**2572.84**	102.9	0.00	-0.13
pr03	144	4	4462.50	**4462.50**	76.2	0.00	4636.24	80.1	3.89	4473.50	4478.26	1541.4	0.25	-3.51	4473.50	4478.61	749.3	0.25	-3.51	4477.15	141.2	0.33	-3.43
pr04	192	4	5804.15	**5804.15**	258.1	0.00	6026.25	310.6	3.83	5810.69	5824.67	2725.3	0.11	-3.58	5810.69	5822.00	1379.0	0.11	-3.58	5821.34	256.8	0.30	-3.40
pr05	240	4	7119.35	**7119.35**	1160.8	0.00	7527.32	296.1	5.73	7121.18	7147.74	4076.8	0.03	-5.40	7135.82	7159.51	2015.1	0.23	-5.20	7163.66	390.6	0.62	-4.83
pr06	288	4	8595.64	**8595.64**	1852.9	0.00	9492.53	405.8	10.43	8638.02	8661.39	4718.7	0.49	-9.00	8646.44	8672.22	2376.9	0.59	-8.91	8722.22	522.8	1.47	6.11
pr07	72	6	1723.63	**1723.63**	3.9	0.00	–	–	–	1725.61	1729.03	917.7	0.11	–	1725.61	1729.79	457.7	0.11	–	1729.09	100.4	0.32	–
pr08	144	6	4004.11	**4004.11**	70.4	0.00	4077.16	145.9	1.82	4007.90	4015.10	1684.2	0.09	-1.70	4015.45	4018.61	857.9	0.28	-1.51	4016.36	201.8	0.31	-1.49
pr09	216	6	5889.02	**5889.02**	432.4	0.00	6291.82	297.9	6.84	5891.65	5923.69	2991.7	0.04	-6.36	5891.65	5913.58	1484.1	0.04	-6.36	5906.75	284.3	0.30	-6.12
pr10	288	6	9108.08	**9108.08**	8784.0	0.00	17574.79	399.8	92.96	9150.65	9167.29	4656.6	0.47	-47.93	9150.65	9159.22	2336.6	0.47	-47.93	9158.48	465.9	0.55	-47.90
Global avg			5518.20	**5518.20**	1158.9	**0.00**	–	–	–	5526.93	5541.51	1963.18	0.11	–	5529.99	5541.86	984.24	0.14	–	5548.14	**196.70**	0.43	–
Global avg GA			5884.17	**5884.17**	1264.1	**0.00**	6734.22	**207.60**	11.54	5893.56	5909.07	2079.38	0.11	**-8.07**	5896.90	5909.46	1043.69	0.14	-8.03	5915.93	209.08	0.40	-7.80

Table 3.13: Detailed results for the MDk-TRP lr data set.

Instance	N_c	N_d	N_v	BKS	Formulations			GA			M-ILS 10 runs					M-ILS 5 runs					M-ILS 1 run			
					Best	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	gap_{GA}	Best	Avg	time	gap_B	gap_{GA}	Best	time	gap_B	gap_{GA}
lr1	10	4	5	545.69	**545.69**	0.6	0.00	**545.69**	0.0	0.00	**545.69**	**545.69**	10.6	0.00	0.00	**545.69**	**545.69**	5.3	0.00	0.00	**545.69**	0.7	0.00	0.00
lr2	10	4	5	832.69	**832.69**	0.1	0.00	**832.69**	0.0	0.00	**832.69**	**832.69**	29.8	0.00	0.00	**832.69**	**832.69**	14.6	0.00	0.00	**832.69**	3.0	0.00	0.00
lr3	10	4	5	832.78	**832.78**	0.3	0.00	**832.78**	0.0	0.00	**832.78**	**832.78**	16.5	0.00	0.00	**832.78**	**832.78**	8.2	0.00	0.00	**832.78**	1.6	0.00	0.00
lr4	25	4	10	2082.28	**2082.28**	0.1	0.00	**2082.28**	0.0	0.00	**2082.28**	2089.09	206.8	0.00	0.00	**2082.28**	2091.49	103.1	0.00	0.00	2109.15	23.2	1.29	1.29
lr5	25	4	10	1827.41	**1827.41**	0.1	0.00	**1827.41**	0.0	0.00	**1827.41**	1841.37	236.3	0.00	0.00	**1827.41**	1837.88	115.9	0.00	0.00	1844.86	22.3	0.95	0.95
lr6	25	4	10	1786.95	**1786.95**	0.1	0.00	**1786.95**	0.0	0.00	**1786.95**	**1786.95**	138.3	0.00	0.00	**1786.95**	**1786.95**	70.6	0.00	0.00	**1786.95**	14.5	0.00	0.00
lr7	50	4	20	5424.57	**5424.57**	1.9	0.00	5444.57	0.0	0.37	**5424.57**	**5424.57**	536.5	0.00	-0.37	**5424.57**	**5424.57**	259.1	0.00	-0.37	**5424.57**	40.7	0.00	-0.37
lr8	50	4	20	3737.38	**3737.38**	2.5	0.00	3760.03	0.0	0.61	**3737.38**	3745.99	299.8	0.00	-0.60	**3737.38**	3743.70	152.7	0.00	-0.60	**3737.38**	27.1	0.00	-0.60
lr9	50	4	20	3802.88	**3802.88**	2.5	0.00	3812.65	0.0	0.26	**3802.88**	3807.09	393.6	0.00	-0.26	3803.00	3810.07	214.8	0.00	-0.25	3803.00	44.1	0.00	-0.25
lr10-25V	50	6	25	2866.73	**2866.73**	1.5	0.00	–	–	–	2870.57	2875.00	415.9	0.13	–	2870.57	2876.24	176.2	0.13	–	2872.77	33.8	0.21	–
lr11-25V	50	6	25	2978.78	**2978.78**	1.3	0.00	–	–	–	2980.63	2989.29	428.8	0.06	–	2981.31	2990.45	199.8	0.08	–	2989.62	30.9	0.36	–
lr12-25V	50	6	25	3090.38	**3090.38**	1.5	0.00	–	–	–	**3090.38**	3094.37	420.1	0.00	–	**3090.38**	3093.64	206.8	0.00	–	3092.76	41.7	0.08	–
lr10-20V	50	6	20	2969.83	**2969.83**	1.3	0.00	2986.69	0.0	0.57	2972.04	2991.55	551.5	0.07	-0.49	2989.60	2996.12	275.4	0.67	0.10	2999.53	46.5	1.00	0.43
lr11-20V	50	6	20	3095.22	**3095.22**	2.1	0.00	3107.61	0.0	0.40	3103.22	3116.61	286.4	0.26	-0.14	3103.22	3108.71	164.9	0.26	-0.14	3113.81	33.1	0.60	0.20
lr12-20V	50	6	20	3171.75	**3171.75**	2.0	0.00	3186.67	0.0	0.47	3189.57	3194.18	419.4	0.56	0.09	3189.57	3191.56	210.8	0.56	0.09	3192.75	38.1	0.66	0.19
lr13	100	4	25	8288.43	**8288.43**	37.8	0.00	8563.04	21.4	3.31	8297.34	8310.99	774.3	0.11	-3.10	8297.34	8309.47	351.5	0.11	-3.10	8324.77	69.3	0.44	-2.78
lr14	100	4	25	7257.31	**7257.31**	30.5	0.00	7519.98	23.7	3.62	**7257.31**	7270.42	759.2	0.00	-3.49	**7257.31**	7265.17	366.6	0.00	-3.49	**7257.31**	82.6	0.00	-3.49
lr15	100	4	25	8625.13	**8625.13**	26.8	0.00	8803.73	23.2	2.07	**8625.13**	8643.69	956.7	0.00	-2.03	**8625.13**	8645.55	467.7	0.00	-2.03	8670.41	82.8	0.52	-1.51
lr16	100	6	25	5265.30	**5265.30**	28.9	0.00	5399.68	26.4	2.55	**5265.30**	5278.97	620.9	0.00	-2.49	**5265.30**	5288.72	326.7	0.00	-2.49	5268.68	77.9	0.06	-2.43
lr17	100	6	25	6107.32	**6107.32**	28.7	0.00	6353.43	24.3	4.03	**6107.32**	**6107.32**	676.8	0.00	-3.87	**6107.32**	**6107.32**	323.8	0.00	-3.87	**6107.32**	82.3	0.00	-3.87
lr18	100	6	25	5788.73	**5788.73**	30.5	0.00	6017.02	21.0	3.94	**5788.73**	5811.32	756.8	0.00	-3.79	**5788.73**	5805.32	391.3	0.00	-3.79	**5788.73**	70.8	0.00	-3.79
Global avg				3827.50	**3827.50**	**9.6**	**0.00**	–	–	–	3829.53	3837.61	425.48	0.06	–	3830.41	3837.34	209.80	0.09	–	3837.88	41.27	0.29	–
Global avg GA				3968.98	**3968.98**	10.9	**0.00**	4047.94	**7.77**	1.23	3971.03	3979.51	426.12	0.06	**-1.14**	3972.01	3979.10	212.39	0.09	-1.11	3980.02	42.24	0.31	-0.89

Table 3.14: Detailed results for the MDk-TRP lr data set with reduced fleet .

Instance	N_c	N_d	N_v	BKS	Formulations			GA			M-ILS 10 runs				M-ILS 5 runs				M-ILS 1 run			
					Best	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	gap_{GA}	Best	Avg	time	gap_B	gap_{GA}	Best	time	gap_B	gap_{GA}
lr1	10	4	4	592.27	**592.27**	0.0	0.00	**592.27**	0.0	0.00	**592.27**	**592.27**	19.9	0.00	0.00	**592.27**	**592.27**	10.0	0.00	0.00	**592.27**	1.5	0.00	0.00
lr2	10	4	4	885.89	**885.89**	0.0	0.00	**885.89**	0.0	0.00	**885.89**	**885.89**	23.8	0.00	0.00	**885.89**	**885.89**	12.0	0.00	0.00	**885.89**	2.4	0.00	0.00
lr3	10	4	4	846.91	**846.91**	0.1	0.00	**846.91**	0.0	0.00	**846.91**	**846.91**	24.2	0.00	0.00	**846.91**	**846.91**	12.2	0.00	0.00	**846.91**	2.4	0.00	0.00
lr4	25	4	4	3267.66	**3267.66**	2.9	0.00	–	–	–	3300.74	3300.74	70.9	1.01	–	3300.74	3300.74	36.0	1.01	–	3300.74	7.3	1.01	–
lr5	25	4	4	3206.83	**3206.83**	1.5	0.00	3267.66	0.0	1.90	**3206.83**	**3206.83**	79.1	0.00	-1.86	**3206.83**	**3206.83**	39.4	0.00	-1.86	**3206.83**	8.6	0.00	-1.86
lr6	25	4	4	2965.13	**2965.13**	3.3	0.00	2965.14	0.0	0.00	**2965.13**	2975.84	52.9	0.00	0.00	**2965.13**	**2965.13**	27.3	0.00	0.00	**2965.13**	5.0	0.00	0.00
lr7	50	4	5	8325.97	**8325.97**	332.4	0.00	8371.32	3.9	0.54	**8325.97**	**8325.97**	195.2	0.00	-0.54	**8325.97**	**8325.97**	95.5	0.00	-0.54	**8325.97**	17.9	0.00	-0.54
lr8	50	4	5	7089.64	**7089.64**	4099.1	0.00	7143.59	21.5	0.76	**7089.64**	**7089.64**	151.6	0.00	-0.76	**7089.64**	**7089.64**	76.9	0.00	-0.76	**7089.64**	16.4	0.00	-0.76
lr9	50	4	5	7390.97	**7390.97**	732.4	0.00	7417.23	2.3	0.36	**7390.97**	7434.33	167.8	0.00	-0.35	**7390.97**	7446.88	85.1	0.00	-0.35	7516.49	18.4	1.70	1.34
lr10	50	6	6	6143.47	**6143.47**	427.1	0.00	6598.34	0.3	7.40	**6143.47**	6154.12	190.4	0.00	-6.89	6151.12	6157.07	97.9	0.12	-6.78	6151.12	19.3	0.12	-6.78
lr11	50	6	6	5756.81	**5756.81**	214.9	0.00	6168.67	4.7	7.15	5785.93	5817.54	171.8	0.51	-6.20	5785.93	5809.85	87.6	0.51	-6.20	5849.86	18.5	1.62	-5.77
lr12	50	6	6	5699.16	**5699.16**	85.3	0.00	5730.72	1.9	0.55	**5699.16**	5706.05	147.2	0.00	-0.55	**5699.16**	5705.44	74.1	0.00	-0.55	**5699.16**	14.0	0.00	-0.55
lr13	100	4	10	11744.60	**11744.60**	3279.2	0.00	12161.63	1563.5	3.55	11777.50	11890.21	341.5	0.28	-3.16	11812.50	11896.26	162.8	0.58	-2.87	11963.00	29.5	1.86	-1.63
lr14	100	4	10	10742.60	**10742.60**	654.6	0.00	11160.95	355.4	3.89	**10742.60**	10827.05	522.8	0.00	-3.75	10760.00	10821.74	257.8	0.16	-3.59	10799.50	52.3	0.53	-3.24
lr15	100	4	10	11035.60	**11035.60**	159.7	0.00	12033.57	2261.5	9.04	**11035.60**	11049.89	478.1	0.00	-8.29	**11035.60**	11064.18	232.6	0.00	-8.29	**11035.60**	46.9	0.00	-8.29
lr16	100	6	10	9442.62	**9442.62**	3537.5	0.00	9685.37	1657.4	2.57	9574.04	9657.29	350.0	1.39	-1.15	9574.04	9636.79	186.5	1.39	-1.15	9680.04	31.8	2.51	-1.06
lr17	100	6	10	9917.56	**9917.56**	1303.2	0.00	10343.79	859.9	4.30	9942.78	10048.18	554.4	0.25	-3.88	9945.93	10052.49	271.9	0.29	-3.85	10069.60	64.4	1.53	-3.55
lr18	100	6	10	9220.77	**9220.77**	3074.7	0.00	9604.71	1251.3	4.16	9240.67	9361.50	435.8	0.22	-3.79	9240.67	9304.71	215.7	0.22	-3.79	9277.06	48.8	0.61	-3.41
Global avg				6348.58	**6348.58**	994.9	**0.00**	–	–	–	6363.67	6398.35	220.96	0.20	–	6367.18	6394.93	110.07	0.24	–	6403.05	**22.53**	0.64	–
Global avg GA				6529.81	**6529.81**	1053.2	**0.00**	6763.40	469.6	2.72	6543.85	6580.56	229.78	0.16	-2.42	6547.56	6576.94	114.43	0.19	-2.39	6585.53	**23.43**	0.62	-2.98

3.3.4 The multi-depot k-traveling repairman problem

In this sections we compare the proposed M-ILS algorithm with the two mathematical formulations and the two configurations of the genetic algorithm (GA in the tables) presented in [10], the only work in the literature dealing with the multi-depot k-traveling repairman problem. For the MDk-TRP, the M-ILS algorithm is executed for each instance with a number of runs equal to 10, 5 and 1. It is to note that both configurations of the algorithm GA are executed with a single run (the stopping criterion being defined by the maximum number of iterations, whose value is defined depending on the size of the instance.)

In order to present a fair comparison between M-ILS and the solution methods proposed in [10], for each instance the global computing times presented in Tables 3.11-3.14 for the latter methods correspond to the scaled values (using a scaling factor equal to 0.61) of:
i) the sum of the computing times reported in [10] for each configuration of GA (since there is no dominance between the two configurations), and
iia) the computing time reported in [10] of the dominant formulation (Model 2) when the time limit is not reached, or *iib*) the sum of the computing times of both formulations when the time limit is reached for Model 2. Furthermore, for each number of runs of M-ILS and each instance, also the following values are reported:

- gap_{GA}: Percentage gap between the Best solution value found by M-ILS ($Best$) and the Best solution value found by GA (Best_{GA}), computed as $\text{gap}_{GA} = 100\frac{(Best - Best_{GA})}{Best_{GA}}$.
- gap_F: Percentage gap between the Best solution value found by M-ILS and the Best solution value found by the Formulations (Best_F), computed as $\text{gap}_F = 100\frac{(Best - Best_F)}{Best_F}$.

The p-pr data set with reduced fleet

The computational results corresponding to the 24 instances of the p-pr data set with reduced fleet are reported in Table 3.11. This data set was proposed in [10], and contains the most challenging MDk-TRP instances due to the large number of customers (N_c) and the small number of vehicles (N_v). According to the results presented in Table 3.11, M-ILS (executed with 10 and 5 runs) can find globally better solutions than those obtained by the formulations (with values of gap_F equal to -0.05 and -0.03, respectively) in shorter computing times (almost three and six times smaller, respectively). The maximum value of gap_B associated with M-ILS, executed with 10 or 5 runs, is equal to 2.15%, while the maximum value of gap_B associated with the formulations is equal to 8.52%. M-ILS, executed with 10 runs, can find the proven optimal solution value for 5 instances and provides new best-known solution values for 3 large instances. The corresponding values of LB and gap_{LB} for the 8 instances for which the formulations have not been solved to proven optimally are the following: p09: 11742.3 (25.47%), p10: 10478.5 (34.13%), p11: 9883.96 (41.61%), p15: 9048.53 (14.61%), p18: 13535.5 (15.86%), pr05: 6436.69 (26.73%), pr06: 7434.05 (25.99%), and pr10: 7645.9 (29.09%).

Compared to GA, M-ILS is superior in terms of solution quality, improving the global best solution value of GA by over 7%, even when a single run is executed. M-ILS can find a better best solution value than that found by GA for all the instances but one, when 10 and 5 runs are executed, and for all the instances but 2 when a single run is performed. The average solution value obtained by M-ILS by executing 10 or 5 runs is better than the best solution value obtained by GA for all the instances but three. The global computing time required for executing M-ILS with 5 runs is slightly smaller than the global computing time required by GA. Nevertheless, the quality of the solutions provided by M-ILS is clearly better. In addition, when M-ILS is executed

with a single run, the global computing time is 5 times smaller than the global computing time required by GA, and the quality of the solutions provided by M-ILS is still considerably better than that of GA.

The p-pr data set with N_v=35

Table 3.12 presents the results for the p-pr data set with $N_v = 35$, obtained by the p-pr data set with the reduced fleet by setting $N_v = 35$ for all 24 instances. We found that for 10 instances of this data set, the values reported in [10] as optimal/best solution values found by the formulations were smaller than the corresponding LB values [1].The 10 instances with the respective values reported in [10], and the corresponding LB values are the following: p01: 660.34 < 707.68, p02: 660.34 < 707.68, p04: 1881.48 < 1926.16, p05: 1871.62 < 1956.75, p06: 1460.6 < 1500.48, p12: 2769.07 < 2897.06, p15: 5618.84 < 5794.11, pr01: 1167.74 < 1260.41, pr02: 2422.94 < 2527.15, and pr07: 1594.15 < 1691.14.

In order to determine the right solution values found by the two MILP formulations proposed in [10] for the instances of this data set, we implemented both MILP formulations and solved all the instances on our computer using the MILP solver Gurobi 9.1.2 [35] with a time limit of 4392 s, which is the scaled value of the time limit of 7200 s imposed in [10]. First, we solved Model 2; only in case the time limit is reached we solved Model 1, and the best solution value found is reported in column Best. All the values presented in Table 3.12 for the MILP formulations correspond to those found by our implementation and should be used in future research. It is to note that all the instances but p18 and pr10 were solved to proven optimality within the time limit. For the instance p18 the best lower bound found was equal to 11364.06, implying an optimality gap equal to 0.61%, while for the instance pr10, the best lower bound found was equal to 9082.66, implying an optimality gap equal to 0.28%. These lower bounds are tighter than LB (11364.06 > 9934.63, and 9082.66 > 7645.90, for the instances p18 and pr10, respectively), and should be used in future research as best lower bounds for these instances. In the Appendix the optimal solutions for two instances are presented. In the results reported in [10], the solution values obtained by algorithm GA for the instances pr01 and pr07 are smaller than the corresponding optimal solution values, hence these values are not considered in Table 3.12. It is to note that Table 3.12 does not include the values of gap_F since the formulations always provide the best-known solution values, which implies that gap_F is equal to gap_B for all the instances of this data set.

According to the results shown in Table 3.12, M-ILS can find the optimal solution values for 8, 5, and 4 instances by performing 10 run, 5 runs, and 1 run, respectively. For all the other instances, it can provide near-optimal solutions independently of the number of runs. The global average value of gap_B is equal to 0.11% by executing M-ILS with 10 runs, 0.14% when it is executed with 5 runs and 0.43% considering a single run. Although, by executing 10 runs for each instance, M-ILS is more time-consuming than the formulations, by considering 5 runs, the global computing time is slightly smaller than that required by the formulations, and by considering a single run, M-ILS is six times faster than the formulations.

Compared to GA, M-ILS is superior in the solution quality since the global value of gap_{GA} is around -8%, independently of the number of runs. For all the instances but two, the best solution value provided by M-ILS is better than that provided by GA (independently of the number of runs). For the two remaining instances, when 10 runs are considered, M-ILS and GA

[1]We have jointly checked the results of the considered instances with the authors of [10], who acknowledged an error in the procedure used to read the input files.

find the same best (optimal) solution value for both instances, while when 5 runs are considered, each algorithm finds a solution better than that of the competitor for one instance. Furthermore, the Avg. solution value provided by M-ILS, by considering 10 and 5 runs, is smaller than (for 20 instances) or equal to (for one instance) the best solution value obtained by GA for all the instances but one. Regarding the computing times of the two metaheuristics, M-ILS is more time-consuming than GA when it is executed with 10 and 5 runs, while it is almost the same when it is executed with a single run.

The lr data set

The computational results corresponding to the 21 instances of the lr data set are reported in Table 3.13. Since the instances lr10, lr11, and lr12 with 25 vehicles were not considered in [10], we solved them optimally under the same conditions mentioned in Section 3.3.4. The values reported in [10] for both GA and the formulations are presented for all the other instances. This is the easiest of the considered data sets since all its instances have been solved to proven optimality by the formulations within very short computing times (at most 37.8 seconds).

The results presented in Table 3.13 show that M-ILS can find the optimal solution value for 15, 14, and 9 instances by performing 10 runs, 5 runs, and 1 run, respectively. For all the other instances, M-ILS can provide near-optimal solution values independently of the number of runs. The average value of gap_B is equal to 0.06% by performing 10 runs, 0.09% for 5 runs and 0.29% considering a single run. M-ILS is more time-consuming than both the formulations and GA, independently of the number of runs. Nevertheless, by executing only one run, M-ILS provides good quality solutions in reasonable computing times (on average around 40 s). For what concerns the values of the solutions provided by M-ILS compared to those obtained by GA, it is possible to conclude that M-ILS outperforms GA independently of the number of runs. By executing M-ILS with 10 runs, the best solution value found by M-ILS is better than (for 12 instances) or equal to (for 5 instances) the best solution value provided by GA for all the instances but one. Similarly, by executing M-ILS with 5 runs, the best solution value found by M-ILS is better than (for 11 instances) or equal to (for 5 instances) the best solution value provided by GA for all the instances but two. By executing a single run, the best solution value provided by M-ILS is better than (for 10 instances) or equal to (for 3 instances) the best solution value provided by GA for all the instances but 5. The good performance of M-ILS is more evident for the large size instances of this data set (N_C=100) for which the average value of gap_{GA} is equal to -2.98% (considering a single run for M-ILS).

The lr data set with reduced fleet

The computational results corresponding to the 18 instances of the lr data set with reduced fleet are reported in Table 3.14. Since for the instances lr5 and lr15 the best solution values of the feasible solutions found by M-ILS are smaller than the optimal solution values reported in [10], we solved these two instances optimally under the same conditions mentioned in Section 3.3.4. The optimal solution values presented in Table 3.14 are the correct ones, and should be used in future research. For what concerns the results reported in [10] regarding GA, the best solution value presented for the instance lr4 is smaller than the optimal solution value, so this value is not considered in Table 3.14. The original values reported in [10] are presented for all the other instances. It is to note that all the instances of this data set have been solved to proven optimality with the formulations.

The results presented in Table 3.14 show that M-ILS can find the optimal solution value for

12, 10 , and 9 instances by performing 10 runs, 5 runs, and 1 run, respectively. The proposed algorithm can find near-optimal solution values for all the remaining instances, obtaining an average value of gap_B equal to 0.20%, 0.24%, and 0.64% when the number of runs executed for each instance is 10, 5, and 1, respectively. Regarding the global computing times, M-ILS is more than four times faster than the formulations and two times faster than GA when 10 runs are performed. M-ILS outperforms GA for what concerns both the computing time and the solution quality. When the number of runs executed is equal to 10 or 5 for each instance, M-ILS finds better solution values than those obtained by GA for 14 instances and the same solution value for the remaining 3 instances. When a single run is considered for M-ILS, it finds solution values better than (for 13 instances) or equal to (for 3 instances) those obtained by GA for all the instances but one. In addition, the average solution value obtained by M-ILS by performing 10 runs is better than (for 12 instances) or equal to (for 3 instances) the best solution value obtained by GA for all the instances but two. Similarly, the average solution value obtained by M-ILS by performing 5 runs is better than (for 13 instances) or equal to (for 3 instances) the best solution value obtained by GA for all the instances but one. M-ILS obtained an average value of gap_{GA} equal to -2.42%, -2.39%, and -1.98% when the number of runs executed for each instance is 10, 5, and 1, respectively. It is to note also that for 6 (7) instances, the average solution value found by M-ILS corresponds to the optimal solution value when 10 (5) runs are performed.

Overall results on the MDk-TRP

After analyzing the results, it is possible to conclude that the proposed algorithm M-ILS overcomes GA in terms of solution quality for all the considered data sets by executing a number of runs equal to 10, 5, or 1 for each instance. The global average value of gap_{GA} is equal to -5.20%, -5.17%, and -4.87% when M-ILS is executed with 10, 5, and 1 runs, respectively. By considering a single run, M-ILS is globally faster than GA for 2 of the 4 data sets, while the average global computing time is similar and competitive (less than 60 s) for the remaining two data sets. By considering the global average computing time (computed over all the instances), M-ILS (executed with 1 run) is three times faster than GA (112.21 s vs. 376.05 s, respectively). For the two data sets with reduced fleets (which contain the most challenging instances), M-ILS is faster than GA when 5 runs are executed. Compared to the formulations, M-ILS finds optimal or near-optimal solution values in globally shorter computing times for most of the considered instances. The global average value of gap_B (computed over all the instances) is equal to 0.11% for the formulations, and 0.17% for M-ILS (when 10 runs are executed), while the average global computing time is 1677.82 s for the formulations and 1077.61 s for M-ILS (when 10 runs are executed).

Regarding the proposed lower bounds, we found that the average value of gap_{LB} for the 10 instances not solved to proven optimality is equal to 24.77%. Also for this problem, despite this value is large, it does not mean that the BKS values for these instances correspond to bad quality solutions. Indeed, the average value of gap_{LB} obtained by considering only the instances solved to proven optimality is equal to 20.52%, which means that the proposed lower bounds are not tight. It is possible to note that the proposed lower bounds provide reasonable good approximations of the optimal solution value for instances with a relative large number of vehicles. The average value of gap_{LB} obtained by considering only the instances in the data sets p-pr with $N_v = 35$ and lr is equal to 6.38%, while when only the instances in the data sets p-pr and lr with reduced fleet are considered, the value obtained is equal to 36.68%.

In order to compare the efficiency of the algorithm M-ILS with that of the algorithm GA, new

Table 3.15: Average results obtained by each metaheuristic considering time limits and target values for each MDk-TRP data set.

Data set	# Instances	GA		M-ILS			
		TV	TL	Best_{TL}	gap_{TL}	#TV	time_{TL}
p-pr with reduced fleet	24	7852.07	740.4	**7293.30**	**-7.08**	**22**	**43.7**
p-pr with N_v=35	22	6734.22	207.6	**5910.76**	**-7.75**	**19**	**18.0**
lr	18	4047.94	7.8	**4001.30**	**-0.15**	9	**6.1**
lr with reduced fleet	17	6763.40	469.6	**6573.52**	**-2.02**	**15**	**10.6**
All the instances	81	6474.61	376.1	**6035.17**	**-4.66**	**65**	**21.4**

experiments were carried out to compare the quality of the solutions obtained by both algorithms within the same global computing time, and to determine the computing time required by M-ILS to reach for each instance a given target value. Let us consider, for each instance, a target value (TV) given by the best solution value obtained by the algorithm GA and a time limit (TL) given by the global computing time required by GA to find the target value.

The summary of the average results of this experiment is presented in Table 3.15 for each data set and for the overall set of instances. The reported columns correspond to: the averages of the target values (TV) and of the time limits (TL), the average of the best solution value found by M-ILS within the time limit (Best_{TL}), the average of the percentage gap between Best_{TL} and TV (gap_{TV}), the number of instances for which the target value is found by M-ILS within the time limit (#TV), and the average of the computing times required by M-ILS to find the target values (time_{TV}). It is to note that, for the instances for which M-ILS cannot find the target value, time_{TV} is equal to the time limit. For the instances for which TL is equal to 0.0 we allow M-ILS to perform only the Constructive phase, which generally requires small computing times, and for small instances requires less than 1 s.

The results show that the M-ILS algorithm can find (for 7 instances) or improve (for 58 instances) the target value for 65 out of 81 instances within the time limit, obtaining a global average value of gap_{TV} equal to -4.50%. Considering all the instances, M-ILS requires considerably less computing time than that required by GA for finding solutions with similar quality, and when M-ILS is executed for the same global time reported for GA it is able to largely outperform GA in terms of solution quality. The only data set for which M-ILS does not totally dominate GA is the lr data set. For these instances M-ILS is generally able to perform only the Constructive procedure. Although the results of both heuristics are competitive in terms of solution quality and computing time, M-ILS performs slightly better than GA, being able to find the target value for half of the instances within the time limit. On the other hand, the average computing times required by GA for finding the target values are 17, 12, and 44 times larger than those required by M-ILS for the p-pr with reduced fleet, p-pr with $N_v = 35$, and lr with reduced fleet data sets, respectively.

The computational experiments presented in this Section show that M-ILS clearly outperforms GA both in terms of solution quality and computing time.

3.3.5 The latency location routing problem

The algorithms proposed in the literature for the solution of the LLRP are the following: the memetic algorithm (MA) and the recursive granular algorithm (RGA) proposed in [11], the exact methods and the GRASP-based iterated local search algorithm (GBILS) presented in [16], and the three simulated annealing-variable neighborhood descent based metaheuristics SA-VND0,

SA-VND1, and SA-VND2 presented in [22]. For the LLRP, the M-ILS algorithm is executed for each instance with a number of runs equal to 30 and 5.

The results reported in [11] indicate that the algorithm MA dominates the RGA approach. Besides, the exact methods proposed in [16] are able to solve to optimality only instances with up to 50 customers. In particular, they solve the instances with 20 customers, from 21 to 36 customers, and with 50 customers, with average scaled computing times equal to 28, 465, and 5312 seconds, respectively. For all the instances with more than 50 customers, the exact methods are able to find, within a time limit of 21960 seconds, feasible solutions with an average value of gap_B (considering the values of BKS reported in Tables 3.17 and 3.18) equal to 5.05%. For large-size instances, the metaheuristic algorithms GBILS, SA-VND0, SA-VND1, and SA-VND2 perform globally better than the other methods for what concerns both the solution quality and the computing times. According to the results presented in [22], the algorithms SA-VND0, SA-VND1, and SA-VND2 are also more effective than RGA for all the considered instances. Therefore, we have excluded the algorithm RGA ([11]) and the exact methods ([16]) for comparison purposes. On the other hand, despite the algorithms SA-VND0, SA-VND1, and SA-VND2 have been proved to outperform the GBILS approach, the latter is included in the comparison since it presents relatively good solution quality and short computing times. Note that no average solution values have been reported in [16] for GBILS.

In order to present a fair comparison, the global computing times reported in Tables 3.16, 3.17 and 3.18 for each instance correspond to:

i) for MA to the times reported in [11], which correspond to the execution of 30 runs. The experiments in [11] were performed on a 3.1 GHz computer with 4 GB RAM. The above is the only information available about this computer, and it does not allow us to determine a scaling factor. Nevertheless, considering the ratio between the corresponding values of GHz, it is possible to estimate that our computer is about 1.2 times faster than that used in [11].

ii) for GBILS to the scaled computing time (considering a scaling factor equal to 0.61) of the times reported in [16], which correspond to the execution of 5 runs;

iii) for SA-VND0, SA-VND1, and SA-VND2 to the computing time reported in [22], corresponding to the global computing time associated with 30 runs for each algorithm. It is to note that the computing times of the mentioned algorithms do not need to be scaled since these algorithms were executed on the same computer on which the M-ILS has been executed;

The computing times reported for M-ILS correspond to the average computing times multiplied by the respective number of runs.

Furthermore, for each number of runs of M-ILS, the following values are reported:

- gap_{MA}: Percentage gap between the Best solution value found by M-ILS ($Best$) and the Best solution value found by MA (Best_{MA}), computed as $gap_{MA} = 100\frac{(Best - Best_{MA})}{Best_{MA}}$.
- gap_{SA}: Percentage gap between the Best solution value found by M-ILS ($Best$) and the Best solution value found by the best among SA-VND0, SA-VND1, and SA-VND2 (Best_{SA}), computed as $gap_{SA} = 100\frac{(Best - Best_{SA})}{Best_{SA}}$.
- gap_{GBILS}: Percentage gap between the Best solution value found by M-ILS and the Best solution value found by GBILS (Best_{GBILS}), computed as $gap_{GBILS} = 100\frac{(Best - Best_{GBILS})}{Best_{GBILS}}$.

The Tuzun–Burke data set

This data set contains the most challenging benchmark instances for the LLRP. Table 3.16 gives the corresponding results. Since no results for GBILS are reported on this data set in [16],

M-ILS is compared only with the MA algorithm presented in [11], and the three metaheuristics presented in [22]. As the table indicates, M-ILS outperforms MA, SA-VND0, SA-VND1, and SA-VND2 regarding both the solution quality and the computing time.

The proposed M-ILS improves the best-known solution value for 32 out of the 36 instances of this data set. The average values of gap_B are 5.56%, 0.77%, 0.84%, and 0.86% for MA, SA-VND0, SA-VND1, and SA-VND2, respectively, while this value is equal to 0.04% for M-ILS. Regarding the computing times, M-ILS is more than two times faster than SA-VND1 and SA-VND2 and 1.5 times faster than SA-VND0 (considering 30 runs for each algorithm). On the other hand, when 30 runs are considered for M-ILS, its computing times are larger than those of MA; nevertheless, when the number of runs is reduced to 5, M-ILS is three times faster than MA, being able to provide a better solution value than MA for all the instances. Furthermore, when 5 runs are considered, M-ILS provides a better solution value than the best found by SA-VND0, SA-VND1, and SA-VND2 for 22 instances, in global computing times more than 9 times shorter than those required by SA-VND0, which is the fastest among the three mentioned algorithms.

The average solution value obtained by M-ILS is better than that obtained by MA, SA-VND0, SA-VND1, and SA-VND2 for 36, 33, 34, and 32 instances, respectively. Furthermore, for seven instances, the average solution value obtained by M-ILS is better than the best solution value obtained by the best among the four competitors. Note that the average solution value provided by M-ILS is better than the best solution value reported for MA for all the instances but one (considering 30 runs). In the same direction, the average gap_{SA} value equals -0.51%, and -0.22%, and the average gap_{MA} value equals -5.17%, -4.90%, when 30 and 5 runs are considered for M-ILS, respectively.

The Prodhon data set

The results of this data set are presented in Table 3.17. Over the 30 instances of this data set, M-ILS, executed with 30 runs, can find the proved optimal solution value for 11 instances, provides new best-known solution values for 16 instances, and finds the current best-known solution value for one instance. The average values of gap_B obtained by M-ILS are equal to 0.05% and 0.19% considering 30 and 5 runs, respectively. These values of gap_B are better than those associated with all the competitors. It is to note that instance 50-5-3 was not solved to proven optimality in [16]. Nevertheless, we implemented the MILP formulation "Model 2" presented in [16] and solved this instance without considering a time limit, proving that the best solution value reported in Table 3.17 corresponds to the optimal one.

Comparing M-ILS with the current state-of-the-art metaheuristics proposed in [22], it is possible to conclude that M-ILS outperforms the three algorithms SA-VND0, SA-VND1, and SA-VND2, obtaining an average value of gap_{SA} equal to -0.24%, and -0.09% when 30, and 5 runs are considered, respectively. Regarding the computing time, M-ILS is 1.7, 2.7, and 2.5 times faster than SA-VND0, SA-VND1, and SA-VND2, respectively (considering 30 runs for each algorithm). When 5 runs are considered, M-ILS can find solution values better than (for 13 instances) or equal to (for 9 instances) the best found by SA-VND0, SA-VND1, and SA-VND2 for 22 instances, in global computing times more than ten times smaller than those required by SA-VND0, which is the fastest among the three mentioned algorithms. Although GBILS is faster than M-ILS (independently of the number of runs), the values of the solutions obtained by M-ILS are considerably better than those found by GBILS. The average value of gap_{GBILS} equals -2.40% and -2.26% considering 30 and 5 runs, respectively. Furthermore, the average solution value obtained by M-ILS (considering 30 runs) is better than or equal to the best solution

value obtained by GBILS for all the instances but 3. Also, considering 5 runs, M-ILS provides a solution value better than (for 25 instances) or equal to (for 4 instances) the best solution value reported for GBILS for all the instances but one. Finally, M-ILS is able to provide a solution value better than (for 28 instances) or equal to (for one instance) that reported for MA for all the instances but one, independently of the number of runs. When 30 runs are considered, M-ILS is more time-consuming than MA, obtaining an average gap_{MA} equal to -3.77%; nevertheless, considering 5 runs, M-ILS is 3.3 times faster than MA, providing an average value of gap_{MA} equal to -3.63%.

Table 3.16: Detailed results for the LLRP Tuzun-Burke data set

Instance	N_c	N_d	N_v	BKS	MA				SA-VND0				SA-VND1				SA-VND2				M-ILS 30 runs						M-ILS 5 runs					
					Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	gap_{MA}	gap_{SA}	Best	Avg	time	gap_B	gap_{MA}	gap_{SA}
111112	100	10	11	3834.91	4017.90	4270.10	1200.0	4.77	3862.86	3971.50	1912.8	0.73	3892.97	3972.93	2390.6	1.51	3882.76	3964.12	2801.2	1.25	**3834.91**	3890.13	2149.2	0.00	-4.55	-0.72	3849.10	3895.17	342.0	0.37	-4.20	-0.36
111122	100	20	11	3612.36	3719.10	4087.50	1200.0	2.95	**3612.36**	3694.70	1934.6	0.00	3633.60	3712.64	2419.2	0.59	3623.69	3687.85	2860.6	0.31	3659.46	3695.69	2130.6	1.30	-1.60	1.30	3683.58	3692.65	342.1	1.97	-0.96	1.97
111212	100	10	10	3919.74	4264.40	4563.50	1245.0	8.79	3960.24	4038.24	1805.4	1.03	3988.11	4067.91	2335.5	1.74	3938.88	4066.13	2692.0	0.49	**3919.74**	4000.94	2455.1	0.00	-8.08	-0.49	**3919.74**	3993.54	394.9	0.00	-8.08	-0.49
111222	100	20	11	4065.04	4278.30	4557.30	1242.0	5.25	4086.74	4140.33	1909.6	0.53	4077.87	4147.90	2376.2	0.32	4068.34	4138.78	2785.2	0.08	**4065.04**	4139.64	2312.7	0.00	-4.98	-0.08	**4065.04**	4125.54	379.0	0.00	-4.98	-0.08
112112	100	10	11	2726.41	2795.60	3049.70	1242.0	2.54	2739.16	2755.53	2281.4	0.47	2740.21	2759.43	2908.3	0.51	2741.82	2756.07	3271.1	0.57	**2726.41**	2749.56	1649.9	0.00	-2.47	-0.47	2747.04	2751.00	265.7	0.76	-1.74	0.29
112122	100	20	11	2057.30	2097.90	2702.60	1329.0	1.97	2060.29	2072.57	2211.6	0.15	2057.45	2078.03	2797.4	0.01	2060.80	2071.85	3205.3	0.17	**2057.30**	2063.89	1193.4	0.00	-1.94	-0.01	2059.08	2059.28	169.6	0.09	-1.85	0.08
112212	100	10	12	1394.65	1442.20	1604.90	1455.0	3.41	1402.97	1416.20	2349.9	0.60	1403.57	1416.72	3050.0	0.64	1397.39	1414.98	3335.3	0.20	**1394.65**	1411.85	1490.7	0.00	-3.30	-0.20	**1394.65**	1411.25	240.1	0.00	-3.30	-0.20
112222	100	20	11	1621.40	1659.00	2084.10	1314.0	2.32	1623.69	1633.00	2522.2	0.14	**1621.40**	1633.94	3137.8	0.00	1626.86	1633.89	3562.2	0.34	1623.39	1630.36	1652.2	0.12	-2.15	0.12	1623.39	1636.93	337.6	0.12	-2.15	0.12
113112	100	10	11	2828.24	2897.70	3162.60	1170.0	2.46	2837.51	2852.63	2043.2	0.33	2835.76	2853.57	2735.6	0.27	2839.50	2857.50	3014.0	0.40	**2828.24**	2841.93	2091.6	0.00	-2.40	-0.27	2837.82	2843.77	377.9	0.34	-2.07	0.07
113122	100	20	11	2772.98	2912.00	3330.00	1248.0	5.01	2776.38	2782.52	2063.4	0.12	2774.36	2784.42	2762.2	0.05	2776.38	2782.07	2998.4	0.12	**2772.98**	2797.08	2753.4	0.00	-4.77	-0.05	**2772.98**	2799.46	445.6	0.00	-4.77	-0.05
113212	100	10	12	1815.62	1832.10	1901.00	1458.0	0.91	1817.00	1823.15	2326.4	0.08	**1815.62**	1822.81	2922.2	0.00	**1815.62**	1822.76	3326.7	0.00	1817.00	1835.80	2023.4	0.08	-0.82	0.08	1817.33	1832.76	382.7	0.09	-0.81	0.09
113222	100	20	11	1876.14	2037.50	2360.60	1239.0	8.60	**1876.14**	1888.46	2117.2	0.00	1879.63	1890.96	2816.5	0.19	1876.93	1888.90	3063.5	0.04	1876.58	1885.31	1940.0	0.02	-7.90	0.02	1882.30	1888.46	323.2	0.33	-7.62	0.33
131112	150	10	16	5410.96	5863.80	6160.70	1797.0	8.37	5473.17	5582.94	4060.2	1.15	5464.21	5570.12	6108.7	0.98	5448.86	5576.01	6409.4	0.70	**5410.96**	5478.43	3854.6	0.00	-7.72	-0.70	5434.61	5490.75	653.7	0.44	-7.32	-0.26
131122	150	20	16	4926.87	5310.30	5649.30	1746.0	7.78	4993.36	5142.06	4462.0	1.35	5009.26	5143.19	6549.2	1.67	4974.28	5105.72	6876.6	0.96	**4926.87**	5053.13	3970.3	0.00	-7.22	-0.95	5003.55	5060.78	663.0	1.56	-5.78	0.59
131212	150	10	16	5528.85	5967.60	6234.20	1746.0	7.94	5679.70	5787.34	4588.6	2.73	5606.31	5785.18	6765.0	1.40	5653.20	5771.63	7051.5	2.25	**5528.85**	5639.90	4007.7	0.00	-7.35	-1.38	**5528.85**	5615.26	693.1	0.00	-7.35	-1.38
131222	150	20	16	5060.71	5261.00	5610.40	1767.0	3.96	5141.89	5284.45	4580.2	1.60	5126.95	5277.65	6627.3	1.31	5134.39	5296.62	7093.9	1.46	**5060.71**	5107.53	4093.6	0.00	-3.81	-1.29	**5060.71**	5090.48	685.6	0.00	-3.81	-1.29
132112	150	10	16	3850.90	4026.70	4246.20	1746.0	4.57	3868.88	3895.91	5963.0	0.47	3883.40	3899.41	8542.7	0.84	3874.44	3894.29	8771.5	0.61	**3850.90**	3881.70	4034.9	0.00	-4.37	-0.46	3856.34	3874.13	573.8	0.14	-4.23	-0.32
132122	150	20	16	3734.83	3874.00	4185.50	1785.0	3.73	3740.10	3795.93	5155.0	0.14	3752.76	3787.72	7796.2	0.48	3755.51	3797.33	7768.9	0.55	**3734.83**	3762.41	3762.9	0.00	-3.59	-0.14	3744.44	3760.24	590.9	0.26	-3.34	0.12
132212	150	10	17	2835.66	2906.00	3129.90	1779.0	2.48	2842.10	2857.43	5839.7	0.23	2837.84	2860.70	8792.6	0.08	2843.18	2857.02	8714.5	0.27	**2835.66**	2850.38	3345.7	0.00	-2.42	-0.08	2844.11	2853.48	580.1	0.30	-2.13	0.22
132222	150	20	17	1651.91	1784.80	2152.30	1773.0	8.04	1660.89	1691.97	6340.4	0.54	1672.86	1697.45	9145.6	1.27	1677.95	1695.15	9288.8	1.58	**1651.91**	1676.40	2824.2	0.00	-7.45	-0.54	1664.08	1678.64	485.9	0.74	-6.76	0.19
133112	150	10	16	4578.87	5034.00	5465.90	1737.0	9.94	4588.37	4619.91	4670.1	0.21	4598.23	4630.69	7313.4	0.42	4596.35	4625.72	7130.4	0.38	**4578.87**	4612.46	3339.0	0.00	-9.04	-0.21	4590.97	4614.54	532.7	0.26	-8.80	0.06
133122	150	20	16	3211.98	3474.00	3849.10	1767.0	8.16	3223.44	3259.45	5317.0	0.36	3225.56	3271.04	8052.6	0.42	3223.40	3248.42	8012.0	0.36	**3211.98**	3236.27	3813.4	0.00	-7.54	-0.35	**3211.98**	3222.46	677.2	0.00	-7.54	-0.35
133212	150	10	17	2903.36	3008.00	3284.50	1770.0	3.60	2911.58	2938.05	5887.3	0.28	2911.35	2938.01	8572.0	0.28	2906.96	2935.00	8754.9	0.12	**2903.36**	2918.06	3116.3	0.00	-3.48	-0.12	2908.51	2917.75	516.9	0.18	-3.31	0.05
133222	150	20	17	2485.07	2617.40	3016.90	1755.0	5.33	2502.97	2550.01	5692.6	0.72	2502.68	2558.34	8274.9	0.71	2501.03	2531.56	8664.8	0.64	**2485.07**	2496.12	3020.2	0.00	-5.06	-0.64	**2485.07**	2497.99	555.3	0.00	-5.06	-0.64
121112	200	10	21	6572.43	7008.70	7371.10	2502.0	6.64	6608.45	6821.20	8392.7	0.55	6621.55	6881.38	13919.9	0.75	6683.24	6848.60	14006.4	1.69	**6572.43**	6632.64	6096.4	0.00	-6.22	-0.55	**6572.43**	6630.65	1038.2	0.00	-6.22	-0.55
121122	200	20	22	5612.03	6039.60	6501.60	2490.0	7.62	5730.53	5954.23	9991.8	2.11	5788.72	5966.38	15672.5	3.15	5784.71	5952.63	16313.2	3.08	**5612.03**	5668.75	5948.1	0.00	-7.08	-2.07	5631.62	5648.04	997.3	0.35	-6.76	-1.73
121212	200	10	21	6409.74	6744.30	7318.60	2559.0	5.22	6503.36	6613.84	8391.2	1.46	6429.62	6608.92	13778.5	0.31	6502.73	6610.46	13910.1	1.45	**6409.74**	6449.92	5906.0	0.00	-4.96	-0.31	6421.16	6446.48	911.5	0.18	-4.79	-0.13
121222	200	20	21	6383.30	6828.50	7567.20	2553.0	6.97	6551.73	6759.22	8391.4	2.64	6562.11	6796.71	14023.1	2.80	6648.20	6776.44	13851.2	4.15	**6383.30**	6526.13	6610.6	0.00	-6.52	-2.57	6480.12	6537.58	1158.5	1.52	-5.10	-1.09
122112	200	10	21	6111.52	6643.80	7106.90	2571.0	8.71	6154.64	6255.10	9463.2	0.71	6184.70	6280.31	18340.7	1.20	6168.32	6268.36	14679.6	0.93	**6111.52**	6208.87	9151.9	0.00	-8.01	-0.70	6167.63	6206.17	1486.9	0.92	-7.17	0.21
122122	200	20	21	3725.07	4012.90	4915.70	2547.0	7.73	3757.37	3782.47	10555.0	0.87	3757.27	3792.68	17314.9	0.86	3744.74	3785.46	16701.0	0.53	**3725.07**	3756.16	4976.8	0.00	-7.17	-0.53	3728.34	3768.28	748.8	0.09	-7.09	-0.44
122212	200	10	21	4025.13	4227.50	4448.10	2535.0	5.03	4046.81	4075.78	9845.3	0.54	4046.42	4078.60	17116.3	0.53	4043.53	4077.42	15490.2	0.46	**4025.13**	4042.78	3871.8	0.00	-4.79	-0.46	4040.88	4045.36	584.4	0.39	-4.41	-0.07
122222	200	20	22	2049.68	2127.90	2345.50	2511.0	3.82	2054.31	2083.56	11266.8	0.23	2052.22	2084.10	18533.2	0.12	2052.16	2079.68	17652.2	0.12	**2049.68**	2056.08	3826.1	0.00	-3.68	-0.12	**2049.68**	2052.86	720.0	0.00	-3.68	-0.12
123112	200	10	22	4868.90	5099.00	5527.40	2559.0	4.73	4916.97	5024.07	10869.1	0.99	4967.11	5047.87	17275.1	2.02	4940.81	5029.64	17259.7	1.48	**4868.90**	4921.23	6253.2	0.00	-4.51	-0.98	4880.27	4917.27	1102.4	0.23	-4.29	-0.75
123122	200	20	22	4675.19	5188.70	5862.90	2544.0	10.98	4725.91	4771.90	10590.0	1.08	4707.61	4785.89	16689.1	0.69	4719.90	4777.07	17021.6	0.96	**4675.19**	4703.17	6099.4	0.00	-9.90	-0.69	4682.46	4690.47	988.0	0.16	-9.76	-0.53
123212	200	10	22	5135.21	5363.00	5678.50	2544.0	4.44	5170.77	5218.43	10206.0	0.69	5178.03	5225.04	17737.1	0.83	5195.48	5247.18	16205.9	1.17	**5135.21**	5174.74	4033.7	0.00	-4.25	-0.69	5143.39	5166.96	619.7	0.16	-4.05	-0.53
123222	200	20	22	2522.89	2657.50	3917.60	2577.0	5.34	2567.20	2629.46	10676.9	1.76	2555.18	2633.86	18065.4	1.28	2553.21	2606.54	17096.2	1.20	**2522.89**	2551.90	3961.3	0.00	-5.07	-1.19	2526.68	2544.00	640.0	0.15	-4.92	-1.04
Global avg.				3799.88	4028.41	4422.78	1861.2	5.56	3835.27	3901.77	5740.9	0.77	3837.85	3909.51	8990.5	0.84	3840.99	3902.19	8934.4	0.86	**3801.30**	3842.98	3715.6	**0.04**	**-5.17**	**-0.51**	3814.16	**3840.57**	**616.8**	0.34	-4.90	-0.22

Table 3.17: Detailed results for the LLRP Prodhon data set

Instance	N_v	BKS	MA				GBILS			SA-VND0				SA-VND1				SA-VND2				M-ILS 30 runs							M-ILS 5 runs						
			Best	Avg	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	gap_{MA}	gap_{GBILS}	gap_{SA}	Best	Avg	time	gap_B	gap_{MA}	gap_{GBILS}	gap_{SA}
20-5-1	5	330.00	337.30	378.00	387.0	2.21	**330.00**	1.0	0.00	**330.00**	**330.00**	125.2	0.00	**330.00**	**330.00**	117.8	0.00	**330.00**	**330.00**	166.4	0.00	**330.00**	**330.00**	167.8	0.00	-2.16	0.00	0.00	**330.00**	**330.00**	30.2	0.00	-2.16	0.00	0.00
20-5-1b	3	608.06	**608.06**	636.80	375.0	0.00	608.06	0.5	0.00	**608.06**	**608.06**	145.7	0.00	**608.06**	**608.06**	122.3	0.00	**608.06**	**608.06**	244.9	0.00	615.66	615.66	113.6	1.25	1.25	1.25	1.25	615.66	615.66	19.7	1.25	1.25	1.25	1.25
20-5-2	5	301.97	304.80	354.40	381.0	0.94	**301.97**	0.7	0.00	**301.97**	**301.97**	106.3	0.00	**301.97**	**301.97**	98.5	0.00	**301.97**	**301.97**	150.5	0.00	**301.97**	**301.97**	161.8	0.00	-0.93	0.00	0.00	**301.97**	**301.97**	27.2	0.00	-0.93	0.00	0.00
20-5-2b	3	486.55	**486.55**	511.20	381.0	0.00	**486.55**	0.8	0.00	**486.55**	**486.55**	158.7	0.00	**486.55**	**486.55**	132.8	0.00	**486.55**	**486.55**	266.6	0.00	**486.55**	**486.55**	114.3	0.00	0.00	0.00	0.00	**486.55**	**486.55**	18.7	0.00	0.00	0.00	0.00
50-5-1	12	843.94	859.90	917.30	546.0	1.89	846.88	153.5	0.35	846.17	849.77	709.2	0.27	846.52	850.10	815.3	0.31	**843.94**	850.40	859.5	0.00	**843.94**	845.80	589.0	0.00	-1.86	-0.35	0.00	**843.94**	845.19	92.3	0.00	-1.86	-0.35	0.00
50-5-1b	6	1293.46	1330.20	1379.80	522.0	2.84	1293.93	65.5	0.04	**1293.46**	1293.71	619.7	0.00	**1293.46**	**1293.54**	602.2	0.00	**1293.46**	1293.55	933.8	0.00	**1293.46**	1293.95	682.8	0.00	-2.76	-0.04	0.00	**1293.46**	**1293.46**	107.5	0.00	-2.76	-0.04	0.00
50-5-2	12	684.13	723.40	786.20	552.0	5.74	691.67	117.4	1.10	**684.13**	694.42	624.0	0.00	**684.13**	692.43	756.3	0.00	**684.13**	694.69	752.8	0.00	**684.13**	690.41	930.0	0.00	-5.43	-1.09	0.00	**684.13**	689.20	164.4	0.00	-5.43	-1.09	0.00
50-5-2b	6	953.25	965.70	1009.40	573.0	1.31	954.88	68.4	0.17	**953.25**	953.50	534.8	0.00	**953.25**	953.35	534.2	0.00	**953.25**	953.33	799.3	0.00	**953.25**	953.67	563.9	0.00	-1.29	-0.17	0.00	**953.25**	953.27	94.9	0.00	-1.29	-0.17	0.00
50-5-2BIS	12	945.45	955.20	981.50	537.0	1.03	952.55	120.7	0.75	949.13	950.77	883.4	0.39	949.57	951.13	1081.1	0.44	950.12	950.93	1024.7	0.49	**945.45**	945.77	1493.0	0.00	-1.02	-0.75	-0.39	**945.45**	946.31	230.3	0.00	-1.02	-0.75	-0.39
50-5-2bBIS	6	803.90	811.80	884.90	534.0	0.98	**803.90**	96.6	0.00	**803.90**	**803.90**	626.9	0.00	**803.90**	**803.90**	649.9	0.00	**803.90**	**803.90**	883.7	0.00	**803.90**	**803.90**	708.2	0.00	-0.97	0.00	0.00	**803.90**	**803.90**	123.5	0.00	-0.97	0.00	0.00
50-5-3	12	831.57	848.10	928.90	612.0	1.99	832.15	119.2	0.07	831.97	835.10	712.3	0.05	833.01	834.91	810.3	0.17	833.59	835.17	863.8	0.24	**831.57**	834.22	1084.5	0.00	-1.95	-0.07	-0.05	**831.57**	833.95	179.3	0.00	-1.95	-0.07	-0.05
50-5-3b	6	1101.57	1163.90	1198.80	531.0	5.66	1106.57	69.4	0.45	**1101.57**	1103.15	538.4	0.00	**1101.57**	1103.95	541.3	0.00	**1101.57**	1103.53	794.5	0.00	**1101.57**	1102.76	467.3	0.00	-5.36	-0.45	0.00	**1101.57**	**1101.57**	80.0	0.00	-5.36	-0.45	0.00
100-5-1	24	2000.80	2030.90	2044.30	891.0	1.50	2035.60	64.8	1.74	2004.33	2023.35	3039.4	0.18	2010.49	2023.78	4791.3	0.48	2008.95	2026.45	3777.3	0.41	**2000.80**	2012.93	2474.4	0.00	-1.48	-1.71	-0.18	2005.30	2013.23	496.7	0.22	-1.26	-1.71	0.05
100-5-1b	12	2311.21	2374.90	2507.80	744.0	2.76	2357.87	102.8	2.02	2311.84	2336.64	2221.6	0.03	2312.53	2337.27	2871.7	0.06	2313.70	2342.80	3126.5	0.11	**2311.21**	2346.56	2164.6	0.00	-2.68	-1.98	-0.03	2316.20	2343.44	383.1	0.22	-2.47	-1.98	0.19
100-5-2	24	1128.43	1226.10	1500.90	852.0	8.66	1144.70	81.1	1.44	1132.36	1135.99	2760.8	0.35	1129.83	1135.49	3976.1	0.12	1131.28	1135.70	3582.7	0.25	**1128.43**	1133.53	2718.1	0.00	-7.97	-1.42	-0.12	1131.17	1133.01	441.3	0.24	-7.74	-1.42	0.12
100-5-2b	11	1507.88	1622.90	1701.00	855.0	7.63	1567.44	96.8	3.95	**1507.88**	1517.11	2530.4	0.00	1510.57	1519.04	3141.3	0.18	1510.57	1519.56	3448.8	0.18	**1507.88**	1512.55	1540.1	0.00	-7.09	-3.80	0.00	1511.06	1514.17	215.0	0.21	-6.89	-3.80	0.21
100-5-3	24	1572.61	1710.40	1726.20	903.0	8.76	1596.77	49.2	1.54	1581.93	1587.20	2784.3	0.59	1581.93	1586.49	4072.3	0.59	1581.93	1587.20	3595.7	0.59	**1572.61**	1581.49	2943.9	0.00	-8.06	-1.51	-0.59	**1572.61**	1581.17	483.5	0.00	-8.06	-1.51	-0.59
100-5-3b	11	1933.00	2054.80	2190.50	870.0	6.30	2032.13	114.9	5.13	1933.70	1950.89	2315.8	0.04	1935.70	1953.85	2932.6	0.14	**1933.00**	1954.97	3247.9	0.00	1934.93	1954.27	2140.7	0.10	-5.83	-4.78	0.10	1934.93	1946.68	374.3	0.10	-5.83	-4.78	0.10
100-10-1	26	1458.80	1524.10	1589.90	1215.0	4.48	1481.56	80.3	1.56	1472.85	1511.00	2994.7	0.96	1470.71	1503.92	4143.4	0.82	1478.01	1510.98	3908.8	1.32	**1458.80**	1465.01	2593.2	0.00	-4.28	-1.54	-0.81	1460.55	1465.33	455.5	0.12	-4.17	-1.54	-0.69
100-10-1b	12	1894.92	1960.70	2138.60	1185.0	3.47	1984.91	100.8	4.75	1901.27	1953.96	2120.3	0.34	1915.77	1972.26	2723.1	1.10	1908.63	1963.53	2999.3	0.72	**1894.92**	1916.90	2211.6	0.00	-3.35	-4.53	-0.33	1911.89	1928.24	372.8	0.90	-2.49	-3.53	0.56
100-10-2	24	1137.59	1175.00	1236.30	1326.0	3.29	1287.50	75.8	13.18	1143.30	1152.81	2899.7	0.50	1142.31	1155.47	4132.2	0.41	1145.93	1155.29	3739.8	0.73	**1137.59**	1145.91	3167.4	0.00	-3.18	-11.64	-0.41	1142.28	1145.11	483.9	0.41	-2.78	-11.64	0.00
100-10-2b	11	1561.40	1625.80	1724.20	1230.0	4.12	1645.07	139.1	5.36	1566.48	1585.67	2208.4	0.33	1566.48	1588.24	2889.5	0.33	1563.99	1578.90	3115.3	0.17	**1561.40**	1570.81	2425.8	0.00	-3.96	-5.09	-0.17	**1561.40**	1565.77	410.8	0.00	-3.96	-5.09	-0.17
100-10-3	25	1204.64	1246.80	1288.30	1332.0	3.50	1216.20	57.2	0.96	1209.20	1221.52	3145.2	0.38	1209.86	1225.98	4290.2	0.43	1211.49	1224.72	4060.6	0.57	**1204.64**	1209.10	1919.8	0.00	-3.38	-0.95	-0.38	1204.95	1208.05	360.6	0.03	-3.36	-0.95	-0.35
100-10-3b	11	1653.83	1799.00	1890.20	1287.0	8.78	1745.05	70.1	5.52	1662.43	1705.63	2146.0	0.52	1665.69	1706.68	2831.1	0.72	1670.17	1707.11	3058.5	0.99	**1653.83**	1673.19	2334.1	0.00	-8.07	-5.23	-0.52	**1653.83**	1669.64	426.0	0.00	-8.07	-5.23	-0.52
200-10-1	49	2780.03	2920.70	3092.40	3414.0	5.06	2861.85	278.2	2.94	2798.58	2854.10	16846.7	0.67	2797.86	2863.27	26673.1	0.64	2803.57	2860.64	23819.4	0.85	**2780.03**	2788.66	6664.8	0.00	-4.82	-2.86	-0.64	2785.75	2790.28	1125.8	0.21	-4.62	-2.86	-0.43
200-10-1b	22	3290.73	3532.20	3809.30	3240.0	7.34	3557.96	304.1	8.12	3368.71	3477.07	11341.6	2.37	3355.70	3478.91	17983.6	1.97	3327.08	3452.84	17334.9	1.10	**3290.73**	3334.98	6272.8	0.00	-6.84	-7.51	-1.09	**3290.73**	3312.41	1090.0	0.00	-6.84	-7.51	-1.09
200-10-2	49	1972.33	2064.20	2153.30	3411.0	4.66	1997.01	342.5	1.25	1984.96	2001.97	17482.5	0.64	1986.55	2004.51	29590.3	0.72	1988.31	2002.48	24959.6	0.81	**1972.33**	1981.41	5560.2	0.00	-4.45	-1.24	-0.64	1979.99	1982.13	954.4	0.39	-4.08	-0.88	-0.25
200-10-2b	23	2325.43	2516.40	2684.50	3141.0	8.21	2473.24	273.7	6.36	2336.11	2379.01	12347.0	0.46	2355.15	2378.21	19720.6	1.28	2368.88	2380.09	18699.6	1.87	**2325.43**	2358.02	5160.5	0.00	-7.59	-5.98	-0.46	2341.77	2353.46	827.8	0.70	-6.94	-5.98	0.24
200-10-3	48	2727.15	2805.90	3066.1	3084.0	2.89	2783.20	323.8	2.06	2741.16	2758.09	14786.4	0.51	2744.67	2757.42	27900.5	0.64	2751.23	2763.79	21623.8	0.88	**2727.15**	2736.49	5479.3	0.00	-2.81	-2.01	-0.51	**2727.15**	2736.07	789.2	0.00	-2.81	-2.01	-0.51
200-10-3b	22	3190.34	3347.00	3454.60	3267.0	4.91	3413.34	281.4	6.99	3242.18	3274.58	9511.1	1.62	3233.89	3267.81	16534.2	1.37	3225.75	3273.68	14933.5	1.11	**3190.34**	3218.10	3924.1	0.00	-4.68	-6.53	-1.10	3208.88	3219.06	604.2	0.58	-4.13	-5.74	-0.52
Global avg.		1494.50	1564.42	1610.33	1272.6	4.03	1546.35	**121.7**	2.59	1502.98	1521.25	3975.5	0.37	1503.92	1522.28	6248.6	0.43	1503.77	1521.76	5692.4	0.45	**1494.82**	1504.82	2292.4	**0.05**	**-3.77**	**-2.40**	**-0.24**	1497.73	**1503.61**	382.1	0.19	-3.63	-2.40	-0.09

Table 3.18: Detailed results for the LLRP Barreto data set

Instance	N_v	BKS	MA				GBILS			SA-VND0				SA-VND1				SA-VND2				M-ILS 30 runs							M-ILS 5 runs						
			Best	Avg	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	gap_{MA}	gap_{GBILS}	gap_{SA}	Best	Avg	time	gap_B	gap_{MA}	gap_{GBILS}	gap_{SA}
Christ-50-5	6	1661.64	1690.80	1782.40	591.0	1.75	–	–	–	**1661.64**	1662.07	541.5	0.00	**1661.64**	1662.13	528.7	0.00	**1661.64**	1662.35	813.7	0.00	**1661.64**	1669.05	614.2	0.00	-1.72	0.00	0.00	**1661.64**	1663.39	107.1	0.00	-1.72	0.00	0.00
Christ-75-10	9	2370.73	2590.30	2689.80	873.0	9.26	–	–	–	2403.79	2459.03	1159.1	1.39	2383.04	2457.45	1384.4	0.52	2408.92	2454.95	1588.4	1.61	**2370.73**	2409.75	1612.4	0.00	-8.48	-0.52	0.00	2391.95	2406.42	284.5	0.90	-7.74	-0.52	0.37
Christ-100-10	8	3791.98	4058.20	4194.90	1023.0	7.02	3984.05	451.9	5.07	**3791.98**	3831.18	1958.7	0.00	3806.39	3838.89	2293.6	0.38	3795.15	3825.37	2876.3	0.08	3803.50	3845.85	2110.8	0.30	-6.28	0.30	0.00	3825.29	3842.92	360.7	0.88	-5.74	0.30	0.88
Gaskell-21-5	4	653.48	658.40	741.10	441.0	0.75	**653.48**	0.9	0.00	**653.48**	**653.48**	116.4	0.00	**653.48**	**653.48**	102.2	0.00	**653.48**	**653.48**	169.5	0.00	**653.48**	**653.48**	143.0	0.00	-0.75	0.00	0.00	**653.48**	**653.48**	23.1	0.00	-0.75	0.00	0.00
Gaskell-29-5	4	1199.33	1224.50	1296.30	468.0	2.10	**1199.33**	5.2	0.00	**1199.33**	**1199.33**	311.0	0.00	**1199.33**	**1199.33**	255.4	0.00	**1199.33**	**1199.33**	485.9	0.00	**1199.33**	**1199.33**	283.2	0.00	-2.06	0.00	0.00	**1199.33**	**1199.33**	46.7	0.00	-2.06	0.00	0.00
Gaskell-32-5b	3	1552.84	1571.00	1668.40	483.0	1.17	**1552.84**	4.9	0.00	**1552.84**	1553.29	417.8	0.00	**1552.84**	1553.29	337.6	0.00	**1552.84**	1553.29	650.7	0.00	**1552.84**	1556.58	284.4	0.00	-1.16	0.00	0.00	**1552.84**	1555.52	43.7	0.00	-1.16	0.00	0.00
Gaskell-36-5	4	1627.17	1642.40	1647.00	522.0	0.94	**1627.17**	3.2	0.00	**1627.17**	**1627.17**	308.3	0.00	**1627.17**	**1627.17**	274.0	0.00	**1627.17**	**1627.17**	465.8	0.00	**1627.17**	1628.12	369.1	0.00	-0.93	0.00	0.00	**1627.17**	**1627.17**	61.0	0.00	-0.93	0.00	0.00
Min-27-5	4	5387.55	**5387.55**	5697.00	834.0	0.00	**5387.55**	84.0	0.00	**5387.55**	**5387.55**	176.3	0.00	**5387.55**	**5387.55**	147.4	0.00	**5387.55**	**5387.55**	267.0	0.00	**5387.55**	**5387.55**	134.0	0.00	0.00	0.00	0.00	**5387.55**	**5387.55**	23.8	0.00	0.00	0.00	0.00
Min-134-8	11	21752.00	23387.00	26012.50	2220.0	7.52	–	–	–	21852.40	22307.28	2225.7	0.46	21910.50	22309.20	2624.0	0.73	21881.80	22278.05	3250.7	0.60	**21752.00**	22442.41	2127.0	0.00	-6.99	-0.46	0.00	21865.00	22297.08	355.4	0.52	-6.62	-0.46	0.06
Or-117-14	7	53798.50	56209.00	61396.20	1545.0	4.48	–	–	–	**53798.50**	54866.72	1263.1	0.00	53859.10	54805.88	1265.5	0.11	**53798.50**	54905.77	1958.4	0.00	54413.90	56794.12	1598.2	1.14	-3.19	1.14	0.00	56228.40	57517.50	273.0	4.52	0.03	4.52	4.52
Global avg.		9379.52	9841.92	10712.56	900.0	3.50	–	–	–	**9392.87**	9554.71	847.8	0.19	9404.10	**9549.44**	921.3	0.17	9396.64	9554.73	1252.6	0.23	9442.21	9758.62	927.6	**0.14**	-4.06	**0.05**	0.58	9639.26	9815.04	**157.9**	0.68	-2.98	0.58	0.58
Global avg GBILS		2368.72	2423.68	2540.78	628.5	2.00	2400.74	**91.7**	0.84	**2368.72**	2375.33	548.1	**0.00**	2371.13	2376.62	568.4	0.06	2369.25	**2374.36**	819.2	0.01	2370.64	2378.48	554.1	0.05	**-3.24**	**0.05**	0.15	2374.28	2377.66	93.1	0.15	-2.98	0.15	0.15

The Barreto data set

This data set considers the less complex instances for the LLRP. Indeed, 6 out of 10 instances of this data set (those for which $N_c \leq 50$) have been solved to proven optimality with the MILP models presented in [16]. The results of this data set are presented in Table 3.18. For the instances Min-134-8 and Or-117-14, no results are reported for GBILS in [16], while for the instances Christ-50-5 and Christ-75-10 the results provided in [16] for GBILS were neglected (as done in [22]), since they correspond to different instances. The last line of Table 3.18 gives the average values (Global avg GBILS) computed by considering only the 6 instances whose values are correctly reported in [16]. Since GBILS obtains the optimal solution value for all the instances reported but for Christ-100-10, the column gap_{GBILS} is not included in the table. According to the results, M-ILS is able to find the proven optimal solution value for 6 instances and, for two instances, new best-known solution values. There are no significant differences concerning the solution quality and the computing times between M-ILS and the three heuristic algorithms presented in [22]. Nevertheless, M-ILS presents the smaller global value for gap_B (0.14%) among all the algorithms.

Although the global computing time required by M-ILS, considering 30 runs, is larger than the one required by GBILS, the average computing times of both algorithms (considering a single run) are similar. Thus, by comparing M-ILS and GBILS, both considering 5 runs, the computing times are equivalent, and both algorithms find the proved optimal solution for 5 instances. Nevertheless, for the instance Christ-100-10 M-ILS provides a better solution value than GBILS with a gap_{GBILS} equal to -3.98%. Finally, regarding MA, M-ILS outperforms it regarding the solution quality in similar computing times. M-ILS provides a better solution value than MA for all the instances but one in which both algorithms provide the same optimal solution value. Furthermore, the average solution value provided by M-ILS is better than the one provided by MA for all the instances. The average value of gap_{MA} equals -3.24% and -2.59% when 30 and 5 runs are considered for M-ILS, respectively. Indeed, M-ILS outperforms MA, even considering 5 runs, in global computing times almost 6 times smaller.

Overall results on the LLRP

After analyzing the results, we can state that the proposed algorithm M-ILS outperforms the state-of-the-art algorithms regarding the solution quality for all the considered data sets. The global average value of gap_B (computed over all the instances) is equal to 0.06% and 0.32% when M-ILS is executed with 30 and 5 runs, respectively. This value is better than the best among all the competitors (0.53% for SA-VND0). Comparing M-ILS to the algorithms proposed in [22], the global average value of gap_{SA} equals -0.33% and -0.07% when M-ILS is executed with 30 and 5 runs, respectively. Considering 30 runs, M-ILS is globally faster than the algorithms SA-VND0, SA-VND1, and SA-VND2 for the two more complex data sets. In contrast, all the algorithms require similar computing times in the third data set (Barreto data set). The average global computing time is 2786.95 s for M-ILS and 4400.23 s for SA-VND0 (the fastest among the state-of-the-art algorithms) when 30 runs are executed for both algorithms. On the other hand, comparing M-ILS to GBILS, the global average value of gap_{GBILS} (computed over all the instances reported for GBILS) is equal to -2.13% and -2.00% when M-ILS is executed with 30, and 5 runs, respectively. Finally, comparing M-ILS to MA, the global average value of gap_{MA} (computed over all the instances reported for MA) equals -4.35% and -4.07% when M-ILS is executed with 30 and 5 runs, respectively.

It is to note that for this problem we do not present the computational experiments regard-

ing the comparison of the computing times required by the considering algorithms to reach the target values. The reason is that most of the values BKS_0 were provided for one of the three metaheuristics presented in [22], and since M-ILS outperforms these algorithms, finding better solution values within shorter computing times, considering a time limit larger than the computing time reported for M-ILS would lead to redundant conclusions. The same situation applies if a target value worse than the best solution value obtained by M-ILS is considered. Furthermore, for the instances in the Barreto data set, M-ILS (considering 5 runs) was able to find the same solution values as those obtained by GBILS (the fastest among the heuristics) in equivalent computing times for all the instances but one, in which M-ILS finds a better solution.

3.4 Conclusions and future research

An effective metaheuristic (M-ILS) is proposed for solving the MDCCVRP, the MDk-TRP, and the LLRP. The algorithm was tested on several benchmark data sets, with a total of 78 instances for the MDCCVRP, 87 instances for the MDk-TRP, and 76 instances for the LLRP. Extensive computational experiments show that M-ILS outperforms in terms of solution quality the state-of-the-art metaheuristic algorithms PLS (proposed in [13] for the MDCCVRP), GA (proposed in [10] for the MDk-TRP), and SA-VND0, SA-VND1, and SA-VND2 (proposed in [22] for the LLRP), with competitive computing times.

The experiments also show that the stability of the proposed metaheuristic allows for a reduction of the number of runs necessary to provide good-quality solutions, implying global computing times shorter than those required by the currently published heuristic methods. Indeed, when M-ILS is executed with a time limit equal to that required by PLS and GA for the MDCCVRP and the MDk-TRP, respectively, M-ILS is able to find solution values better than those obtained by the mentioned competitors. For the LLRP, M-ILS requires shorter computing times and finds better quality solutions than those corresponding to the algorithms SA-VND0, SA-VND1, and SA-VND2, considering the same number of runs (30). Indeed, M-ILS outperforms the mentioned algorithms even when the number of runs considered is much smaller (5).

According to the results reported in Section 3.3, the proposed metaheuristic is globally the most effective algorithm for the considered problems when challenging instances, in which the number of customers is large, and the number of vehicles is relatively small, must be solved. In addition, the proposed algorithm provides optimal or near-optimal solution values for the easiest instances.

Based on the obtained results, it is possible to suggest the application of the proposed methodology to other related problems as future research directions. Some examples could be extensions of single-depot latency vehicle routing problems studied in the literature, for example, including time windows (generalizing the problem studied in [36]), considering priorities for the customers (generalizing the problem studied in [37], or combining truck and drones in last-mile delivery operations (generalizing the problem studied in [38]). Also, since M-ILS can provide good quality solutions for large-size instances within short computing times compared to those required by the exact methods, M-ILS could be useful to solve real-life problems related to humanitarian logistics ([37] and [23]) or other applications. Another interesting variant of the problem (especially suitable for post-disaster management) could consider pickup and delivery decisions, where some vital products, such as water and food, must be delivered, while injured people must be picked-up.

Appendix: Optimal solution for the MDk-TRP instances with $N_v = 35$

Instance p01

Route 1	**Route 6**	**Route 11**	**Route 16**	**Route 21**	**Route 26**	**Route 31**
D1 - 13	D3 - 38	D3 - 16	D1 - 15	D3 - 30	D1 - 41	D1 - 17 - 37
Route 2	**Route 7**	**Route 12**	**Route 17**	**Route 22**	**Route 27**	**Route 32**
D4 - 21	D2 - 1	D2 - 23 - 43	D1 - 4 - 18	D4 - 22	D1 - 42	D3 - 49 - 33
Route 3	**Route 8**	**Route 13**	**Route 18**	**Route 23**	**Route 28**	**Route 33**
D4 - 35 - 36	D1 - 25	D3 - 10	D3 - 39	D2 - 48 - 7	D2 - 6 - 24	D2 - 46 - 11
Route 4	**Route 9**	**Route 14**	**Route 19**	**Route 24**	**Route 29**	**Route 34**
D2 - 14	D2 - 26	D2 - 12	D2 - 32	D3 - 5	D1 - 44 - 45	D3 - 34
Route 5	**Route 10**	**Route 15**	**Route 20**	**Route 25**	**Route 30**	**Route 35**
D4 - 29 - 2	D4 - 20 - 3	D2 - 27 - 8	D2 - 47	D3 - 9 - 50	D4 - 28 - 31	D1 - 19 - 40

Instance p12

Route 1	**Route 8**	**Route 15**	**Route 22**	**Route 29**
D1 - 26	D1 - 5 - 13	D1 - 28 - 36	D2 - 42 - 50 - 58 - 66 - 74	D1 - 14
Route 2	**Route 9**	**Route 16**	**Route 23**	**Route 30**
D1 - 2 - 18	D2 - 44 - 52 - 60 - 68 - 76	D1 - 6 - 22 - 30	D2 - 45 - 53 - 61 - 69 - 77	D1 - 38
Route 3	**Route 10**	**Route 17**	**Route 24**	**Route 31**
D1 - 10	D1 - 9	D1 - 34	D1 - 39	D1 - 27
Route 4	**Route 11**	**Route 18**	**Route 25**	**Route 32**
D1 - 4 - 12 - 20	D2 - 46 - 54 - 62 - 70 - 78	D1 - 3	D2 - 51 - 67	D2 - 47
Route 5	**Route 12**	**Route 19**	**Route 26**	**Route 33**
D2 - 65	D1 - 7 - 15 - 23 - 31	D2 - 41 - 49 - 57 - 73	D1 - 11 - 19 - 35	D2 - 71 - 79
Route 6	**Route 13**	**Route 20**	**Route 27**	**Route 34**
D2 - 48 - 56 - 64 - 72 - 80	D1 - 40	D1 - 8 - 16 - 24 - 32	D2 - 59 - 75	D2 - 55 - 63
Route 7	**Route 14**	**Route 21**	**Route 28**	**Route 35**
D1 - 1	D1 - 17 - 33	D1 - 25	D1 - 21 - 29 - 37	D2 - 43

Bibliography

[1] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez, and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," Computers & Industrial Engineering, vol. 79, pp. 115–129, 2015.

[2] J. F. Sze, S. Salhi, and N. Wassan, "The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search," Transportation Research Part B: Methodological, vol. 101, pp. 162–184, 2017.

[3] S. U. Ngueveu, C. Prins, and R. Wolfler Calvo, "An effective memetic algorithm for the cumulative capacitated vehicle routing problem," Computers & Operations Research, vol. 37, no. 11, pp. 1877–1885, 2010.

[4] K. Corona-Gutiérrez, S. Nucamendi-Guillén, and E. Lalla-Ruiz, "Vehicle routing with cumulative objectives: a state of the art and analysis," Computers & Industrial Engineering, p. 108054, 2022.

[5] J. N. Tsitsiklis, "Special cases of traveling salesman and repairman problems with time windows," Networks, vol. 22, pp. 263–282, may 1992.

[6] M. Fischetti, G. Laporte, and S. Martello, "Delivery man problem and cumulative matroids," Operations Research, vol. 41, pp. 1055–1064, dec 1993.

[7] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, "The minimum latency problem," in Proceedings of the Annual ACM Symposium on Theory of Computing, vol. Part F129502, pp. 163–171, Association for Computing Machinery, may 1994.

[8] J. Fakcharoenphol, C. Harrelson, and S. Rao, "The k-traveling repairmen problem," ACM Transactions on Algorithms (TALG), vol. 3, no. 4, pp. 40–es, 2007.

[9] F. Ángel-Bello, Y. Cardona-Valdés, and A. Álvarez, "Mixed integer formulations for the multiple minimum latency problem," Operational Research, vol. 19, no. 2, pp. 369–398, 2019.

[10] M. E. Bruni, S. Khodaparasti, I. Martínez-Salazar, and S. Nucamendi-Guillén, "The multi-depot k-traveling repairman problem," Optimization Letters, vol. 16, pp. 2681–2709, 2022.

[11] M. Moshref-Javadi and S. Lee, "The latency location-routing problem," European Journal of Operational Research, vol. 255, no. 2, pp. 604–619, 2016.

[12] E. Lalla-Ruiz and S. Voß, "A POPMUSIC approach for the Multi-Depot Cumulative Capacitated Vehicle Routing Problem," Optimization Letters, vol. 14, no. 3, pp. 671–691, 2020.

[13] X. Wang, T.-M. Choi, Z. Li, and S. Shao, "An Effective Local Search Algorithm for the Multidepot Cumulative Capacitated Vehicle Routing Problem," IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp. 1–11, 2020.

[14] G. Mattos Ribeiro and G. Laporte, "An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem," Computers & Operations Research, vol. 39, no. 3, pp. 728–735, 2012.

[15] C. M. Damião, J. M. P. Silva, and E. Uchoa, "A branch-cut-and-price algorithm for the cumulative capacitated vehicle routing problem," 4OR, pp. 1–25, 2021.

[16] S. Nucamendi-Guillén, I. Martínez-Salazar, S. Khodaparasti, and M. E. Bruni, "New formulations and solution approaches for the latency location routing problem," Computers & Operations Research, vol. 143, p. 105767, 2022.

[17] X. Wang, T.-M. Choi, H. Liu, and X. Yue, "A novel hybrid ant colony optimization algorithm for emergency transportation problems during post-disaster scenarios," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 4, pp. 545–556, 2016.

[18] İ. Kara, B. Y. Kara, and M. K. Yetiş, "Cumulative vehicle routing problems," in Vehicle Routing Problem, pp. 85–98, In-Teh, 2008.

[19] F. Angel-Bello, A. Alvarez, and I. García, "Two improved formulations for the minimum latency problem," Applied Mathematical Modelling, vol. 37, no. 4, pp. 2257–2266, 2013.

[20] S. Nucamendi-Guillén, I. Martínez-Salazar, F. Angel-Bello, and J. M. Moreno-Vega, "A mixed integer formulation and an efficient metaheuristic procedure for the k-travelling repairmen problem," Journal of the Operational Research Society, vol. 67, no. 8, pp. 1121–1134, 2016.

[21] S. Nucamendi-Guillén, F. Angel-Bello, I. Martínez-Salazar, and A. E. Cordero-Franco, "The cumulative capacitated vehicle routing problem: New formulations and iterated greedy algorithms," Expert Systems with Applications, vol. 113, pp. 315–327, 2018.

[22] A. Osorio-Mora, C. Rey, P. Toth, and D. Vigo, "Effective metaheuristics for the latency location routing problem," International Transactions in Operational Research, 2023.

[23] M. Ajam, V. Akbari, and F. S. Salman, "Routing multiple work teams to minimize latency in post-disaster road network restoration," European Journal of Operational Research, vol. 300, no. 1, pp. 237–254, 2022.

[24] S. Zhong, R. Cheng, Y. Jiang, Z. Wang, A. Larsen, and O. A. Nielsen, "Risk-averse optimization of disaster relief facility location and vehicle routing under stochastic demand," Transportation Research Part E: Logistics and Transportation Review, vol. 141, p. 102015, 2020.

[25] M. E. Bruni and S. Khodaparasti, "A variable neighborhood descent matheuristic for the drone routing problem with beehives sharing," Sustainability, vol. 14, no. 16, p. 9978, 2022.

[26] K. Helsgaun, "An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems," Roskilde University, 2017.

[27] S. Martello and P. Toth, "Bin-packing problem," in Knapsack problems: algorithms and computer implementations, pp. 221–245, John Wiley & Sons, Inc., 1990.

[28] J. W. Escobar, R. Linfati, and P. Toth, "A two-phase hybrid heuristic algorithm for the capacitated location-routing problem," Computers & Operations Research, vol. 40, no. 1, pp. 70–79, 2013.

[29] J. W. Escobar, R. Linfati, P. Toth, and M. G. Baldoquin, "A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem," Journal of heuristics, vol. 20, no. 5, pp. 483–509, 2014.

[30] J. W. Escobar, R. Linfati, M. G. Baldoquin, and P. Toth, "A granular variable tabu neighborhood search for the capacitated location-routing problem," Transportation Research Part B: Methodological, vol. 67, pp. 344–356, 2014.

[31] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," in Handbook of metaheuristics, pp. 129–168, Springer, 2019.

[32] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," Operations research, vol. 21, no. 2, pp. 498–516, 1973.

[33] IBM, "IBM ILOG CPLEX 20.1 user's manual," 2021.

[34] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," Operations Research Perspectives, vol. 3, pp. 43–58, 2016.

[35] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021.

[36] N. A. Kyriakakis, I. Sevastopoulos, M. Marinaki, and Y. Marinakis, "A hybrid tabu search–variable neighborhood descent algorithm for the cumulative capacitated vehicle routing problem with time windows in humanitarian applications," Computers & Industrial Engineering, vol. 164, p. 107868, 2022.

[37] M. Bruni, S. Khodaparasti, and P. Beraldi, "The selective minimum latency problem under travel time variability: An application to post-disaster assessment operations," Omega, vol. 92, p. 102154, 2020.

[38] M. Bruni, S. Khodaparasti, and M. Moshref-Javadi, "A logic-based benders decomposition method for the multi-trip traveling repairman problem with drones," Computers & Operations Research, vol. 145, p. 105845, 2022.

4 A risk-averse latency location-routing problem with stochastic travel times

Abstract

In this chapter, a latency location routing problem with stochastic travel times (LLRP-STT) is investigated. The problem is modelled as a two-stage stochastic program, by setting the location decisions at the first stage, and the routing decisions at the second one. A risk-averse decision maker is assumed. An efficient and effective multi-start variable neighborhood search algorithm is proposed for tackling the problem. Furthermore, a sampling method is presented for solving instances with continuous probability distribution. Finally, several insights are provided.

Keywords: LLRP, cumulative routing, uncertainty, variable neighborhood search, simheuristic

4.1 Introduction

The location-routing problem (LRP) is a combination of two well-known combinatorial optimization problems: *i*) the facility location problem (FLP), and *ii*) the vehicle routing problem (VRP). Several variants of the problem have been studied in the literature, being the capacitated location routing problem (CLRP) the one that attracted more attention. In this paper we focus on a relatively new variant of the LRP: the latency location routing problem (LLRP). Given a set of customers to be served and a set of potential depots, the problem consists of determining the subset of depots to open, the customers and the vehicles to be assigned to each open depot and the routes to be constructed to fulfill the demand of the customers. The objective is to minimize the sum of the arrival times at the customers, i.e., the latency. The following assumptions are considered for the LLRP:

- there is a fleet of homogeneous capacitated vehicles;
- there is a set of potential uncapacitated depots;
- at most p depots can be opened;
- each customer (demand node) has a non-negative demand that must be fulfilled exactly once by a single vehicle;
- due to the nature of the objective function, the routes are considered "open", e.g., each vehicle finishes its route after visiting its last customer;
- the sum of the demands of the customers visited by each route cannot exceed the vehicle capacity.

This chapter is based on the paper: Osorio-Mora A, Saldanha-da-Gama. F, Toth P. A risk-averse latency location-routing problem with stochastic travel times. submitted to European Journal of Operational Research. 2023 Oct.

The chapter is organized as follows. An in-deep literature review is presented in Section 4.2. Section 4.3 presents a formal description of the problem and a mathematical formulation. The proposed solution method is described in Section 4.4. The computational experiments are described, and the respective results are reported and analyzed in Section 4.5. Finally, in Section 4.6, a summary of our findings and future directions are drawn.

It is to note that no survey paper has been published addressing location routing problems under uncertainty. Consequently, Section 4.2.1 can be consider a contribution itself since a detailed overview of the area is given. There, the interested reader can easily find potential research avenues.

4.2 Literature review

LRPs have received much attention in the past years. However, there is still a gap in the literature when it comes to hedging against uncertainty in such problems. In the same way, latency routing problems started to receive more attention in the past 10 years, due to their importance when post-disaster operations and other customer-centric problems have to be faced. Nevertheless, there is still a gap regarding the inclusion of uncertainty in this kind of problems. Next, we review the related literature both for LRPs and latency routing problems under uncertainty. For comprehensive surveys on the LRPs, the reader should refer to [1], [2], [3], [4] and [5] (in chronological order). For an overview on FLPs under uncertainty the reader can refer to [6]. For a review on VRPs under uncertainty, the interested reader can refer to [7], and [8, 9]. Finally, for a recent survey on latency vehicle routing problems the reader can refer to [10].

4.2.1 Location routing problems under uncertainty

To the best of the authors' knowledge, the first LRP considering uncertainty was studied by [11]. The authors cast the problem as a two-stage stochastic programming problem. Uncertainty stems from the demand. The first stage consists of deciding the location of the facilities and the routes for the vehicles. The operating cost of the depots, the cost for using the vehicles, and the transportation cost of the planned routes are accounted for at this stage. In the second stage, a recourse action is performed if the capacity of the vehicles is violated: the vehicle must return to the depot to empty its load and then resume the planned route from the customer in which the capacity was, or would be, exceeded. These violations have a penalty associated. Two mathematical models are proposed seeking to minimize the first-stage costs and an additional constraint induced by the second stage decision: *i*) the probability of route failure does not exceed a given threshold, and *ii*) the expected penalty of any route does not exceed a fraction of its planned cost. A branch-and-bound algorithm (B&B) was proposed for solving the problem. Gaussian demand are assumed.

A two-stage stochastic LRP using uncapacitated vehicles can be found in [12]. A major distinguishing feature is that the authors assume that if the actual demand of the customers served by a depot exceeds its capacity, then some customers (randomly chosen) are not served and a penalty is incurred. The authors assume that the second-stage routes follow the same sequence of customers decided *a priori* but skipping the selected customers. The objective is to minimize the sum of the fixed costs for opening the depots plus the total expected penalty. The demand of the customers is assumed to follow a Bernoulli distribution. This calls for approximating the recourse function since its exact value can hardly be assessed. A heuristic algorithm is proposed: *i*) a constructive phase is considered in which the location, the allocation,

and the routes are decided using different exact and heuristic procedures, and *ii*) an improvement phase is performed that is based on local search.

[13] and [14] address the CLRP with stochastic demands, casting it also as a two-stage stochastic programming problem. In both works, the authors consider a "safety stock" in the vehicles, i.e., when a given proportion of the capacity of the vehicle is already used, the vehicle returns to the depot to fully unload/replenish, and then the route is resumed. A particle swarm optimization (PSO) heuristic is proposed in the first work while a simheuristic algorithm is considered in the second. The demands are assumed to follow a Poisson and log-Normal distribution, respectively.

[15] assume fuzzy demands. The authors seek to minimize the total cost of the system, which includes the fixed cost for opening depots and using vehicles as well as the variable costs of the routes. A fuzzy chance-constrained programming (FCCP) model is derived. Nevertheless, due to the difficulty in tackling the model using a general-purpose solver, the authors consider a relaxation for providing a lower bound on the optimal value. For the whole problem, a heuristic algorithm is proposed. [16] study a multi-period version of the above problem. Location decisions must be made in the first period and should remain fixed throughout the planning horizon, while the routing decisions can change according to the actual demands observed along the different periods. A penalty is assumed for not serving customers, which is included in the objective function. An FCCP model is derived and tackled using an off-the-shelf solver, which, nonetheless is not efficient for large-scale instances. Thus, a heuristic procedure similar to that devised by [15] is proposed.

A CLRP with time windows (CLRP-TW) under uncertainty in demands and travel times is investigated by [17]. The problem involves a fleet of homogeneous capacitated vehicles and a set of heterogeneous capacitated depots. Uncertainty is captured by means of fuzzy numbers. The objective function accounts for the total cost associated with the location of the depots plus the routing costs and the total additional travel distance due to route failures. The authors introduce a FCCP model and propose a simulated annealing (SA) algorithm for tackling the problem. A similar problem is analyzed by [18]. Nevertheless, now only the demands are assumed to be fuzzy. The objective function seeks to minimize the total cost that results from adding the transportation costs, the fixed costs for using depots and vehicles, and the penalties for violating the time windows of the customers. The problem is tackled using a variable neighborhood search algorithm (VNS) which was hybridized with an evolutionary algorithm.

A bi-objective stochastic demand CLRP with time windows and a heterogeneous fleet is investigated by [19] considering a risk-averse decision maker The objectives to minimize are the conditional value-at-risk (CVaR), using the regret as the loss function, both for the total cost and the latency. The problem, is cast as a two-stage stochastic programming problem. Not only are penalties assumed for not fully supplying customers but also for over-supplying them. The authors assume that the demands of the customers lie in a given interval. The amount to supply becomes a decision to make. The authors introduce a metaheuristic that consists of solving a single-objective LRP using a genetic algorithm (GA) followed by an approximation of the Pareto frontier obtained using a non-dominating sorting algorithm type II (NSGA-II).

A LRP subject to infrastructure disruptions is called a reliable location-routing problem (RLRP). Such a problem is investigated by [20], who considered disruptions both at the depots and also in the vehicles. The former corresponds to random reductions in the capacities, while the latter imply that some routes cannot be operated. The objective is to minimize the total cost of the system, which includes the fixed costs for opening depots, the transportation costs for visiting customers, and the disruptions costs. The authors derive mixed-integer-linear pro-

gramming (MILP) models for three scenarios: moderate, cautious, and pessimistic. The models are solved using a general-purpose solver for small- to medium-sized instances. For large-scale ones a heuristic algorithm based on simulated annealing is proposed. For the computation tests the authors assume that the disruptions at the depots and those related with the vehicles are well described by uniform and binomial probability distributions, respectively.

[21] studied a RLRP considering disruptions only at the depots. In particular, it is assumed that when a depot is disrupted it cannot be used. A two-stage decision making process is assumed. The goal is to decide the depots to open, the routes to consider for a fleet of homogeneous capacitated vehicles, and the allocation of the routes to depots in the two stages. In particular, if a route is allocated to a certain depot in the first stage and the depot turns out to be disrupted, then a backup operation must be performed. This is accomplished by re-allocating the route to another depot without changing the sequence in which the customers are visited. This incurs a cost higher than that for the original planed routes. Furthermore, if all the backup depots for a certain route are disrupted, then a cost is assumed for not serving the customers. The authors also assume independence in terms failure of the depots with equal failure probabilities. An integer programming (IP) model is derived that is solved using an algorithm combining Lagrangian relaxation with column generation.

A similar problem was analyzed by [22]: a two-stage stochastic RLRP is investigated considering that the depots to open and the planned routes are ex ante decisions (before the failures occur) and the final routing decisions are ex post decisions (considering the non-disrupted opened facilities). Unlike the problem studied by [21], it is now possible to adapt routes as a result of the observed disruptions. A two-stage stochastic mixed-integer programming (MIP) model is proposed. The complexity of the problem calls for using an approximate procedure. A sampling technique is the choice made. It can be looked at as a particular case of the sample average approximation method [23]. In particular, the authors impose that only those scenarios with the highest probability of occurrence are chosen. In the computational tests presented, the disruption risk is assumed to be uniformly distributed in a given set. Only small instances can be solved using an off-the-shelf solver. For this reason, a matheuristic combining different heuristics and exact procedures is proposed for handling large-sized instances.

A robust LRP with time windows considering electric vehicles is studied by [24]. The problem consists of deciding the location of a set of recharging stations as well as the routes of the vehicles considering linear recharging and consumption rates. Uncertainty is assumed for the demands, time windows and service times of the customers. The robustness method used by the authors is the adversarial approach. A parallelized adaptive large neighborhood search (ALNS) algorithm is developed for finding feasible solutions to the problem.

More recently, [25] consider a LRP for electric vehicles with stochastic demands. The facilities to locate correspond to battery-swap stations. In addition to the location decisions, the first stage also considers the priori routes. The second stage decisions correspond to the actual routes, which may include a replenishment trip to the depot if the observed demand exceeds the capacity of the vehicle. An algorithm hybridizing VNS and PSO is proposed. In the instances tested computationally the demands of the customers are uniformly distributed in a three-value discrete set.

A robust optimization approach for the drone latency location routing problem (DLLRP) with uncertain travel times is introduced by [26]. The problem is to decide the location of fulfillment centers (depots), where the drones can recharge their battery, and the route that the drones must follow, considering not only the visits to the customers but also the recharging stops at the depots. Due to the source of uncertainty, the energy consumption constraints may be violated.

A branch & check algorithm is proposed for solving the robust problem, and a case study is also presented.

A stochastic CLRP with pickup and deliveries for multiple products is investigated by [27]. Both the depots and the vehicles have specific capacities for each product. The authors considered stochastic demands following a Poisson distribution. The problem is modeled as a two-stage stochastic programming problem. In the first stage, the location, the number of vehicles and the routes are decided; in the second stage a recourse action considers a penalty for exceeding the capacities of the vehicles and of the depots. A tabu search (TS) heuristic is developed for approximating the optimal solution.

A different type of LRP under uncertainty is investigated by [28]. The problem captures features such as a maximum length acceptable for the routes and split deliveries (i.e. the customers can be visited more than once by a vehicle). The stochastic demands are captured using a queuing network at each customer. The problem is tackled using a heuristic that combines the three-dimensional space-filling curve algorithm and an extended Clarke–Wright procedure.

In the literature, we can also observe some work that embeds LRP under uncertainty within network design problems. This is the case with multi-echelon LRPs, which have much practical relevance namely, in Logistics and Supply Chain Management. The location-routing-inventory problem (LRIP) under uncertainty has also attracted the attention of the scientific community in the past years. Next, we briefly refer to these cases.

A bi-objective multi-echelon LRP with time windows is studied by [29]. Uncertainty stems from demands and service times. The problem was inspired by the SARS-CoV-2 (COVID-19) pandemic, where hazardous materials have to be managed. This calls for two types of facilities to be located: treatment centers (that can be partially or permanently open), and temporary transfer stations. The authors considered that large size customers (e.g. hospitals) can be directly allocated to the treatment centers, while small customers (e.g. laboratories) must be served as part of a route collecting the hazardous materials at the transfer stations before sending them to the treatment centers. The objective is to minimize the total cost as well as the risk of infection to which the populations close to open facilities are exposed. Three future scenarios are considered and a variant of the $\varepsilon-$constrained method with a branch-and-price algorithm is proposed for determining the Pareto front.

The two-echelon multi-period CLRP with stochastic demands is investigated by [30]. The problem is an extension of the CLRP in which an upper level facility must also be located (in addition to the classical location-routing decisions). The authors develop a two-stage stochastic programming model considering the location and capacity decisions at the first stage, and the routing decisions at the second stage. The demands are assumed to be captured by a Gaussian distribution. A Benders decomposition algorithm is proposed.

[31] investigate a stochastic LRP in the context of disaster management. The demand in the affected areas, the number of vehicles available (and its respective medical personnel), and the state of the infrastructure are the three main uncertainty sources considered. The problem is addressed within a three-stage stochastic programming modeling framework with some information being revealed at each stage. Initially, the location of the distribution centers, the amount of aid available at them, and the number of vehicles available at each opened distribution center must be decided. In the second stage, after the demand is revealed, the routes of the vehicles are decided. Finally in the third stage, after information about infrastructure conditions is revealed, a recourse action is performed for the routes that cannot be used. The goal is to maximize the total utility, which is mainly related to the demand fulfilled, considering the urgency associated

with each affected area. The proposed model is solved using a commercial solver and it is tested using several instances, including real-life-based ones.

[32] deal with a multi-modal network design problem under uncertainty. Again, a two-stage decision making process is assumed. The first phase seeks to determine how to send the products from the supplier to the distribution centers (depots for the second phase), being able to combine transportation modes. For performing a multi-modal shipment it is necessary to locate intermodal changing points. The second phase corresponds to a LRP. Time windows are considered. Uncertainty in demand is captured using fuzzy numbers. The goal is to minimize the total cost of the system. A fuzzy MIP model is derived that is solved (for small instances) using a commercial solver. The complexity of the problem also motivates the development of a genetic algorithm seeking feasible solutions.

[33] present a multi-objective humanitarian logistics network design problem under uncertainty. Three different types of facilities must be located, and the routes for the delivery of relief items must also be decided. The goal is to minimize (*i*) the total cost of the system, (*ii*) the total time of the relief operations, and (*iii*) a workload balance objective. Several sources of uncertainty are considered: the capacities, the demands and the number of people injured. Uncertainty sets are assumed for those unknowns. Regarding the costs and the transportation times, they are captured using fuzzy numbers. Altogether, a so-called robust neutrosophic model is derived which is tackled using a commercial solver.

A CLRP considering stochasticity both in the demands and in the infrastructure operationality (affecting the transportation times) is investigated by [34]. The authors were inspired by a humanitarian logistics application. A two-stage decision making process is assumed. In the first stage, the location and the inventory associated with the depots must be decided. The second-stage decision regards the routes and flow transshipment between the depots. The objective is to minimize the weighted setup cost of the depots (first stage) plus the worst-case response time (second stage). The second-stage "cost" also accounts for a weighted penalty for not serving the customers. The authors propose a linearization for their model and tackle it using an off-the-shelf solver.

The location-routing-inventory problem (LRIP) is an extension of the LRP in which inventory decisions are also to be made. Usually, the inventory decisions lead to non-linear cost objective functions. Furthermore, most of the LRIPs are presented as multi-echelon network design problems in which different types of facilities must be located (e.g. suppliers and distribution centers). [35] are pioneers in this context. The authors consider Gaussian demands and economies of scale for the transportation and inventory costs. In order to simplify the problem, approximation costs are derived for the routing part. A MINLP formulation is derived that is tackled using a Lagrangian-relaxation-based algorithm.

A LRIP with stochastic demands seeking for minimizing the total cost of the system is presented in [36]. The authors assume a Gaussian distribution for the demands of the customers. In addition to the location, routing, and inventory decisions, the capacity of the open depots must also be decided. The problem is addressed through a mixed-integer-convex programming (MICP) model solved with a commercial solver. A metaheuristic combining tabu search and simulated annealing is also proposed.

A similar problem was studied by [37] where a bi-objective multi-product multi-period LRIP with stochastic demands is investigated. Again, Gaussian demands are assumed. The objective functions to minimize are the total cost and the maximum time for delivering commodities to customers. A bi-objective MINLP model is proposed which is solved (for small instances) with the ε-constrained method. For dealing with large-scale instances four metaheuristic algo-

rithms are proposed out of which the so-called multi-objective imperialist competitive algorithm (MOICA) turned out to be the most effective.

A multi-objective closed-loop supply chain network design problem under uncertainty is studied by [38]. This problem can be looked at as a special case of the multi-period LRIP in which additional decisions are to be made. The distribution centers to open must be decided together with the decision about which ones will also be used as re-manufacturing facilities. The capacities of these facilities is also a decision to make. The routing is only performed for the delivery case from the distribution centers to the retailers (customers), while the products to be re-manufactured are sent directly from the retailers to the re manufacturing centers. Uncertainty is associated both with the demand to deliver and the amount to be re-manufactured. Such uncertainty is assumed to be described by a Gaussian distribution. the authors seek to optimize economic, environmental and social objectives. For solving the problem, a hybrid two-stage approach was proposed that, nonetheless, is able to solve only small and medium size instances. For larger ones a self-adaptive genetic algorithm metaheuristic is proposed.

A multi-objective stochastic LRIP for hazardous waste management is presented in [39]. The objectives to minimize are the total cost as well as the environmental risks associated with opening facilities and hazmat transportation. The authors consider uncertainty both in the demands and in the risk parameters. The problem is tackled using a multi-objective simheuristic.

[40] investigate a multi-period multi-depot four-echelon supply chain network design problem for perishable products assuming fuzzy demands. In addition to location, routing and inventory, the capacity of the facilities is also a decision to make. A MILP model is derived. Small instances are tackled by a general-purpose solver. For larger instances a hybrid metaheuristic combining a GA and PSO is proposed.

A robust multi-period LRIP in the context of healthcare logistics is studied by [41]. The problem consists of deciding the location of warehouses, the inventory level at those facilities, and the routes from warehouses to the hospitals. Demand for multiple products is assumed to be uncertain. Direct shipment from the supplier, (thus not being part of a route) is allowed when the inventory level is not enough for satisfying the demand. Nevertheless, in such a case, a high cost is incurred. A robust MILP model is derived whose objective function accounts for the total cost of the system. A general solver is used for tackling the model. The robust optimization setting adopted in this work follows the methodology proposed in [42].

A multi-objective LRIP with stochastic demands in the context of agricultural biofuel supply chain is studied by [43]. The problem is modeled as a multi-echelon supply chain featuring multiple periods and multiple products. In addition to the location, routing and inventory decisions, capacity expansions, and the possibility to open/close facilities in different periods are decisions to make. The authors develop a two-phase heuristic algorithm based on simulated annealing.

[44] presented a multi-objective LRIP with fuzzy demands applied to perishable multi-product materials. The three objectives to minimize are the total cost, the latency, and the pollution. A mathematical model, able to solve small-size instances through the $\varepsilon-$constrained method, is derived. Furthermore, two genetic-algorithm-based metaheuristics are proposed to tackle more complex instances for the problem.

Table 4.1: Literature review: LRPs under uncertainty and related problems.

Problem	Authors	Type	Uncertainty source	Objective	Solution method
LRP	[11]	S	d	C	E
	[12]	S	d	C	H
	[13]	S	d	C	H
	[14]	S	d	C	H
	[15]	F	d	C	E,H
	[16]	F	d	C	E,H
	[17]	F	d,t	C,T	E,H
	[18]	F	d	C	H
	[19]	S	d	C,T	H
	[20]	S	O	C	E,H
	[21]	S	O	C	E
	[22]	S	O	C	H
	[24]	R	d,O	C	E,H
	[25]	S	d	C	H
	[26]	R	t	T	E
	[27]	S	d	C	H
	[28]	S	d	C	H
ME+LRP	[29]	S	d,t	C,O	E
	[30]	S	d	C	E
	[31]	S	d,q,c,t,O	O	E
	[32]	F	d	C	E,H
	[33]	R,F	d,q,c,t	C,T,O	E
	[34]	S	d,t	C,T	E
LRIP	[35]	S	d	C	E
	[36]	S	d	C	E,H
	[37]	S	d	C,T	E,H
	[38]	S	d	C,O	H
	[39]	S	d,O	C,O	H
	[40]	F	d	C,O	E,H
	[41]	R	d	C	E
	[43]	S	d	O	E,H
	[44]	F	d	C,T,O	E,H
LTP	[45]	S	d	O	H

S: stochastic, R: robust, F: fuzzy
d: demand, q: capacity, t: time, c: costs, O: other
C: cost-based, T: time-based, O: other
E: exact, H: heuristic

The location-transportation problem (LTP) is a generalization of the LRP in which several transportation modes are available A multi-period LTP with stochastic demands is presented by [45]. The problem is cast as a two-stage stochastic problem with recourse in which the objective is to maximize the expected profit. The authors consider the location of the depots as the first-stage decisions. The transportation planning is the recourse decision. The uncertainty in the demands was considered in two dimensions: *i*) the time between consumer orders (according to an exponential distribution), and *ii*) the order quantity (according to a log-Normal distribution). A SAA approach is proposed. However, using a general-purpose solver only small instances can be tackled. For this reason, the authors also present a heuristic algorithm based on tabu search. More recently, insights related to different types of anticipation methods for the uncertainty sources were studied in [46].

The Table 4.1 presents a summary with the surveyed papers.

4.2.2 Latency routing problems under uncertainty

The LLRP is a generalization of the cumulative capacitated vehicle routing problem (CCVRP). The CCVRP considers a single depot and a fleet of capacitated vehicles. Furthermore, the CCVRP generalizes the traveling repairman problem (TRP) which consists of a single depot and a single uncapacitated vehicle. The TRP is also known in the literature as the delivery man problem, and the minimum latency problem.

The LLRP, introduced in [47], considers homogeneous capacitated vehicles, and uncapacitated depots. Furthermore, no fixed costs for using the vehicles and opening the depots are considered. For solving the deterministic version of the problem, two heuristic algorithms are proposed: a memetic algorithm and a recursive granular algorithm. [48] present efficient MILP formulations, exact branch-and-cut algorithms and a greedy randomized heuristic. The exact methods are able to solve to proven optimality instances with up to 50 customers in reasonable computing times. More recently, [49] proposed three effective metaheuristics combining simulated annealing and variable neighborhood descent. The most effective metaheuristic algorithm currently published for the LLRP is the iterated local search method presented by [50] which outperforms the currently-published heuristic and exact algorithms.

After examining the literature related to LRP under uncertainty, it is possible to conclude that the only three works addressing a LLRP (at least considering latency as one of the objectives to minimize) are these presented in [19], [44], and [26]. The first two mentioned works consider the demand as uncertain source, while the third one considers uncertain travel times under a robust approach.

Due to the nature of the latency-based routing problems, it seems natural to consider uncertain travel times, and to adopt a risk-averse strategy for these problems. Indeed, the literature shows that most of the efforts regarding latency routing problems under uncertainty have followed such directions.

The traveling repairman problem with profits and uncertain travel times was studied in [51]. It consists of visiting a subset of customers such that the global expected profit (that depends on the expected arrival time) minus the variance of the latency is maximized. The problem was modeled using a non-linear single objective formulation, and was solved by using a beam search heuristic algorithm. The k-TRP with profits and uncertain travel times, a generalization of the above mentioned problem that considers a set of k uncapacitated vehicles, was studied in [52]. The problem was modeled as a weighted non-linear single-objective problem, in which the risk aversion is measured by a "weight parameter" that multiplies both components of the objective function: the expected profit and the variance of the latency. A GRASP algorithm was proposed

for solving the problem. A similar approach is proposed in [53] for the selective k-TRP with uncertain travel times. The main differences with respect to the problems mentioned before are the inclusion of service level constraints, and the objective function, which considers the expected latency and the variance of the latency in the weighted sum.

In this chapter we study an LLRP with stochastic travel times (LLRP-STT) considering a risk-averse decision maker, and propose a mathematical formulation and a metaheuristic for its solution. The problem is cast as a two-stage stochastic programming model in which the first stage seeks for determining the best depots to be opened, while the second stage decisions correspond to the route planning. To the best of our knowledge this is the first time the LLRP-STT is addressed. Furthermore, a multi-start variable neighborhood search metaheuristic algorithm is developed as an effective solution method. The algorithm allows handling a large number of scenarios, which is a key factor to obtain a good representation of the underlying randomness. The proposed metaheuristic can also handle instances with a large number of customers, which is not possible by means of the mathematical model.

4.3 Problem description and modeling aspects

In the specialized literature two most commonly used risk measures are the value-at-risk (VaR), and the conditional value-at-risk (CVaR). Putting into context, the VaR can be defined as a quantile of the latency distribution, while the CVaR is the expected latency beyond VaR. The risk-averse measure to be considered in this chapter is the CVaR, which has been shown to be a good risk indicator, while exhibiting appealing mathematical properties, which is not the case with the VaR [54].

For modeling the LLRP-STT we propose a two-stage stochastic MILP formulation based on the second model proposed in [48] for the LLRP. It corresponds to a multi-layer formulation, which has been proved to be effective for solving different latency vehicle routing problems.

Let us consider the following sets:

- C : Set of customers.
- D : Set of potential locations for the depots (assumed to be uncapacitated).
- V : Set of nodes, $V = C \cup D$.
- K : Set of identical vehicles.
- Ω: Set of scenarios. We assume that each scenario specifies one possible realization of the travel time matrix.

The parameters are summarized in the following:

- p: Maximum number of depots to be opened.
- d_i: Deterministic demand of customer $i \in C$, with $d_i \geq 0$.
- t_{ij}^w: Travel time between nodes i and j ($i, j \in V, i \neq j$, with $t_{ij}^w = t_{ji}^w$) under scenario $w \in \Omega$.
- Q: Capacity of the vehicles.
- π_w: Probability of occurrence associated with scenario $w \in \Omega$, with $\pi_w > 0 \ \forall w \in \Omega$ and $\sum_{w \in \Omega} \pi_w = 1$.

- α: Confidence level, with $0 \leq \alpha < 1$.

The LLRP-STT can be defined as follows. Let us consider the complete undirected graph $G = (V, E)$, where E is the set of edges connecting nodes in V. Each edge $(i, j) \in E$, with $i \neq j$, has an associated non-negative stochastic travel time t_{ij}^w under each scenario $w \in \Omega$, which satisfies the triangle inequality. The LLRP-STT comprises two decision levels. The first, which must be implemented before uncertainty is revealed, correspond to the selection of at most p depots to be opened. The second-stage decisions, which are implemented after uncertainty is revealed, correspond to the definition of the routes that the vehicles must perform starting from an open depot. If a depot $i \in D$ is opened, it must be used in all the scenarios $w \in \Omega$, i.e., at least one vehicle must perform a route based upon that depot. This assumption is reasonable for real-life problems since the decision to build facilities (such as hospitals or shelters) is a long-term decision, and this type of facility has always vehicles allocated to it in order to be operative. Each customer must be visited once. The routes of the vehicles (second stage decisions) may change depending on the realized values of the random data concerning the travel times (t_{ij}^w), and the sum of the demands associated with each route in each scenario must not exceed the capacity of the vehicle. The objective is to minimize the CVaR_α of the latency of the system. It is to note that in latency routing problems when no fixed cost is considered for using the vehicles, and the triangle inequality holds for the travel times t_{ij}^w, the optimal solution always uses $\min\{|K|, |C|\}$ vehicles. On the other hand, in an optimal solution, the number of open depots can be smaller than p (for example, when the number of depots "close" to the customers is smaller than p).

A new set must be introduced in order to explain the proposed mathematical model. Let us consider L as the set of levels in the multi-layer network. $|L|$ corresponds to the upper bound on the number of nodes in the routes. It is computed by sorting the demands of the customers in ascending order, and then defining $|L| - 1$ as the maximum number of demands such that their sum is less than or equal to the capacity of the vehicles. The meaning of the levels in the multi-layer network is the opposite w.r.t. the position of the nodes in the routes, that is, the first level of the multi-layer network represents the last customers in the routes. Since the latency of a route can be computed as the sum of the values obtained by multiplying the number of remaining customers in the route after each edge by the travel time associated with the edge, the objective of the multi-layer network is, indeed, to determine the number of times that each edge impacts in the objective function. This is determined by the level at which the edge appears in the multi-layer network.

The following decision variables are introduced for defining the MILP formulation of the LLRP-STT:

- z_i: 1 if depot $i \in D$ is opened, 0 otherwise. These variables correspond to the first-stage decisions in our stochastic problem.

- x_{ir}^w: 1 if customer $i \in C$ appears in level r ($r \in L$) of the multi-layer network under scenario $w \in \Omega$, 0 otherwise.

- y_{ijr}^w: 1 if the edge starting from node $i \in V$, $i \neq j$, incident to customer $j \in C$ that appears in level $r \in L$, $r \neq |L|$ of the multi-layer network, is in the solution under scenario $w \in \Omega$, 0 otherwise. When y_{ijr}^w is equal to 1, r represents the number of customers in the route after edge (i, j), including customer j.

- v_{ij}^w: The load of the vehicle when it departs from depot $i \in D$ to visit customer $j \in C$, under scenario $w \in \Omega$.

- l_{ij}^w: The total load of the vehicle when it departs from customer $i \in C$ to arrive at customer $j \in C$, $i \neq j$, under scenario $w \in \Omega$.

- η: Latency VaR_α.

- Ψ_w: Auxiliary variable used to measure the excess latency with respect to η under scenario $w \in \Omega$.

In order to provide a clear explanation of the formulation, we present the optimal solution (considering the deterministic problem, i.e., $|\Omega| = 1$) for one instance that consists of 20 customers, 5 candidate depots, at most 2 depots to be opened, and 3 vehicles (see Figure 4.1). This corresponds to instance 20-5-2b from the Prodhon data set, usually used as benchmark for the LLRP [47, 48, 49, 50]. In the optimal solution, the depots to be opened are *dep*4 and *dep*5. Two routes (R1 and R2) must be performed from depot *dep*4 and one route (R3) from depot *dep*5. R1=*dep*4-1-6-4-2-5-8, visiting 6 customers, R2=*dep*4-10-3-9-7-15-14-18 visiting 7 customers, and R3=*dep*5-12-11-13-16-20-17-19 visiting 7 customers. This means that depot *dep*4 will appear at levels 7 and 8 of the multi-layer network, and depot *dep*5 will appear at level 8. It is to note that each customer can appear only once in the multi-layer network, while each depot can appear more than once. The level r at which each customer i appears is defined by variable x_{ir}. The edges (i, j) shown at level r in Figure 4.1 represent the variables y_{ijr} whose value is equal to 1. We are omitting the scenario index because we are assuming $|\Omega| = 1$.

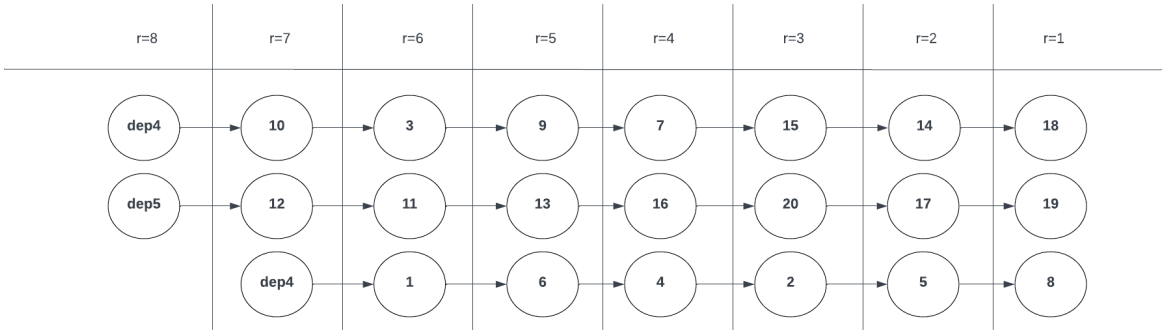


Figure 4.1: Representation of the optimal solution in the multi-layer network for a specific instance.

The proposed stochastic MILP formulation corresponds to (4.1)–(4.22):

$$\text{minimize} \quad \eta + \frac{1}{1-\alpha} \sum_{w \in \Omega} \pi_w \Psi_w, \tag{4.1}$$

subject to

$$\sum_{i \in D} z_i \leq p, \tag{4.2}$$

$$\Psi_w \geq \left(\sum_{i \in D} \sum_{j \in C} \sum_{r \in L \setminus \{|L|\}} rt_{ij}^w y_{ijr}^w + \sum_{i \in C} \sum_{j \in C \setminus \{i\}} \sum_{r \in L \setminus \{|L|\}} rt_{ij}^w y_{ijr}^w \right) - \eta \qquad \forall w \in \Omega, \tag{4.3}$$

$$\sum_{r \in L} x_{ir}^w = 1 \qquad \forall w \in \Omega, \forall i \in C, \tag{4.4}$$

$$\sum_{r \in L \setminus \{|L|\}} \sum_{i \in D} \sum_{j \in C} y_{ijr}^w \leq |K| \qquad \forall w \in \Omega, \tag{4.5}$$

$$\sum_{r \in L \setminus \{|L|\}} \sum_{j \in C} y_{ijr}^w \geq z_i \qquad \forall w \in \Omega, \forall i \in D, \tag{4.6}$$

$$\sum_{i \in C} x_{i1}^w = \sum_{r \in L \setminus \{|L|\}} \sum_{i \in D} \sum_{j \in C} y_{ijr}^w \qquad \forall w \in \Omega, \tag{4.7}$$

$$\sum_{j \in C \setminus \{i\}} y_{ijr}^w = x_{ir+1}^w \qquad \forall w \in \Omega, \forall i \in C, \forall r \in L \setminus \{|L|\}, \tag{4.8}$$

$$\sum_{i \in V \setminus \{j\}} y_{ijr}^w = x_{jr}^w \qquad \forall w \in \Omega, \forall j \in C, \forall r \in L \setminus \{|L|\}, \tag{4.9}$$

$$z_i \geq \sum_{r \in L \setminus \{|L|\}} y_{ijr}^w \qquad \forall w \in \Omega, \forall i \in D, \forall j \in C, \tag{4.10}$$

$$\sum_{h \in D} v_{hi}^w + \sum_{j \in C \setminus \{i\}} l_{ji}^w - \sum_{j \in C \setminus \{i\}} l_{ij}^w = d_i \qquad \forall w \in \Omega, \forall i \in C, \tag{4.11}$$

$$v_{ij}^w \geq d_j^w \sum_{r \in L \setminus \{|L|\}} y_{ijr}^w \qquad \forall w \in \Omega, \forall i \in D, \forall j \in C, \tag{4.12}$$

$$l_{ij}^w \geq d_j^w \sum_{r \in L \setminus \{|L|\}} y_{ijr}^w \qquad \forall w \in \Omega, \forall i, j \in C | i \neq j, \tag{4.13}$$

$$v_{ij}^w \leq Q \sum_{r \in L \setminus \{|L|\}} y_{ijr}^w \qquad \forall w \in \Omega, \forall i \in D, \forall j \in C, \tag{4.14}$$

$$l_{ij}^w \leq (Q - d_i) \sum_{r \in L \setminus \{|L|\}} y_{ijr}^w \qquad \forall w \in \Omega, \forall i, j \in C | i \neq j, \tag{4.15}$$

$$z_i \in \{0, 1\} \qquad \forall i \in D, \tag{4.16}$$

$$x_{ir}^w \in \{0, 1\} \qquad \forall w \in \Omega, \forall i \in C, \forall r \in L, \tag{4.17}$$

$$y_{ijr}^w \in \{0, 1\} \qquad \forall w \in \Omega, \forall i \in V, \forall j \in C | i \neq j, \forall r \in L \setminus \{|L|\}, \tag{4.18}$$

$$v_{ij}^w \geq 0 \qquad \forall w \in \Omega, \forall i \in D, \forall j \in C, \tag{4.19}$$

$$l_{ij}^w \geq 0 \qquad \forall w \in \Omega, \forall i, j \in C | i \neq j, \tag{4.20}$$

$$\Psi_w \geq 0 \qquad \forall w \in \Omega, \tag{4.21}$$

$$\eta \geq 0. \tag{4.22}$$

The objective function (4.1) minimizes the CVaR_α of the latency of the system. Constraint (4.2)

establishes that at most p depots can be opened. The constraints (4.3)–(4.15) are imposed for each scenario $w \in \Omega$. Constraints (4.3) calculate the difference between the latency of the system and the VaR_α. Constraints (4.4) ensure that each customer is visited exactly once, it is, each customer must appear in one and only one level of the multi-layer network. In inequalities (4.5) the maximum number of vehicles to be used is established. Constraints (4.6) establish that if depot i is opened, it must be used in all the scenarios (at least one vehicle must perform a route starting from the depot). Constraints (4.7)–(4.9) allow to determine the sequence in which the nodes appear in the multi-layer network (consequently in the routes), linking variables x and y. Equalities (4.7) state that the number of customers visited at the end of each route (level 1 in the multi-layer network) is equal to the number of vehicles performing routes. Constraints (4.8) ensure that the edge (i, j) is in the solution only if i is visited exactly before j, i.e., if i appears in the immediately upper level in the multi-layer network w.r.t. j (which appears in level r). Constraints (4.9) ensure that if a customer j appears in level r, there must be an incident edge to j. Inequalities (4.10) guarantee that the routes start only at open depots. The load of each vehicle at each customer, is calculated with constraints (4.11). In (4.12), the load with which each vehicle starts the route is computed. Similarly, in (4.13) the load with which the vehicle departs from a certain customer is calculated, and it must be at least equal to the demand of the forthcoming customer in the route. The vehicle capacity constraints are ensured using (4.14) and (4.15) for the cases when the vehicle is departing from the depot, or from a customer, respectively. Finally in (4.16)–(4.22) the domain of the variables is presented.

The deterministic LLRP is NP-Hard [47]. Since the LLRP-STT can be reduced to the LLRP when $|\omega| = 1$, the LLRP-STT is NP-Hard. The current state-of-the-art exact methods for the LLRP (proposed in [48]) are able to solve to proven optimality only instances with up to 50 customers. The stochastic problem is even more challenging to solve.

It is to note that the feasibility conditions for the LLRPT-STT are the same w.r.t. the LLRP, and regard the capacity of the vehicles. For a certain instance, if the respective bin-packing problem has a feasible solution with a number of bins less than or equal to $|K|$, the instance is feasible. We assume that the parameters of the instances are such that the feasibility set is non-empty.

4.4 Methodology

In this Section our multi-start VNS (MS-VNS) is introduced. Furthermore, a sampling method used for tackling instances with continuous probability distribution is presented.

4.4.1 Heuristic algorithm

The MS-VNS algorithm is composed of two main phases. The key factors that determine the success of the proposed method are: *i*) the right selection of the depots to be opened, and *ii*) the right allocation of vehicles to the open depots, and their corresponding routes. These factors are accounted for in the corresponding first and second phases, respectively. The first phase seeks a set of promising depot configurations, providing a pool of initial solutions, each one with a different depot configuration. The second phase corresponds to an improvement phase with different local search routines, which are applied to each of the initial solutions in the pool.

A summary of the MS-VNS is presented in Algorithm 7, where CVaR_α^* and Z^* represent the best solution value found and the depot configuration associated with it, respectively. The details of each phase and the respective procedures are provided in the following two sections.

Algorithm 7: The MS-VNS algorithm.

Input: *Data*, *Algorithm parameters*
Output: CVaR_α^* & Z^* (Best feasible solution found for the stochastic problem)
Apply the first phase
return: $Start = \{s_0^1, s_0^2, ..., s_0^{|Start|}\}$
for $z \in \{1, ..., |Start|\}$ **do**
  for $w \in \Omega$ **do**
    $Update(s_0^z, w)$
    Apply the second phase to s_0^z
    return: s_{bf}^{zw}, Lat_w^z (Best feasible solution found for scenario w using depot configuration z and its respective latency)
  end
  Compute CVaR_α^z
end
return: CVaR_α^* & Z^*

The algorithm starts by generating a set of initial feasible solutions $Start = \{s_0^1, s_0^2, ..., s_0^{|Start|}\}$, each one considering a different depot configuration. Then for each depot configuration $z \in \{1, ..., |Start|\}$ an improvement procedure is applied to the associated solution s_0^z obtaining the best feasible solution s_{bf}^z for the considered depot configuration. Finally, the best solution among all the s_{bf}^z is reported as the global best feasible solution s_{bf}.

First phase

The generation of set *Start* is done as follows. In order to find promising configurations, that may be good for all the scenarios, the following four steps are repeated for each scenario $w \in \Omega$.

1. A giant Traveling Salesman Problem (TSP) tour, considering all customers and minimizing the sum of the travel times t_{ij}^w associated with scenario w, is constructed using the LKH-3 heuristic proposed in [55].

2. A partitioning procedure is applied by starting from an initial customer and by splitting the giant tour into clusters of consecutive customers such that the total load of each cluster is less than or equal to Q.

   If the number of clusters created is smaller than $|K|$, a splitting procedure is applied, which consists of a local search procedure comprising the following steps.

   - Select the cluster m containing the edge (i, j) with the largest travel time $(i, j \in C)$.
   - Split the cluster m by removing edge (i, j). Two new sub-clusters ($m1$ and $m2$) are created: $m1$ is the sub-cluster composed of the customers belonging to cluster m until customer i; $m2$ is the sub-cluster composed by the customers of cluster m from customer j to the final customer of cluster m.

   On the other hand, if the number of clusters created is larger than $|K|$, a repair procedure is applied to delete the least-loaded clusters until the number of clusters equals $|K|$. Each customer, say i, belonging to a least-loaded cluster is considered according to the order in which it is visited within the cluster. It is removed from its current position and is inserted

in its best position, say k, in a different cluster, say j, so as to minimize the score defined in equation (4.23):

$$ScoreIN_{ik}^{j} = \Delta instime_{j}^{ik} + \theta[\max\{0, (dc_j + d_i) - Q\}] \tag{4.23}$$

where $\Delta instime_{j}^{ik}$ represents the variation of the travel time of cluster j caused by the insertion of customer i in position k; dc_j represents the current load of cluster j, and θ represents a penalty (large positive value). If the total load of a cluster (say cluster j) exceeds the vehicle capacity, a swapping procedure is applied to two customers (say customers k and i, with $d_i < d_k$) with respect to their current clusters (say clusters j and ℓ, respectively, with $j \neq \ell$, and $dc_\ell < dc_j$), so as to minimize the following score:

$$\begin{aligned} ScoreSW_{ik}^{j\ell} = \Delta time_{j}^{ki} + \Delta time_{\ell}^{ik} + \theta[max\{0, (dc_j - d_k + d_i) - Q\} \\ + max\{0, (dc_\ell + d_k - d_i) - Q\}] \end{aligned} \tag{4.24}$$

where $\Delta time_{j}^{ki}$ (resp. $\Delta time_{\ell}^{ik}$) represents the variation of the travel time of cluster j (resp. cluster ℓ) caused by the exchange of the customers k and i. If no feasible splitting of the customers into $|K|$ clusters is found by this swapping procedure the clustering process is repeated starting from the customer following the previous initial customer.

Contrary to the methodology proposed in [50], where all the possible starting customers are evaluated, the clustering process proposed here is stopped when a feasible set of exactly $|K|$ clusters is found. Hereafter, the feasible set of clusters is denoted by CL.

3. For each depot $i \in D$ and for each cluster $j \in CL$, we define an allocation cost l_{ij}, which represents the total latency of the route starting from depot i and composed by the customers in cluster j. This allocation cost is obtained by applying an intra-route local search procedure $IntraVND$ (all details are presented in the following section) to the path corresponding to cluster j. This step also provides information about how the route should be if cluster j is selected to start from depot i (this information is stored).

4. With all the previous information, the pool of depot configurations to be explored is defined. First, a so-called *main configuration* with at most p open depots is obtained by solving the location-allocation integer linear programming (ILP) model (4.25)–(4.32), with the binary variables A_{ij} equal to 1 iff cluster $j \in CL$ is assigned to depot $i \in D$, and Z_i equal to 1 iff depot $i \in D$ is opened.

$$\text{minimize} \sum_{i \in D} \sum_{j \in CL} l_{ij} A_{ij}, \tag{4.25}$$

$$\text{subject to:} \tag{4.26}$$

$$\sum_{i \in D} A_{ij} = 1 \qquad \forall j \in CL, \tag{4.27}$$

$$Z_i \leq \sum_{j \in CL} A_{ij} \qquad \forall i \in D, \tag{4.28}$$

$$A_{ij} \leq Z_i \qquad \forall i \in D, \forall j \in CL, \tag{4.29}$$

$$\sum_{i \in D} Z_i \leq p, \tag{4.30}$$

$$A_{ij} \in \{0, 1\} \qquad \forall i \in D, \forall j \in CL, \tag{4.31}$$

$$Z_i \in \{0, 1\} \qquad \forall i \in D. \tag{4.32}$$

The objective function (4.25) seeks to minimize the total latency. Constraints (4.27) ensure that each cluster is allocated to exactly one depot. Constraints (4.28) establish that if depot i is opened, it must be used (at least one cluster must be allocated to it). Constraints (4.29) impose that the clusters can be allocated only to open depots. Constraint (4.30) ensures that the number of open depots is at most p. Finally, constraints (4.31) and (4.32) define the domain of the variables.

Model (4.25)–(4.32) is solved NC times (where NC is a given parameter), each time by imposing $Z_i = 0$, where i is the index of the open depot with the k^{th} smallest number of clusters allocated in the *main configuration*, with $k \in \{1, NC\}$.

Each time a promising configuration is selected the following information is stored: set of open depots, allocation of the routes to the depots, routes, latency, number of times that the current configuration has been selected, and scenario in which it was selected. This information determines in fact, a feasible solution for the problem. If a configuration was already selected, the objective functions are compared, and the one with the smallest latency is kept. This is the end of step 4.

A feasible LLRP solution corresponds to the obtained set of open depots and to the respective routes assigned to each depot. For each configuration z a feasible solution s_0^z is obtained and it is included in the set *Start*. The solutions in *Start* are sorted in decreasing order according to the number of times that were selected. The latency is used as tiebreaker. The first $\min\{MP, |Start|\}$ (where MP is a given parameter) solutions of *Start* are kept for being evaluated in the next phase, while the remaining solutions (if they exist) are discarded.

Despite the fact that several ingredients are similar to those considered for the LLRP by [50] and other related problems such as the CLRP [56], there are new aspects that make our methodology innovative and more efficient (in terms of computing times) with respect to the mentioned works. In the mentioned papers, the procedures of clustering, calculation of the allocation cost, and the location decision, are repeated N_c times in the first phase (initial solution for the mentioned papers). It means that (relatively) time consuming procedures such as applying local search and solving MILP models are repeated several times (specially for instances with a large number of customers). It is to note that the constructive phase proposed in [50] provides

good quality solutions, and also valuable information about the pool of depot configurations to explore (similar to the set *Start*). This exploration of depot configurations proved to be much more effective, both in terms of solution quality and computing time, than the random exploration proposed in [49].

In the current work we seek to find a good reduced pool of depot configurations as done in [50], but in a shorter computing time. Other ideas for reducing the pool of depot configurations have been also applied for solving the CLRP (see [57]). Furthermore, the routing optimization procedures that require considerable amount of computing time are avoided. In this way, the LKH-3 procedure used both in [49] and in [50] for constructing the initial solutions, and as a post-optimization routine at the end of the algorithms, is not considered in our MS-VNS. In addition, the local search procedures used in this work for the routing optimization only explore three neighborhoods: *i*) *Insertion*, *ii*) *Swap*, and *iii*) 2-*Opt*, which are the neighborhoods that produce the largest improvements in the previously mentioned works. In this way, faster local search routines are applied.

Second phase—Improvement

An improvement phase is applied to each promising depot configuration (starting from each solution $s_0^z \in Start$), and for each scenario $w \in \Omega$. That is, for each depot configuration we solve $|\Omega|$ deterministic multi-depot cumulative capacitated vehicle routing problems (MDCCVRP) obtaining the latency Lat_w^z of scenario w when the depot configuration z is used. With these values it is possible to trivially compute the CVaR_α associated with each configuration z, CVaR_α^z. Finally, the configuration with the smallest CVaR_α is selected, and the corresponding solution is reported as the best feasible solution found for the stochastic problem.

The improvement phase (see Algorithm 8) starts by setting the current solution s_c and the best feasible solution s_{bf}^z to the current depot configuration z, i.e., s_0^z. It is to note that the objective function of s_0^z is updated for each scenario $w \in \Omega$ ($Update(s_0^z, w)$). Then the local search procedure VND (see Algorithm 9) is applied to s_c until no improvement is found, obtaining the solution s_{vnd}. Infeasible solutions w.r.t. the capacity of the vehicles are allowed during the second phase. For a solution s_c, its solution value $f(s_c)$ considers a penalty a for each unit exceeding the capacity, where a is a parameter. If the solution is feasible, the penalty term is equal to 0. Let us consider $Neigh = \{Insertion, Swap, 2\text{-}Opt\}$ as the set of neighborhoods to explore. Since the proposed VNS algorithm contains randomized components it is repeated $tryVNS$ times, and the best solution among the $tryVNS$ obtained solutions is stored. The current best feasible solution is stored in s_{aux}, and it is updated for each try $t \in tryVNS$. The VNS algorithm consists of repeating for $iter_{max}$ iterations the following steps.

i) The current neighborhood to explore is set equal to the first one, i.e. *Insertion*,

ii) Until all the neighborhoods in $Neigh$ are explored, the following steps are repeated:

- Apply a shaking procedure, *Shake*, to the current solution s_c, obtaining a solution s_c'. The shaking procedure consists of applying SK random moves in the current neighborhood n.
- Apply Algorithm 9 to s_c', obtaining a solution s_c''. This algorithm consists of a VND descendent local search procedure which explores sequentially all the neighborhoods in $Neigh$ under the best improve criterion. First it explores all the moves in *Insertion* until no improvement is found, next it does the same for *Swap*, and then for 2-*Opt*. If after exploring all the neighborhoods an improvement was found in any of the neighborhoods the search is restarted from *Insertion*. The search stops when no improvement is found

in any of the three neighborhoods. Each time an improved solution s_c'' is obtained, it is compared with the best feasible solution s_{bf}^z, whose value is updated in case a better feasible solution is found ($IsFeasible(s_c'')$).

- If s_c'', the solution obtained after applying *Shake* and VND, is better than s_c, the current neighborhood n is set equal to the first one, and the current solution is set equal to s_c''. Otherwise, the next neighborhood is explored.

iii) If the current iteration number, say *iter*, is smaller than $iter_{RR}$, then the procedure *RouteRelocation* is applied to s_c obtaining an updated current solution s_c. *RouteRelocation* consists of randomly selecting a route r from an open depot with more than one route allocated, and reallocate r to a different open depot. The $IntraVND$ local search procedure is applied to r. This procedure is similar to the VND procedure already described, nevertheless, after exploring all the neighborhoods the search is not restarted. If there is no open depot with more than one route allocated, the procedure *RouteRelocation* is skipped. If the current iteration number *iter* is greater than or equal to $iter_{RR}$ the search is intensified applying step *ii*) to the current best feasible solution s_{bf}^z.

It is to note that for the particular case in which $|\Omega| = 1$, the problem is reduced to the deterministic LLRP. Consequently, a by-product of our work is, in fact, an efficient metaheuristic algorithm for the (deterministic) LLRP. Although the main contribution of this work is the stochastic approach used for facing the LLRP, the MS-VNS turns out to be a very effective method for finding feasible solutions, in short computing times, for the deterministic LLRP.

4.4.2 Sampling method

In this section we present a sampling method for tackling instances with continuous probability distributions, i.e., an infinite number of scenarios. The method is also valid if the number of scenarios is finite but too large for allowing solving the whole stochastic problem. The proposed sampling method is a general framework that can be applied both using the proposed model (4.1)–(4.22) or the MS-VNS algorithm. If the MS-VNS is used for tackling the problem, then the sampling method can be considered a simheuristic [58].

The sampling method starts by generating M samples, with each sample consisting of N independent randomly generated scenarios. For each sample $m \in \{1, ..., M\}$ the stochastic problem with a finite number of scenarios is solved obtaining the corresponding objective function value $\text{CVaR}_{\alpha m}^*$ and the corresponding depot configuration Z^{*m} (as described in the previous sections). A vector *PromisingLocation* stores the obtained depot configurations Z^{*m}, counting how many times the configuration is selected and its corresponding $\text{CVaR}_{\alpha m}^*$ value (the smallest $\text{CVaR}_{\alpha m}^*$ value is stored in case the configuration is selected more than once). This vector is sorted in descending order according to the number of times each configuration is selected, using the $\text{CVaR}_{\alpha m}^*$ as tiebreaker. These steps define the first phase of our procedure. Then, the T most promising depot configurations are evaluated in a second phase, repeating the following steps for each configuration $t \in \{1, ..., T\}$.

i) The depots to be opened are fixed according to the corresponding element of the vector $PromisingLocation_t$. If the mathematical model is used for solving the problem, it is equivalent to fix the values of variables Z_i, $\forall i \in D$. On the other hand, if the MS-VNS algorithms is being considered then it is equivalent to use the current promising depot configuration as starting point for the second phase of the algorithm (*PromisingLocation* is equivalent to *Start*).

ii) A single-scenario LLRP considering a randomly generated scenario and the fixed locations is solved. When the MS-VNS is considered, this step is equivalent to solving the second phase

Algorithm 8: The Improvement phase - VNS algorithm.

Input: s_0^z, $Neigh$, $iter_{max}$, $iter_{RR}$, SK, a, $tryVNS$
Output: s_{bf}^z ($Best\ feasible\ solution\ for\ the\ cuurent\ depot\ configuration$)

$s_c = s_{bf}^z = s_0^z$
$s_{vnd} = VND(s_c, s_{bf}^z, Neigh, a)$
$s_{aux} = s_{vnd}$
for $t = 0; t < tryVNS, t++$ **do**
  $iter = 0$
  $s_c = s_{bf}^z = s_{vnd}$
  `/* VNS */`
  while $(iter < iter_{max})$ **do**
    $n = 1$
    while $(n \leq |Neigh|)$ **do**
      $s_c' = Shake(s_c, n, SK, a)$
      $s_c'' = VND(s_c', s_{bf}^z, Neigh, a)$
      $n = n + 1$
      if $(f(s_c'') < f(s_c))$ **then**
        $s_c = s_c''$
        $n = 1$
      end
    end
    if $(iter < iter_{RR})$ **then**
      $s_c = RouteRelocation(s_c)$
    else
      $s_c = s_{bf}^z$
    end
    $iter = iter + 1$
  end
  if $(f(s_{bf}^z) < f(s_{aux}))$ **then**
    $s_{aux} = s_{bf}^z$
  end
end
$s_{bf}^z = s_{aux}$
return: s_{bf}^z

Algorithm 9: The VND search procedure.

Input: $s'_c,\ s^z_{bf},\ Neigh,\ a$
Output: s'_c $(New\ current\ solution),\ s^z_{bf}$ $(New\ current\ best\ feasible\ solution)$

```
1  flag_vnd = true
2  while (flag_vnd = true) do
3      flag_vnd = false
4      for (each n ∈ Neigh) do
5          flag_neigh = true
6          while (flag_neigh = true) do
7              s_vnd = sol(n, s'_c)
8              if (f(s_vnd) < f(s'_c)) then
9                  s'_c = s_vnd
10                 flag_vnd = true
11                 if (f(s'_c) < f(s^z_bf) and IsFeasible(s'_c)) then
12                     s^z_bf = s'_c
13                 end
14             else
15                 flag_neigh = false
16             end
17         end
18     end
19 end
```

return: $s'_c,\ s^z_{bf}$

for a single scenario. This step is repeated until N' independent randomly generated scenarios are evaluated. For each scenario $n \in \{1, ..., N'\}$ the latency Lat_n^t is stored.

iii) Finally, the estimated CVaR, say $\widehat{CVaR_\alpha}^t$ is calculated for each depot configuration t. The computation of $\widehat{CVaR_\alpha}^t$ is done straightforwardly as follows:

- Sort in ascending order all the N' values of Lat_n^t.
- Select the k^{th} smallest Lat_n^t, where k is equal to $\alpha N'$. This value represents the estimated VaR_α, $\widehat{\eta}_t$, when configuration t is selected.
- Calculate the expected value of all the latencies Lat_n^t larger than or equal to $\widehat{\eta}_t$. This value corresponds to $\widehat{CVaR}_\alpha^t$.

After all the promising depot configurations are evaluated, the one with the smallest value of $\widehat{CVaR_\alpha}^t$ is selected as the best depot configuration Z^*, and the corresponding value of $\widehat{CVaR_\alpha}^t$ is reported as the best feasible solution value $\widehat{CVaR_\alpha}^*$ for the problem .

Algorithm 10: The sampling method.

Input: Data, N, M, N', T, MS-VNS parameters
Output: $\widehat{CVaR}_\alpha^*$ Z^*
Generate M samples, each consisting of N independent randomly generated scenarios.
for $m \in M$ **do**
  Solve the stochastic problem using sample m.
  return: $\text{CVaR}_{\alpha m}^*$, Z^{*m}
end
return: $PromisingLocation$
for $t \in T$ **do**
  Fix the open depots according to $PromisingLocation_t$
  for $n \in N'$ **do**
    Solve the deterministic LLRP for a randomly generated independent scenario n using the depot configuration $PromisingLocation_t$
    return: Lat_n^t
  end
  return: $\widehat{CVaR}_\alpha^t$
end
return: $\widehat{CVaR}_\alpha^*$ Z^*

4.5 Computational experiments

The proposed MILP model and the MS-VNS algorithm were implemented in C++. All the experiments were carried out on an AMD Ryzen 7 2700X Eight-Core Processor running at 3.7 GHz with 64 GB RAM, under Linux Ubuntu 18.04 operative system. The MILP model was solved with CPLEX 20.1 under the default parameter configuration, with a time-limit equal to 10800 seconds. The MS-VNS algorithm was executed considering 5 random seeds (i.e., 5 runs were executed for each instance) in a single thread.

The parameter tuning process for the MS-VNS algorithm was performed using the *iterated racing for automatic algorithm configuration* (IRACE) method (see [59]). IRACE is an open-source software that applies sampling procedures with an elitist criterion in order to identify promising parameter configurations. This software has been used for the tuning process of many algorithms in the literature, and its output is a set of parameter configurations, which correspond to the most promising ones. Then, after preliminary experiments, the best configuration was selected considering the obtained solution quality and the required computing time. The following values were considered in the tuning procedure: $iter_{RR} \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$, $a \in \{0.1, 0.3, 0.5, 1, 3, 5\}$, $iter_{max} \in \{10, 20, 30\}$, $SK \in \{3, 5, 7, 10, 15\}$, $NC \in \{1, 2, \lceil \frac{p}{2} \rceil, p\}$, $MP \in \{3, 5, 10, |Start|\}$, $tryVNS \in \{3, 5, 10\}$. It is clear that when the parameters NC, MP, and $tryVNS$ take their largest value the algorithm must perform better w.r.t. when these parameters take small values. Nevertheless, in our preliminary experiments we found that this situation imply large computing times, without a considerable improvement in the quality of the solutions found.

The following values were selected for each parameter: $iter_{RR} = 0.3$, $a = 1$, $iter_{max} = 30$, $SK = 15$, $NC = p$, $MP = 10$, and $tryVNS$=3.

4.5.1 Test instances for the problem

Because the LLRP with stochastic travel times has been introduced in this work, there are no benchmark instances available. Hence, test instances were specifically generated for performing the tests we report in this section. We have considered as a starting point a subset of instances from the Prodhon data set initially introduced by [60] for the CLRP, and recently adapted as benchmark data set for the deterministic LLRP by [47]. The instances used in this work are: 20-5-1, 20-5-1b, 20-5-2, and 20-5-2b. Due to the complexity of the stochastic problem we propose 8 new deterministic small size instances with 10 customers that can be solved to optimality by CPLEX using the proposed model. The new instances were created taking as starting point the mentioned instances, i.e., we kept a subset of nodes, considering the original coordinates and demands.

We assume that each scenario occurs with the same probability, i.e.,. $\pi_w = \frac{1}{|\Omega|}$, $\forall w \in \Omega$.

For the confidence level we considered $\alpha = 0.9$.

For generating the stochastic travel times we considered the following two cases:

- For each edge (i, j), the travel times $t_{ij} \sim LogNormal(\mu, \sigma)$ with $\mu = \ln(\widehat{t_{ij}}) - \frac{\sigma^2}{2}$, and $\sigma = 1$, where $\widehat{t_{ij}}$ is the original travel time of the deterministic instance. Since $\sigma = 1$ we have that the expected value of the travel time for the edge (i, j) is $E[t_{ij}] = \widehat{t_{ij}}$.
- For each edge (i, j), the travel times $t_{ij} \sim unif\{0.2\widehat{t_{ij}}, 1.8\widehat{t_{ij}}\}$. We have that $E[t_{ij}] = \widehat{t_{ij}}$.

After the generation of the random data the Floyd–Warshall algorithm [61] is applied to the travel time matrix of each scenario to ensure that the triangle inequality holds.

For each base instance, 20, 30, 40, and 50 scenarios were generated according to the procedure already described.

A total of 96 instances were considered. The instances are available to any reader upon a request to the authors.

4.5.2 Evaluation of the proposed methods

Table 4.2: Detailed results for the LLRP-STT instances.

				CPLEX			MS-VNS		
Ω	C	Distribution	Instance	CvaR_{CPLEX}	time	mipgap	Best	time	Dev_{CPLEX}
20	10	LOGN	10-5-1	**62.45**	262.5	0.00	63.00	61.2	0.88
			10-5-1b	**128.93**	1569.3	0.00	**128.93**	50.9	0.00
			10-5-2	**54.06**	844.9	0.00	**54.06**	59.1	0.00
			10-5-2b	**72.84**	195.1	0.00	**72.84**	51.8	0.00
			10-10-1	**60.02**	2365.6	0.00	60.26	62.6	0.40
			10-10-1b	**95.96**	2972.9	0.00	**95.96**	53.2	0.00
			10-10-2	**53.20**	1585.6	0.00	**53.20**	61.7	0.00
			10-10-2b	**69.35**	405.2	0.00	**69.35**	52.0	0.00
		UNIF	10-5-1	**131.57**	337.7	0.00	**131.57**	62.0	0.00
			10-5-1b	**256.10**	1699.1	0.00	**256.10**	51.9	0.00
			10-5-2	**98.02**	442.7	0.00	**98.02**	59.0	0.00
			10-5-2b	**138.10**	235.7	0.00	**138.10**	50.0	0.00
			10-10-1	**110.71**	1440.2	0.00	111.09	64.2	0.34
			10-10-1b	**188.72**	2254.5	0.00	191.92	53.6	1.69
			10-10-2	**103.57**	3919.0	0.00	**103.57**	60.6	0.00
			10-10-2b	**136.05**	457.8	0.00	**136.05**	52.5	0.00
	20	LOGN	20-5-1	322.71	10800.0	70.81	104.67	384.8	-67.56
			20-5-1b	913.99	10800.0	82.13	177.88	342.7	-80.54
			20-5-2	338.22	10800.0	69.10	110.49	399.5	-67.33
			20-5-2b	404.61	10800.0	62.67	160.66	335.1	-60.29
		UNIF	20-5-1	641.02	10800.0	64.02	238.49	380.5	-62.80
			20-5-1b	1633.89	10800.0	77.68	389.92	340.8	-76.14
			20-5-2	582.84	10800.0	65.15	217.18	399.7	-62.74
			20-5-2b	1412.55	10800.0	77.33	340.13	344.1	-75.92
30	10	LOGN	10-5-1	**69.52**	9398.2	0.00	**69.52**	92.7	0.00
			10-5-1b	**139.69**	6418.5	0.00	**139.69**	78.8	0.00
			10-5-2	**52.86**	698.3	0.00	54.83	89.5	3.74
			10-5-2b	**79.49**	2549.1	0.00	**79.49**	77.6	0.00
			10-10-1	**56.02**	10588.3	0.00	**56.02**	94.8	0.00
			10-10-1b	**103.86**	2724.2	0.00	**103.86**	80.3	0.00
			10-10-2	**55.43**	5451.6	0.00	57.85	92.9	4.37
			10-10-2b	**76.47**	1111.9	0.00	**76.47**	79.9	0.00
		UNIF	10-5-1	**136.57**	1181.6	0.00	**136.57**	91.7	0.00
			10-5-1b	**266.45**	1727.1	0.00	**266.45**	77.2	0.00
			10-5-2	**106.70**	941.6	0.00	**106.70**	89.8	0.00
			10-5-2b	**141.94**	1392.7	0.00	**141.94**	78.4	0.00
			10-10-1	**109.55**	2086.8	0.00	**109.55**	95.8	0.00
			10-10-1b	**192.86**	7715.1	0.00	**192.86**	80.2	0.00
			10-10-2	**103.19**	10415.9	0.00	103.22	92.9	0.03
			10-10-2b	**150.62**	1948.2	0.00	**150.62**	74.7	0.00
	20	LOGN	20-5-1	343.72	10800.0	69.89	114.55	582.3	-66.67
			20-5-1b	749.22	10800.0	77.61	181.82	516.9	-75.73
			20-5-2	408.86	10800.0	76.94	102.69	605.1	-74.88
			20-5-2b	460.77	10800.0	66.07	165.57	516.4	-64.07
		UNIF	20-5-1	627.99	10800.0	65.21	229.25	585.7	-63.49
			20-5-1b	1387.33	10800.0	72.15	403.42	518.6	-70.92
			20-5-2	705.13	10800.0	70.81	213.17	602.6	-69.77
			20-5-2b	998.65	10800.0	67.04	353.68	510.9	-64.58

Continued on next page

Table 4.2 – *Continued from previous page*

				CPLEX			MS-VNS		
Ω	C	Distribution	Instance	CvaR_{CPLEX}	time	mipgap	Best	time	Dev_{CPLEX}
40	10	LOGN	10-5-1	**68.39**	8620.1	0.00	**68.39**	122.0	0.00
			10-5-1b	153.42	10800.0	8.71	147.30	103.7	-3.99
			10-5-2	56.30	10800.0	1.95	56.04	123.3	-0.46
			10-5-2b	**74.25**	3351.9	0.00	74.25	103.9	0.00
			10-10-1	184.72	10800.0	67.76	61.57	127.2	-66.67
			10-10-1b	98.67	10800.0	7.96	94.94	106.9	-3.78
			10-10-2	71.29	10800.0	29.79	54.27	123.6	-23.88
			10-10-2b	**76.41**	7447.2	0.00	**76.41**	104.5	0.00
		UNIF	10-5-1	**132.74**	6078.9	0.00	132.74	124.3	0.00
			10-5-1b	**255.42**	10600.4	0.00	255.42	103.1	0.00
			10-5-2	**105.03**	851.8	0.00	105.03	120.1	0.00
			10-5-2b	**141.36**	372.0	0.00	141.36	102.5	0.00
			10-10-1	294.51	10800.0	62.50	112.19	126.3	-61.91
			10-10-1b	196.74	10800.0	1.59	196.51	107.2	-0.12
			10-10-2	155.22	10800.0	31.92	108.53	116.8	-30.08
			10-10-2b	**136.54**	6059.6	0.00	136.54	105.9	0.00
	20	LOGN	20-5-1	351.62	10800.0	69.79	116.48	778.8	-66.87
			20-5-1b	2336580.00	10800.0	99.99	176.76	681.3	-99.99
			20-5-2	345.86	10800.0	72.47	101.98	802.3	-70.51
			20-5-2b	625.54	10800.0	74.96	169.88	688.3	-72.84
		UNIF	20-5-1	642.99	10800.0	66.17	227.95	771.5	-64.55
			20-5-1b	4882820.00	10800.0	99.99	425.23	686.3	-99.99
			20-5-2	613.57	10800.0	67.52	207.83	814.6	-66.13
			20-5-2b	1209.39	10800.0	72.96	342.62	684.2	-71.67
50	10	LOGN	10-5-1	177.80	10800.0	59.07	75.28	156.8	-57.66
			10-5-1b	**132.99**	2154.8	0.00	132.99	129.9	0.00
			10-5-2	88.71	10800.0	31.08	65.16	148.8	-26.55
			10-5-2b	80.06	10800.0	3.24	79.90	129.8	-0.20
			10-10-1	101.32	10800.0	44.65	57.10	158.8	-43.64
			10-10-1b	**104.94**	5227.9	0.00	104.94	133.4	0.00
			10-10-2	99.68	10800.0	48.13	53.36	152.5	-46.47
			10-10-2b	75.53	10800.0	0.02	75.53	133.3	0.00
		UNIF	10-5-1	191.49	10800.0	29.02	135.92	159.4	-29.02
			10-5-1b	**258.74**	3556.8	0.00	258.74	129.3	0.00
			10-5-2	**103.04**	2722.8	0.00	103.06	150.4	0.02
			10-5-2b	**138.97**	7753.2	0.00	138.97	129.8	0.00
			10-10-1	248.30	10800.0	56.69	108.80	160.3	-56.18
			10-10-1b	207.43	10800.0	3.22	203.81	134.9	-1.75
			10-10-2	250.59	10800.0	59.70	103.05	153.1	-58.88
			10-10-2b	**142.48**	5805.5	0.00	142.48	131.7	0.00
	20	LOGN	20-5-1	366.93	10800.0	72.68	109.05	970.5	-70.28
			20-5-1b	2406240.00	10800.0	99.99	196.38	855.6	-99.99
			20-5-2	354.59	10800.0	72.60	107.96	997.1	-69.55
			20-5-2b	2234180.00	10800.0	99.99	166.36	857.3	-99.99
		UNIF	20-5-1	997108.00	10800.0	99.98	237.60	965.4	-99.98
			20-5-1b	4920480.00	10800.0	99.99	417.02	854.9	-99.99
			20-5-2	634.77	10800.0	69.84	208.26	1011.3	-67.19
			20-5-2b	4448540.00	10800.0	99.99	336.37	830.1	-99.99
		Global Avg.		231783.63	7270.2	31.46	**149.18**	**281.3**	**-30.45**

The detailed results obtained both by the model and MS-VNS for each instance are presented in Table 4.2. The values presented in boldface correspond to proven optimal solution values.

In order to evaluate the performance of the proposed heuristic method let us consider Dev_{CPLEX} as the deviation between the best values found by MS-VNS and CPLEX. It is computed as $\text{Dev}_{CPLEX} = 100\frac{(Best - CVaR_{CPLEX})}{CVaR_{CPLEX}}$, where $Best$ corresponds to the best solution value found by MS-VNS (within the 5 runs) and $CVaR_{CPLEX}$ corresponds to the best value reported by CPLEX within the time-limit.

The computing times reported in Table 4.2 for MS-VNS correspond to the global computing time, it is the time required for running each instance 5 times.

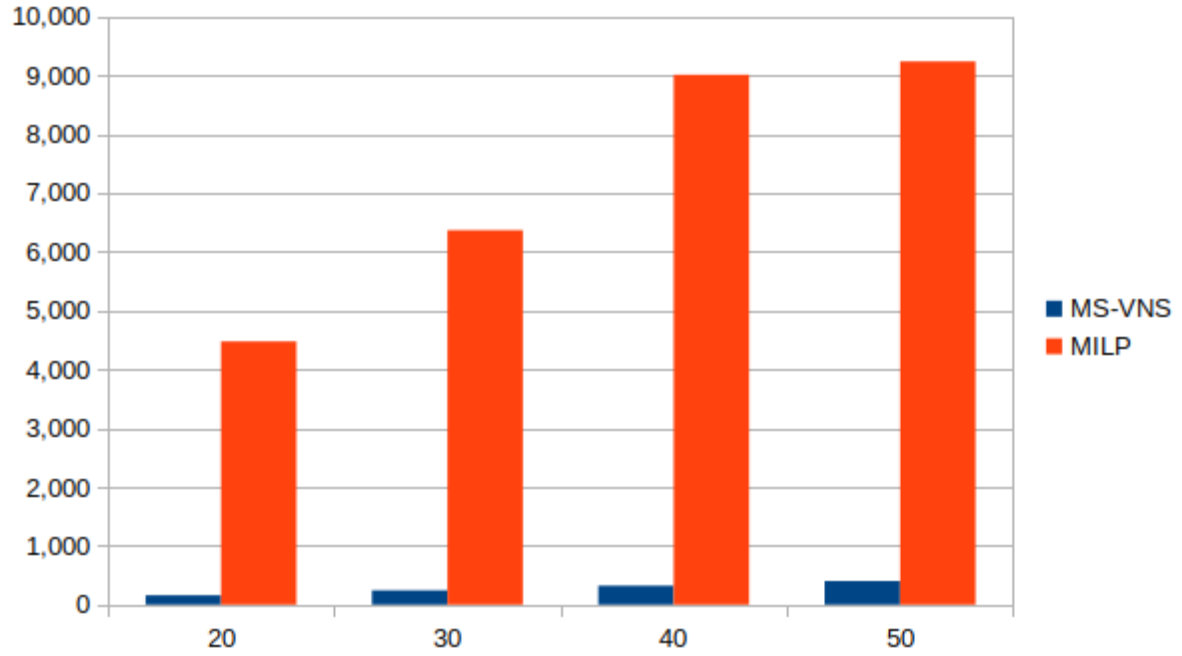


Figure 4.2: Average computing time (s) required by the proposed methods for solving instances with 20, 30, 40 and 50 scenarios.

The proposed model was solved to proven optimality by the solver for 46 out of the 96 instances within the time-limit. However, none of the instances with 20 customers was solved to optimality. Moreover the average mipgap reported for these instances is equal to 77.3%. The model can solve to proven optimality all the instances with 10 customers considering up to 30 scenarios.

MS-VNS is able to find a solution value better than (50) or equal to (38) the solution value reported by CPLEX for 88 out of the 96 instances analyzed in computing times considerably shorter.

Considering only the instances solved to proven optimality, the heuristic is able to find the same solution value for 38 out of the 46 respective instances. For the 8 remaining instances the average value of Dev_{CPLEX} is equal to 1.43%. For all the instances analyzed the global computing time of MS-VNS (considering the 5 runs) is smaller than the computing time required by CPLEX. On the other hand, for all the instances with 20 customers the heuristic algorithm provides a solution value better than CPLEX.

The computing times required by the solver and by the heuristic is depicted in Figure 4.2.

Note that the solver reaches the time-limit for almost all the instances with 40 and 50 scenarios (34 out of 48). The computing times needed by the model are above 25 times larger than these required by MS-VNS. The behavior of the MS-VNS in terms of computing time makes it quite appealing.

Regarding the probability distribution underlying the travel times, we can see (Table 4.2) that the solver requires for the log-Normal distribution slightly larger computing times than those required for the Uniform one (7658.80 s and 6884.37 s, respectively). In the same way, the mipgap is 32.08% and 30.84%, with 21 and 25 instances solved to proven optimality for the log-Normal and Uniform distribution, respectively. On the other hand, MS-VNS performs similarly irrespective of the probability distribution.

4.5.3 The relevance of considering a stochastic modeling framework

Stochastic programming problems are often more challenging to solve than their deterministic counterparts. For the case of the (deterministic) LLRP, the mathematical formulations presented by [48] can easily deal with 20-customers instances. On the other hand, for all the instances with $|C| = 20$, the proposed stochastic programming model is not able to be solved to proven optimality even for 20 scenarios. Therefore, it is important to study the relevance of using the stochastic model over simpler versions namely these considering the expected values of the stochastic parameters. In this section we present an analysis of the relevance of the stochastic problem by considering adapted versions of two well-known measures: the expected value of perfect information (EVPI), and the value of the stochastic solution (VSS). The EVPI assumes a risk-neutral decision maker. It can be defined as the the maximum value that the decision maker is willing to pay to get a complete knowledge about the future. It corresponds to the difference between the optimal solution value of the stochastic problem $\upsilon(P)$ and the value of the wait and see solution WSS, which is the expected value of the solutions obtained when the decision maker has perfect information about the future. The VSS compares the optimal stochastic solution value $\upsilon(P)$ with the value of EV, i.e., the value of the expected value solution (the expected values of the stochastic parameters are considered and the resulting deterministic problem is solved). Since these two metrics are defined for risk-neutral decision makers, we propose the following adapted versions considering a risk-averse (RA) attitude, based on those proposed by [62].

- RAVPI=$100\frac{\upsilon(P)-RAWSS}{\upsilon(P)}$, where RAWSS corresponds to the CVaR_α calculated after finding the optimal solution value for each single scenario problem $w \in \Omega$. Since $\upsilon(P) \geq$ RAWSS, it is always true that RAVPI ≥ 0.
- RAVSS=$100\frac{RAEVS-\upsilon(P)}{RAEVS}$, where RAEVS can be computed as follows: *i*) Solve the single-scenario problem considering the expected values of the random variables, obtaining the optimal depot configuration associated with it. *ii*) Solve the second-stage problem for each scenario $w \in \Omega$ considering the previously mentioned depot configuration (fixing the depots to be open). With all that information it is possible to compute the CVaR_α associated with the depot configuration selected by considering the expected values of the random variables. Since $\upsilon(P) \leq$ RAEVS, it is always true that RAVSS ≥ 0.

It is to note that both metrics are presented as percentage in order to make it easy to interpret irrespective of the magnitude of the parameters of each instance. Furthermore, since the computation of RAVPI and RAVSS requires solving to optimality the stochastic problem, only the 46 instances that have been solved to proven optimality were considered for the analysis.

Table 4.3: Detailed values of RAVPI and RAVSS for the LLRP-STT instances solved to proven optimality.

Instance	$v(P)$	RAWSS	RAVPI	RAEVS	RAVSS
10-5-1-E20-LOGN	62.45	56.26	9.92	64.22	2.76
10-5-1b-E20-LOGN	128.93	117.39	8.95	130.43	1.15
10-5-2-E20-LOGN	54.06	46.57	13.85	60.48	10.63
10-5-2b-E20-LOGN	72.84	68.23	6.33	74.17	1.79
10-10-1-E20-LOGN	60.02	54.27	9.58	64.03	6.26
10-10-1b-E20-LOGN	95.96	88.30	7.97	96.50	0.56
10-10-2-E20-LOGN	53.20	52.12	2.03	61.06	12.86
10-10-2b-E20-LOGN	69.35	65.76	5.18	73.33	5.43
10-5-1-E30-LOGN	69.52	65.60	5.64	71.51	2.79
10-5-1b-E30-LOGN	139.69	125.86	9.90	148.36	5.85
10-5-2-E30-LOGN	52.86	48.12	8.97	63.79	17.14
10-5-2b-E30-LOGN	79.49	74.69	6.04	79.63	0.17
10-10-1-E30-LOGN	56.02	48.37	13.65	59.29	5.53
10-10-1b-E30-LOGN	103.86	98.90	4.77	117.19	11.38
10-10-2-E30-LOGN	55.43	49.16	11.31	72.30	23.33
10-10-2b-E30-LOGN	76.47	73.63	3.73	79.03	3.23
10-5-1-E40-LOGN	68.39	58.19	14.92	69.01	0.89
10-5-2b-E40-LOGN	74.25	71.52	3.67	75.67	1.88
10-10-2b-E40-LOGN	76.41	73.14	4.28	77.32	1.19
10-5-1b-E50-LOGN	132.99	129.43	2.68	140.13	5.09
10-10-1b-E50-LOGN	104.94	99.88	4.83	112.84	6.99
10-5-1-E20-UNIF	131.57	125.74	4.43	140.58	6.41
10-5-1b-E20-UNIF	256.10	231.38	9.65	264.80	3.29
10-5-2-E20-UNIF	98.02	94.50	3.59	117.12	16.31
10-5-2b-E20-UNIF	138.10	135.66	1.77	143.09	3.49
10-10-1-E20-UNIF	110.71	98.00	11.49	118.89	6.88
10-10-1b-E20-UNIF	188.72	177.11	6.15	208.99	9.70
10-10-2-E20-UNIF	103.57	93.10	10.11	113.85	9.02
10-10-2b-E20-UNIF	136.05	128.17	5.80	136.51	0.34
10-5-1-E30-UNIF	136.57	134.16	1.76	141.91	3.77
10-5-1b-E30-UNIF	266.45	236.69	11.17	295.94	9.97
10-5-2-E30-UNIF	106.70	105.30	1.31	117.00	8.81
10-5-2b-E30-UNIF	141.94	134.15	5.49	143.90	1.36
10-10-1-E30-UNIF	109.55	101.02	7.79	116.38	5.86
10-10-1b-E30-UNIF	192.86	178.41	7.49	198.21	2.70
10-10-2-E30-UNIF	103.19	95.49	7.45	106.75	3.34
10-10-2b-E30-UNIF	150.62	143.87	4.48	151.44	0.54
10-5-1-E40-UNIF	132.74	125.94	5.12	133.56	0.61
10-5-1b-E40-UNIF	255.42	237.65	6.96	257.66	0.87
10-5-2-E40-UNIF	105.03	97.77	6.91	106.78	1.64
10-5-2b-E40-UNIF	141.36	137.11	3.00	145.51	2.86
10-10-2b-E40-UNIF	136.54	130.33	4.55	140.35	2.71
10-5-1b-E50-UNIF	258.74	249.67	3.51	270.22	4.25
10-5-2-E50-UNIF	103.04	94.65	8.14	113.66	9.34
10-5-2b-E50-UNIF	138.97	133.00	4.29	139.31	0.24
10-10-2b-E50-UNIF	142.48	135.57	4.85	144.87	1.64
Gloval avg.	118.96	111.30	6.64	125.16	5.28

The detailed results are reported in Table 4.3.

For the RAVPI the minimum, average, and maximum values obtained were 1.31%, 6.64%, and 14.92%, respectively. On the other hand, for the RAVSS the minimum, average, and maximum values obtained were 0.17%, 5.28%, and 23.33%, respectively. The above mentioned values allow us to conclude the following:

In the case we have perfect information about the future, the solution value can be at least 1.31% better than if uncertainty is not disclosed. The average and maximum RAVPI values suggest that capturing uncertainty in the problem is of major relevance.

Since for all the instances analyzed RAVSS is larger than 0.0%, the optimal location decisions for the stochastic problem are never equal to those induced by the deterministic problem (considering the expected values of the travel times). On average the solutions obtained by adopting the simplified expected value problem are 5.28% worse than those found by the stochastic model, and the difference can be up to 23.33%. In real life situations, e.g., disaster operations management in general, or in humanitarian logistics in particular, the above mentioned percentages can be quite significant. To not consider the stochastic problem in the planning may imply arriving much later to provide assistance to the affected areas.

By comparing the results aggregated by probability distribution, we found that both metrics are larger for log-Normal instances than for Uniform ones (RAVPI:7.53%, 5.89%, and RAVSS: 6.04%, 4.64% for log-Normal and Uniform, respectively). Despite the above, its possible to see that even for instances with Uniform distribution these values are not negligible, which means that it is relevant to capture stochasticity irrespective of the underlying probability distribution.

4.5.4 The effect of considering a risk-averse decision maker

In this Section we analyze how the solutions change when a risk-neutral decision maker is considered ($\alpha = 0.0$, i.e., all the scenarios come into play). For this experiment only the 46 instances solved to proven optimality were considered. They were solved to proven optimality by setting $\alpha = 0.0$.

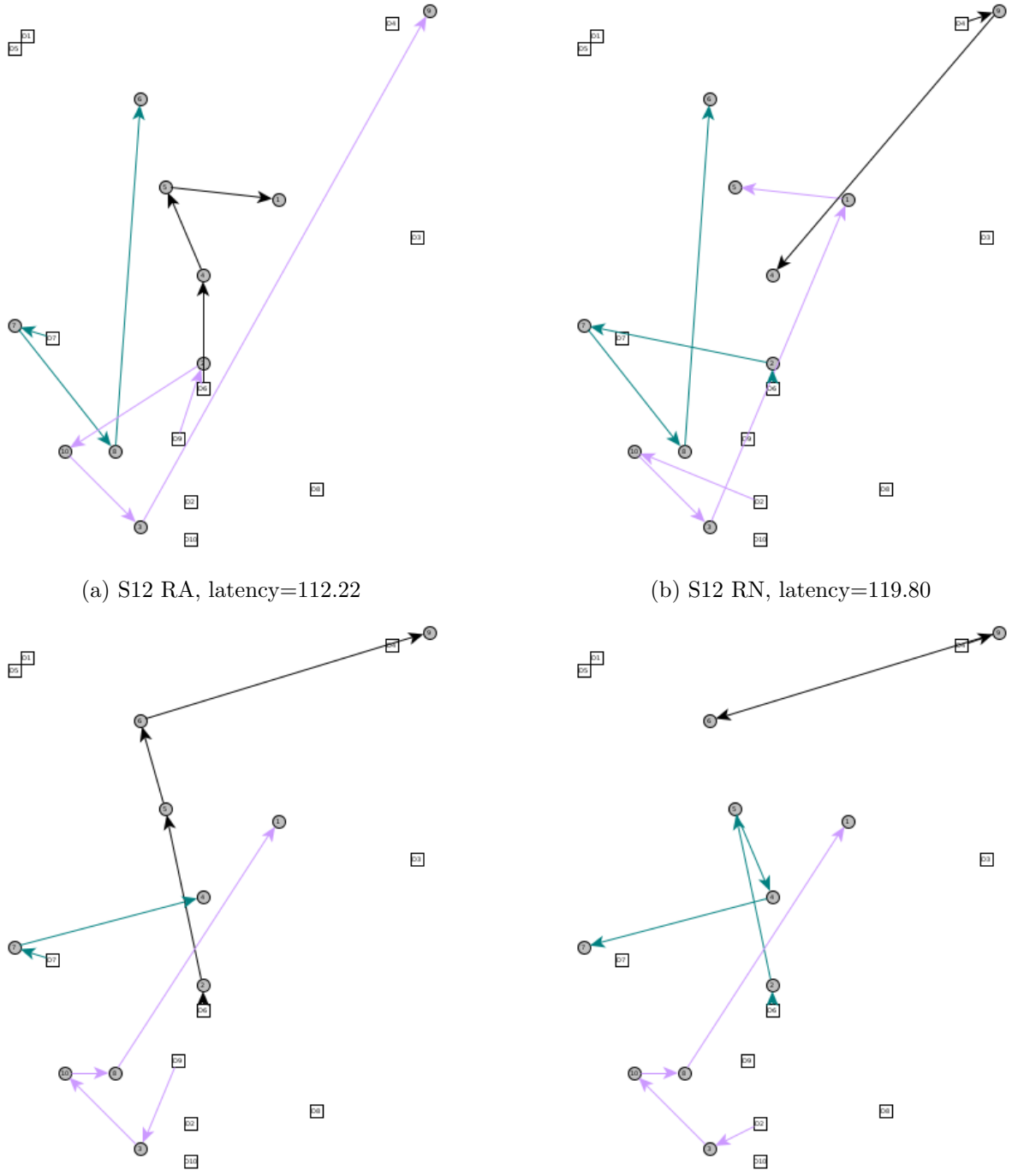
The detailed results are presented in Table 4.4.

For 35 out of the 46 instances analyzed the set of depots opened by considering the risk-averse (RA) is different w.r.t. that if a risk-neutral (RN) decision maker is considered. Naturally, the second stage decisions are conditioned by the first stage decisions.

In order to understand how risk-aversion influences the solutions let us consider the example presented in Figure 4.3 for the instance 10-10-1-E20-UNIF. The depots to be open under the risk-averse attitude are 6, 7 and 9, while for the risk-neutral case the depots to be open are 2, 4, and 6. In Figure 4.3 the circles represent the customers, the squares the depots, and each color represent a vehicle. The Figures 4.3a and 4.3c show the optimal solution of the second stage problem for the (1-α) worst-case scenarios (namely S12 and S18), with $\alpha = 0.9$. In Figures 4.3b and 4.3d the optimal solution of the second stage problem for the same scenarios, but considering the depots opened when $\alpha = 0$, are presented. Despite the latency of S18 is smaller by considering a risk-neutral attitude, it does not mean that the CVaR_α (with $\alpha = 0.9$) associated with the depot configuration 2-4-6 is smaller than 110.71. Indeed, if CVaR_α is calculated by fixing the mentioned depots its value is equal to 114.02; the above is explained by the fact that the (1-α) worst-case scenarios for the mentioned depot configuration are S12 and S3 instead of S12 and S18. A major conclusion drawn from this figure is that the solution in the worst-case scenarios is clearly taken into account for deciding the depots.

Table 4.4: Detailed results for the LLRP-STT instances solved to proven optimality considering both risk-averse and risk-neutral attitudes.

	RA		RN	
Instance	$CVAR_\alpha$	depots	E[latency]	depots
10-5-1-E20-LOGN	62.45	2-5	52.17	1-2
10-5-1b-E20-LOGN	128.93	1-3	97.21	1-5
10-5-2-E20-LOGN	54.06	3-4-5	41.23	4-5
10-5-2b-E20-LOGN	72.84	4	56.14	1-4
10-10-1-E20-LOGN	60.02	6-9-10	45.01	6-7-9
10-10-1b-E20-LOGN	95.96	7-10	76.69	3-10
10-10-2-E20-LOGN	53.20	4-5	39.16	5-10
10-10-2b-E20-LOGN	69.35	3-4	56.23	1-4
10-5-1-E30-LOGN	69.52	2-4-5	51.38	2-4-5
10-5-1b-E30-LOGN	139.69	3-4	101.34	1-3
10-5-2-E30-LOGN	52.86	3-5	39.14	4-5
10-5-2b-E30-LOGN	79.49	1-3	55.40	1-4
10-10-1-E30-LOGN	56.02	4-6-7	45.89	4-6-7
10-10-1b-E30-LOGN	103.86	7-10	76.98	9-10
10-10-2-E30-LOGN	55.43	5-6-10	41.35	5-10
10-10-2b-E30-LOGN	76.47	3-4	55.21	1-4
10-5-1-E40-LOGN	68.39	2-4-5	53.52	2-4
10-5-2b-E40-LOGN	74.25	4	54.41	1-4
10-10-2b-E40-LOGN	76.41	4	53.76	1-4
10-5-1b-E50-LOGN	132.99	3-5	100.52	3-5
10-10-1b-E50-LOGN	104.94	9-10	79.50	9-10
10-5-1-E20-UNIF	131.57	2-3-4	108.50	2-4
10-5-1b-E20-UNIF	256.10	3-5	210.48	1-3
10-5-2-E20-UNIF	98.02	3-4-5	84.33	4-5
10-5-2b-E20-UNIF	138.10	4	103.90	1-4
10-10-1-E20-UNIF	110.71	6-7-9	88.98	2-4-6
10-10-1b-E20-UNIF	188.72	5-10	161.30	3-10
10-10-2-E20-UNIF	103.57	3-5-10	81.83	3-5
10-10-2b-E20-UNIF	136.05	4	109.87	1-4
10-5-1-E30-UNIF	136.57	2-4	109.39	2-4
10-5-1b-E30-UNIF	266.45	3	220.94	1-3
10-5-2-E30-UNIF	106.70	4-5	87.34	5
10-5-2b-E30-UNIF	141.94	3-4	110.25	1-4
10-10-1-E30-UNIF	109.55	6-7-9	92.16	6-7-9
10-10-1b-E30-UNIF	192.86	9-10	161.80	9-10
10-10-2-E30-UNIF	103.19	5-10	85.85	5-10
10-10-2b-E30-UNIF	137.67	1-4	112.73	1-4
10-5-1-E40-UNIF	132.74	2-4-5	111.09	2-4
10-5-1b-E40-UNIF	255.42	3-5	216.31	1-3
10-5-2-E40-UNIF	105.03	4-5	84.99	5
10-5-2b-E40-UNIF	141.36	4	113.69	1-4
10-10-2b-E40-UNIF	136.54	4	107.22	1-4
10-5-1b-E50-UNIF	258.74	1-3	215.21	1-3
10-5-2-E50-UNIF	103.04	4-5	81.82	4-5
10-5-2b-E50-UNIF	138.97	4	114.28	1-4
10-10-2b-E50-UNIF	142.48	4	108.20	1-4
Gloval avg.	118.68		94.67	



(a) S12 RA, latency=112.22

(b) S12 RN, latency=119.80

(c) S18 RA, latency=109.20

(d) S18 RN, latency=96.11

Figure 4.3: Risk-averse and risk-neutral solutions for the instance 10-10-1-E20-UNIF.

4.5.5 Sampling method - tackling instances with continuous probability distributions

In this section the terms MS-VNS and simheuristic are used interchangeably, referring to the execution of the sampling method by means of the MS-VNS heuristic. For these experiments we considered the following values for each parameter: $M = 100$, $N \in \{10, 20, 30\}$, $N' = 2000$, and $T = 5$.

Let us consider AVG and STD as the average and the standard deviation of the optimal solution values associated with each sample $m \in \{1, 2, ..., M\}$ ($\text{CVaR}^*_{\alpha m}$), respectively, calculated as follows: $AVG = \frac{1}{M}\sum_{m=1}^{M} CVaR^*_{\alpha m}$, and $STD = \sqrt{\frac{1}{M-1}\left[(\sum_{m=1}^{M}(CVaR^*_{\alpha m})^2) - M(AVG^2)\right]}$.

Since the number of samples is large ($M = 100$), due to the Central Limit theorem, it is possible to assume the mean normally distributed. Thus, the 95% confidence interval (CI) for the true optimal solution value of each instance can be defined as $CI : (AVG \pm \frac{1.96STD}{\sqrt{M}})$. In other words, we are 95% confident that the true optimal solution value of each instance is between CI^- and CI^+, where CI^- and CI^+ are the lower and upper limits of the CI, respectively. For the computation of the CI of each instance, the sampling configuration with $N = 30$ was considered.

The detailed results obtained by applying the sampling method are presented in Table 4.5. For each instance, in addition to CI^- and CI^+, the columns Best, $DevCI^-$, $DevCI^+$ and Z^* are reported. Best corresponds to an upper bound, equal to the minimum value between the values of $\widehat{CVaR_\alpha}^*$ obtained by CPLEX and MS-VNS; Z^* corresponds to the subset of depots opened in the best solution; $DevCI^-$ and $DevCI^+$ correspond to the percentage deviation between Best and CI^- and CI^+, respectively, whose values are computed as $DevCI = 100\frac{(Best-CI)}{CI}$ (replacing CI by CI^- and CI^+ when it corresponds). For each method the columns $\widehat{CVaR_\alpha}^*$, $DevBest$, and time are reported. $\widehat{CVaR_\alpha}^*$ corresponds to the best solution value obtained for each instance after executing the sampling method considering N equal to 10, 20, and 30. $DevBest$ corresponds to the percentage deviation between $\widehat{CVaR_\alpha}^*$ and Best, computed as $DevBest = 100\frac{(\widehat{CVaR_\alpha}^* - Best)}{Best}$. The column time reports the global computing time after executing the sampling method with N equal to 10, 20 and 30.

The results indicate that for 5 out of the 16 instances analyzed the upper bound value found is within the CI. Therefore, we can further refine the CI of the 5 mentioned instances setting it equal to (CI^-, Best). The average value of $DevCI^-$ and $DevCI^+$ is 2.94% and 0.92%, respectively. CPLEX finds a better solution value than MS-VNS for 11 instances. The average value of $DevBest$ is equal to 0.29% and 0.56% for CPLEX and MS-VNS, respectively. The average global computing times required by CPLEX is of two orders of magnitude higher than the time required by MS-VNS.

The average computing times required by CPLEX and MS-VNS for executing each phase of the sampling method, considering N equal to 10, 20, and 30, are depicted in Figure 4.4. When using CPLEX, the first phase of the sampling method requires large computing times (26295, 178684, and 694027 seconds for N equal to 10, 20, and 30, respectively), while the second phase, which does not depend on the value of N, requires relatively short computing times (3595 seconds on average). The computing times required by the simheuristic are considerably smaller than those required by CPLEX. Furthermore, the differences in computing time between the first and second phase are not that large; the first phase requires 517, 1074, and 1619 seconds for N equal to 10, 20 and 30, respectively, while the second phase requires on average 519 seconds.

After analyzing the results it is possible to conclude that the proposed simheuristic provides

good approximations for the optimal solution values when it comes to tackle instances with continuous probability distributions, in relatively short computing times. On the other hand, the extremely large computing times required by CPLEX do not provide solution values considerably better than those provided by the simheuristic.

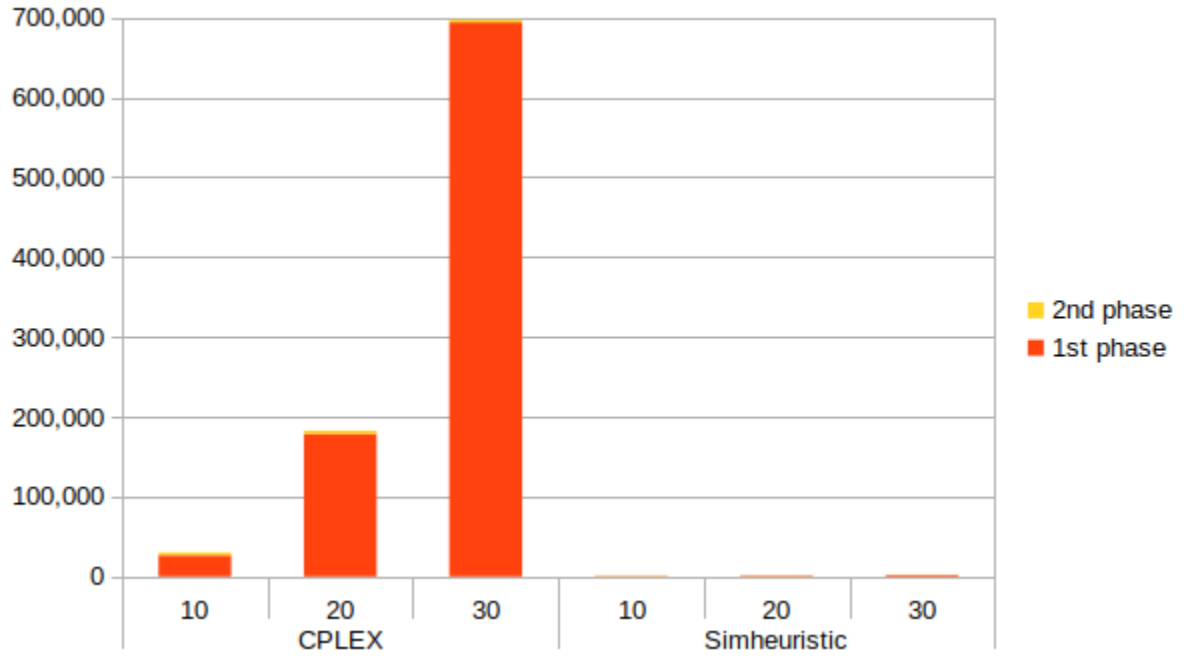


Figure 4.4: Average computing time (s) required by CPLEX and the MS-VNS heuristic for performing the sampling method when N is equal to 10, 20, and 30.

Table 4.5: Detailed results of the sampling method.

Instance		CI$^-$	CI$^+$	Best	DevCI$^-$	DevCI$^+$	Z^*	CPLEX			Simheuristic		
								$\widehat{CVaR_\alpha}^*$	DevBest	time	$\widehat{CVaR_\alpha}^*$	DevBest	time
10-5-1	LOGN	72.76	74.31	74.25	2.06	-0.08	2-4-5	**74.25**	0.00	2121887	75.20	1.28	5257
	UNIF	136.63	138.89	138.66	1.49	-0.17	1-2-4	**138.66**	0.00	299474	139.32	0.48	5204
10-5-1b	LOGN	133.53	136.60	139.38	4.38	2.03	1-3	**139.38**	0.00	462122	140.95	1.13	4436
	UNIF	266.96	271.37	273.04	2.28	0.61	1-3	**273.04**	0.00	368225	274.37	0.49	4397
10-5-2	LOGN	57.09	58.57	60.02	5.13	2.48	3-5	**60.02**	0.00	1173995	60.97	1.59	5046
	UNIF	109.02	110.70	112.75	3.42	1.85	4-5	**112.75**	0.00	327517	113.15	0.36	4873
10-5-2b	LOGN	73.40	75.46	75.60	2.99	0.18	1-4	**75.60**	0.00	282057	75.77	0.22	4339
	UNIF	139.34	142.00	141.50	1.55	-0.35	1-4	142.41	0.64	175364	**141.50**	0.00	3875
10-10-1	LOGN	56.31	57.44	59.16	5.07	3.01	2-6-7	59.68	0.88	2581919	**59.16**	0.00	5446
	UNIF	110.79	112.35	114.32	3.18	1.75	4-6-7	114.93	0.54	1092863	**114.32**	0.00	5409
10-10-1b	LOGN	100.27	102.62	103.96	3.68	1.31	9-10	**103.96**	0.00	754276	104.92	0.92	4578
	UNIF	199.65	202.58	203.67	2.01	0.53	9-10	204.90	0.61	473698	**203.67**	0.00	4575
10-10-2	LOGN	53.90	55.03	56.86	5.48	3.33	5-10	**56.86**	0.00	2518287	57.72	1.52	5245
	UNIF	104.97	106.93	107.26	2.18	0.31	5-10	**107.26**	0.00	1105198	107.80	0.51	5131
10-10-2b	LOGN	73.33	75.20	74.01	0.93	-1.58	1-4	75.44	1.94	536810	**74.01**	0.00	4509
	UNIF	139.23	141.79	141.01	1.28	-0.55	1-4	**141.01**	0.00	283003	141.62	0.43	3930
Global avg		114.20	116.36	117.21	2.94	0.92		**117.51**	**0.29**	909793	117.78	0.56	**4766**

Table 4.6: Summarized results for the LLRP benchmark instances.

Data set	#	#BKS0	#new BKS	BKS	Best	Avg	time	gap_B
Tuzun-Burke	36	1	13	3796.62	3822.30	3876.79	2152.8	0.84
Prodhon	30	6	5	1494.04	1504.20	1519.69	1040.6	1.14
Barreto	10	3	1	9378.70	9603.06	9755.00	322.0	1.90
All the instances	76	10	19	3622.19	3667.88	3719.80	1472.9	1.10

4.5.6 The deterministic (single-scenario) problem

In this section, the new heuristic introduced in Section 4.4.1 is compared with the currently published heuristic algorithms when it comes to tackling the deterministic LLRP. We performed experiments on 76 LLRP benchmark instances. All the currently published algorithms [47, 49, 50] but GBILS presented by [48] (5 runs) considered 30 runs for each instance. In order to present a fair comparison, for each instance we executed 30 runs of MS-VNS, and we set the parameter $tryVNS$ equal to 1. In that way for each run of each instance it is solved once.

In the Appendix, the tables 4.8–4.10 present the detailed results for each instance. For each instance we report the value of BKS, which is the updated best known solution value (considering the results obtained by MS-VNS). In addition, for each instance and each algorithm we present the following columns:

- Best: Best solution value obtained by the respective algorithm for each instance after the respective number of runs for each instance.

- Avg: Average solution value obtained by the respective algorithm for each instance after the respective number of runs for each instance.

- time: Global computing time in seconds. It is to note that the computing times reported for the MS-VNS correspond to the scaled global computing time using a "scaling factor" equal to 0.88, since the computer used in this work is slower than that used in [50]. The value of the "scaling factor" was calculated as the ratio between the single thread scorings of the two computers, which can be obtained in https://www.cpubenchmark.net/. The computing times associated with the competitors are these reported by [50], that were already scaled.

- gap_B: Percentage gap between the best solution value found by the respective algorithm (Best) and the value of BKS, computed as $gap_{BKS} = 100\frac{(Best-BKS)}{BKS}$.

Table 4.6 presents the aggregated results obtained by MS-VNS regarding the three benchmark data sets for the LLRP. The following columns were added to Table 4.6:

- #: Number of instances of each data set.

- #BKS_0: Number of instances for which MS-VNS is able to find the same solution value BKS_0, which is the currently published best known solution.

- #new BKS: Number of instances for which MS-VNS is able to find a solution value better than BKS_0.

Table 4.7: MS-VNS VS the currently published heuristic algorithms: defeats, draws, wins.

Competitor	Data set	L	D	W	Total
MA [47]	Tuzun-Burke	3	0	33	36
	Prodhon	3	1	26	30
	Barreto	3	1	6	10
	Total	9	2	65	76
GBILS [48]	Tuzun-Burke	-	-	-	0
	Prodhon	4	2	24	30
	Barreto	2	3	1	6
	Total	6	5	25	36
Best SA-VND [49]	Tuzun-Burke	13	0	23	36
	Prodhon	11	4	15	30
	Barreto	6	3	1	10
	Total	30	7	39	76
M-ILS [50]	Tuzun-Burke	22	1	13	36
	Prodhon	18	6	6	30
	Barreto	6	3	1	10
	Total	46	10	20	76

MS-VNS is able to find a solution value better than (19) or equal to (10) BKS_0 for 29 out of 76 instances, with an average value gap_B equal to 1.10%, and an average global computing time equal to 1472.9 s.

Table 4.7 presents the number of instances for which MS-VNS finds a solution value worse than (L), equal to (D), and better than (W) the respective competitor, for each data set. It is to note that Best SA-VND corresponds to the best solution value obtained among the three algorithms proposed in [49], for each instance. After considering the results reported in Tables 4.6 and 4.7, and the detailed results provided in the Apendix it is possible to conclude the following: MS-VNS globally outperforms MA and GBILS in terms of solution quality. In terms of computing time MS-VNS and MA have similar performances, while the global computing time of MS-VNS (30 runs) is larger than the one reported for GBILS (5 runs). If the average computing times (computing time required for a single run) are compared, both values are similar. MS-VNS generally provides solutions with similar quality with respect to the three SA-VND metaheuristics, requiring global computing times 3 times smaller than the SA-VND0, which is the fastest among the three algorithms. MS-VNS is not able to outperform M-ILS in terms of solution quality, nevertheless, it is able to find competitive results in shorter (almost 2 times) computing times.

4.6 Conclusions

Real-life problems in disaster operations management usually must deal with disruptions in the network that directly affect the travel times. It implies that solving a simple deterministic problem considering the expected values of the travel times may lead to sub-optimal solutions for the real problem. Motivated by that fact, a novel problem called LLRP-STT is studied in this chapter. The problem was formulated as a two-stage stochastic programming problem in which the location decisions were considered at the first stage, and the routing decisions

at the second stage. Furthermore, due to the nature of the problem, a risk-averse decision maker was considered, with the CVaR as risk measure. A multi-layer formulation for modeling the problem (solved by CPLEX), and a heuristic algorithm called MS-VNS were proposed. Extensive computational experiments proved the effectiveness of the proposed heuristic, which was also competitive w.r.t. the currently published algorithms for the deterministic LLRP. Several insights were provided. With numerical experiments we highlighted the importance of considering a stochastic approach instead of solving a deterministic version of the problem. Also, how the solution changes when a risk-neutral attitude is considered instead of a risk-averse one. A sampling method was proposed for tackling instances with an infinite number of scenarios.

Several research avenues emerge from this work. First, in addition to the travel times, other sources of uncertainty can be considered, e.g., demand or capacities, which imply considering new recourse functions. Second, other LRP under uncertainty can be studied. These directions can be easily identified by the reader in the literature review section presented in this chapter, nevertheless, in our opinion, some interesting problems are those related to the application of emerging technologies such as electric vehicles [24, 25] and drones [26]. Naturally, due to the complexity of LRPs under uncertainty, heuristic methods can be developed for tackling large-size instances of such problems.

Appendix: Detailed results for the deterministic LLRP

Table 4.8: Detailed results for the LLRP Tuzun-Burke data set.

Instance	C	D	K	BKS	MA				SA-VND0				SA-VND1				SA-VND2				M-ILS				MS-VNS			
					Best	Avg	time	gap%	Best	Avg	time	gap%	Best	Avg	time	gap%	Best	Avg	time	gap%	Best	Avg	time	gap%	Best	Avg	time	gap%
111112	100	10	11	3834.91	4017.90	4270.10	1200.0	4.77	3862.86	3971.50	1912.8	0.73	3892.97	3972.93	2390.6	1.51	3882.76	3964.12	2801.2	1.25	**3834.91**	3890.13	2149.2	0.00	3856.67	3921.40	736.6	0.57
111122	100	20	11	3612.36	3719.10	4087.50	1200.0	2.95	**3612.36**	3694.70	1934.6	0.00	3633.60	3712.64	2419.2	0.59	3623.69	3687.85	2860.6	0.31	3659.46	3695.69	2130.6	1.30	3684.02	3730.78	451.3	1.98
111212	100	10	10	3919.74	4264.40	4563.50	1245.0	8.79	3960.24	4038.24	1805.4	1.03	3988.11	4067.91	2335.5	1.74	3938.88	4066.13	2692.0	0.49	**3919.74**	4000.94	2455.1	0.00	3943.04	4006.67	831.7	0.59
111222	100	20	11	4065.04	4278.30	4557.30	1242.0	5.25	4086.74	4140.33	1909.6	0.53	4077.87	4147.90	2376.2	0.32	4068.34	4138.78	2785.2	0.08	**4065.04**	4139.64	2312.7	0.00	4076.66	4163.50	520.7	0.29
112112	100	10	11	2726.41	2795.60	3049.70	1242.0	2.54	2739.16	2755.53	2281.4	0.47	2740.21	2759.43	2908.3	0.51	2741.82	2756.07	3271.1	0.57	**2726.41**	2749.56	1649.9	0.00	2742.42	2756.85	576.4	0.59
112122	100	20	11	2057.30	2097.90	2702.60	1329.0	1.97	2060.29	2072.57	2211.6	0.15	2057.45	2078.03	2797.4	0.01	2060.80	2071.85	3205.3	0.17	**2057.30**	2063.89	1193.4	0.00	2131.25	2141.52	743.3	3.59
112212	100	10	12	1394.65	1442.20	1604.90	1455.0	3.41	1402.97	1416.20	2349.9	0.60	1403.57	1416.72	3050.0	0.64	1397.39	1414.98	3335.3	0.20	**1394.65**	1411.85	1490.7	0.00	1442.83	1483.98	779.2	3.75
112222	100	20	11	1621.40	1659.00	2084.10	1314.0	2.32	1623.69	1633.00	2522.2	0.14	**1621.40**	1633.94	3137.8	0.00	1626.86	1633.89	3562.2	0.34	1623.39	1630.36	1652.2	0.12	1624.64	1636.58	585.9	0.20
113112	100	10	11	**2826.52**	2897.70	3162.60	1170.0	2.52	2837.51	2852.63	2043.2	0.39	2835.76	2853.57	2735.6	0.33	2839.50	2857.50	3014.0	0.46	2828.24	2841.93	2091.6	0.00	**2826.52**	2833.19	782.2	0.00
113122	100	20	11	2772.98	2912.00	3330.00	1248.0	5.01	2776.38	2782.52	2063.4	0.12	2774.36	2784.42	2762.2	0.05	2776.38	2782.07	2998.4	0.12	**2772.98**	2797.08	2753.4	0.00	**2772.98**	2775.43	770.3	0.00
113212	100	10	12	1815.62	1832.10	1901.00	1458.0	0.91	1817.00	1823.15	2326.4	0.08	**1815.62**	1822.81	2922.2	0.00	**1815.62**	1822.76	3326.7	0.00	1817.00	1835.80	2023.4	0.08	1980.82	2026.77	788.5	9.10
113222	100	20	11	**1874.42**	2037.50	2360.60	1239.0	8.70	1876.14	1888.46	2117.2	0.09	1879.63	1890.96	2816.5	0.28	1876.93	1888.90	3063.5	0.13	1876.58	1885.31	1940.0	0.02	**1874.42**	1905.90	950.8	0.00
131112	150	10	16	5410.96	5863.80	6160.70	1797.0	8.37	5473.17	5582.94	4060.2	1.15	5464.21	5570.12	6108.7	0.98	5448.86	5576.01	6409.4	0.70	**5410.96**	5478.43	3854.6	0.00	5439.69	5507.89	1779.3	0.53
131122	150	20	16	**4899.08**	5310.30	5649.30	1746.0	8.39	4993.36	5142.06	4462.0	1.92	5009.26	5143.19	6549.2	2.25	4974.28	5105.72	6876.6	1.53	4926.87	5053.13	3970.3	0.00	**4899.08**	5063.17	1775.3	0.00
131212	150	10	16	5528.85	5967.60	6234.20	1746.0	7.94	5679.70	5787.34	4588.6	2.73	5606.31	5785.18	6765.0	1.40	5653.20	5771.63	7051.5	2.25	**5528.85**	5639.90	4007.7	0.00	5613.38	5706.22	1965.6	1.53
131222	150	20	16	5060.71	5261.00	5610.40	1767.0	3.96	5141.89	5284.45	4580.2	1.60	5126.95	5277.65	6627.3	1.31	5134.39	5296.62	7093.9	1.46	**5060.71**	5107.53	4093.6	0.00	5104.39	5135.02	1648.7	0.86
132112	150	10	16	**3845.59**	4026.70	4246.20	1746.0	4.71	3868.88	3895.91	5963.0	0.61	3883.40	3899.41	8542.7	0.98	3874.44	3894.29	8771.5	0.75	3850.90	3881.70	4034.9	0.00	**3845.59**	3881.63	1495.6	0.00
132122	150	20	16	**3725.06**	3874.00	4185.50	1785.0	4.00	3740.10	3795.93	5155.0	0.40	3752.76	3787.72	7796.2	0.74	3755.51	3797.33	7768.9	0.82	3734.83	3762.41	3762.9	0.00	**3725.06**	3784.14	1479.5	0.00
132212	150	10	17	2835.66	2906.00	3129.90	1779.0	2.48	2842.10	2857.43	5839.7	0.23	2837.84	2860.70	8792.6	0.08	2843.18	2857.02	8714.5	0.27	**2835.66**	2850.38	3945.7	0.00	2877.26	2902.22	1906.3	1.47
132222	150	20	17	1651.91	1784.80	2152.30	1773.0	8.04	1660.89	1691.97	6340.4	0.54	1672.86	1697.45	9145.6	1.27	1677.95	1695.15	9288.8	1.58	**1651.91**	1676.40	2824.2	0.00	1663.67	1716.13	1861.6	0.71
133112	150	10	16	**4572.43**	5034.00	5465.90	1737.0	10.09	4588.37	4619.91	4670.1	0.35	4598.23	4630.69	7313.4	0.56	4596.35	4625.72	7130.4	0.52	4578.87	4612.46	3339.0	0.00	**4572.43**	4631.26	2135.0	0.00
133122	150	20	16	3211.98	3474.00	3849.10	1767.0	8.16	3223.44	3259.45	5317.0	0.36	3225.56	3271.04	8052.6	0.42	3223.40	3248.42	8012.0	0.36	**3211.98**	3236.27	3813.4	0.00	3214.08	3243.38	1925.9	0.07
133212	150	10	17	**2898.19**	3008.00	3284.50	1770.0	3.79	2911.58	2938.05	5887.3	0.46	2911.35	2938.01	8572.0	0.45	2906.96	2935.00	8754.9	0.30	2903.36	2918.06	3116.3	0.00	**2898.19**	2907.65	1914.8	0.00
133222	150	20	17	2485.07	2617.40	3016.90	1755.0	5.33	2502.97	2550.01	5692.6	0.72	2502.68	2558.34	8274.9	0.71	2501.03	2531.56	8664.8	0.64	**2485.07**	2496.12	3020.2	0.00	2487.73	2524.40	1756.1	0.11
121112	200	10	21	6572.43	7008.70	7371.10	2502.0	6.64	6608.45	6821.20	8392.7	0.55	6621.55	6881.38	13919.9	0.75	6683.24	6848.60	14006.4	1.69	**6572.43**	6632.64	6096.4	0.00	6606.70	6667.00	3661.9	0.52
121122	200	20	22	5612.03	6039.60	6501.60	2490.0	7.62	5730.53	5954.23	9991.8	2.11	5788.72	5966.38	15672.5	3.15	5784.71	5952.63	16313.2	3.08	**5612.03**	5668.75	5948.1	0.00	5632.10	5721.38	4156.7	0.36
121212	200	10	21	**6373.04**	6744.30	7318.60	2559.0	5.83	6503.36	6613.84	8391.2	2.04	6429.62	6608.92	13778.5	0.89	6502.73	6610.46	13910.1	2.03	6409.74	6449.92	5906.0	0.00	**6373.04**	6461.08	3447.4	0.00
121222	200	20	21	6383.30	6828.50	7567.20	2553.0	6.97	6551.73	6759.22	8391.4	2.64	6562.11	6796.71	14023.1	2.80	6648.20	6776.44	13851.2	4.15	**6383.30**	6526.13	6610.6	0.00	6469.81	6617.36	3379.5	1.36
122112	200	10	21	6111.52	6643.80	7106.90	2571.0	8.71	6154.64	6255.10	9463.2	0.71	6184.70	6280.31	18340.7	1.20	6168.32	6268.36	14679.6	0.93	**6111.52**	6208.87	9151.9	0.00	6209.87	6300.28	4203.3	**0.41**
122122	200	20	21	3725.07	4012.90	4915.70	2547.0	7.73	3757.37	3782.47	10555.0	0.87	3757.27	3792.68	17314.9	0.86	3744.74	3785.46	16701.0	0.53	**3725.07**	3756.16	4976.8	0.00	3747.27	3801.35	3736.9	0.60
122212	200	10	21	**4024.03**	4227.50	4448.10	2535.0	5.06	4046.81	4075.78	9845.3	0.57	4046.42	4078.60	17116.3	0.56	4043.53	4077.42	15490.2	0.48	4025.13	4042.78	3871.8	0.00	**4024.03**	4050.57	2765.5	0.00
122222	200	20	22	**2043.45**	2127.90	2345.50	2511.0	4.13	2054.31	2083.56	11266.8	0.53	2052.22	2084.10	18533.2	0.43	2052.16	2079.68	17652.2	0.43	2049.68	2056.08	3826.1	0.00	**2043.45**	2061.10	3602.5	0.00
123112	200	10	22	**4861.59**	5099.00	5527.40	2559.0	4.88	4916.97	5024.07	10869.1	1.14	4967.11	5047.87	17275.1	2.17	4940.81	5029.64	17259.7	1.63	4868.90	4921.23	6253.2	0.00	**4861.59**	4913.01	4690.6	0.00
123122	200	20	22	**4670.83**	5188.70	5862.90	2544.0	11.09	4725.91	4771.90	10590.0	1.18	4707.61	4785.89	16689.1	0.79	4719.90	4777.07	17021.6	1.05	4675.19	4703.17	6099.4	0.00	**4670.83**	4724.53	4336.9	0.00
123212	200	10	22	5135.21	5363.00	5678.50	2544.0	4.44	5170.77	5218.43	10206.0	0.69	5178.03	5225.04	17737.1	0.83	5195.48	5247.18	16205.9	1.17	**5135.21**	5174.74	4033.7	0.00	5148.21	5201.68	3800.3	0.25
123222	200	20	22	**2519.00**	2657.50	3917.60	2577.0	5.50	2567.20	2629.46	10676.9	1.91	2555.18	2633.86	18065.4	1.44	2553.21	2606.54	17096.2	1.36	2522.89	2551.90	3961.3	0.00	**2519.00**	2659.29	5570.5	0.00
Global avg.				3796.62	4028.41	4422.78	**1861.2**	5.64	3835.27	3901.77	5740.9	0.84	3837.85	3909.51	8990.5	0.92	3840.99	3902.19	8934.4	0.94	**3801.30**	**3842.98**	3715.6	**0.04**	3822.30	3876.79	2152.8	**1.84**

Table 4.9: Detailed results for the LLRP Prodhon data set.

Instance	K	BKS	MA				GBILS			SA-VND0				SA-VND1				SA-VND2				M-ILS				MS-VNS			
			Best	Avg	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B
20-5-1	5	330.00	337.30	378.00	387.0	2.21	**330.00**	1.0	0.00	**330.00**	**330.00**	125.2	0.00	**330.00**	**330.00**	117.8	0.00	**330.00**	**330.00**	166.4	0.00	**330.00**	**330.00**	167.8	0.00	**330.00**	330.00	15.5	0.00
20-5-1b	3	608.06	**608.06**	636.80	375.0	0.00	608.06	0.5	0.00	**608.06**	**608.06**	145.7	0.00	**608.06**	**608.06**	122.3	0.00	**608.06**	**608.06**	244.9	0.00	615.66	615.66	113.6	1.25	646.65	646.65	10.3	6.35
20-5-2	5	301.97	304.80	354.40	381.0	0.94	**301.97**	0.7	0.00	**301.97**	**301.97**	106.3	0.00	**301.97**	**301.97**	98.5	0.00	**301.97**	**301.97**	150.5	0.00	**301.97**	**301.97**	161.8	0.00	316.33	316.33	11.5	4.76
20-5-2b	3	486.55	**486.55**	511.20	381.0	0.00	**486.55**	0.8	0.00	**486.55**	**486.55**	158.7	0.00	**486.55**	**486.55**	132.8	0.00	**486.55**	**486.55**	266.6	0.00	**486.55**	**486.55**	114.3	0.00	**486.55**	486.55	9.5	0.00
50-5-1	12	843.94	859.90	917.30	546.0	1.89	846.88	153.5	0.35	846.17	849.77	709.2	0.27	846.52	850.10	815.3	0.31	**843.94**	850.40	859.5	0.00	**843.94**	845.80	589.0	0.00	844.63	854.60	163.9	0.08
50-5-1b	6	1293.46	1330.20	1379.80	522.0	2.84	1293.93	65.5	0.04	**1293.46**	1293.71	619.7	0.00	**1293.46**	**1293.54**	602.2	0.00	**1293.46**	1293.55	933.8	0.00	**1293.46**	1293.95	682.8	0.00	**1293.46**	1296.25	95.9	0.00
50-5-2	12	684.13	723.40	786.20	552.0	5.74	691.67	117.4	1.10	**684.13**	694.42	624.0	0.00	**684.13**	692.43	756.3	0.00	**684.13**	694.69	752.8	0.00	**684.13**	690.41	930.0	0.00	**684.13**	687.14	158.6	0.00
50-5-2b	6	953.25	965.70	1009.40	573.0	1.31	954.88	68.4	0.17	**953.25**	953.50	534.8	0.00	**953.25**	953.35	534.2	0.00	**953.25**	953.33	799.3	0.00	**953.25**	953.67	563.9	0.00	953.31	964.61	138.1	0.01
50-5-2BIS	12	945.45	955.20	981.50	537.0	1.03	952.55	120.7	0.75	949.13	950.77	883.4	0.39	949.57	951.13	1081.1	0.44	950.12	950.93	1024.7	0.49	**945.45**	945.77	1493.0	0.00	**945.45**	965.10	175.9	0.00
50-5-2bBIS	6	803.90	811.80	884.90	534.0	0.98	**803.90**	96.6	0.00	**803.90**	**803.90**	626.9	0.00	**803.90**	**803.90**	649.9	0.00	**803.90**	**803.90**	883.7	0.00	**803.90**	**803.90**	708.2	0.00	932.69	936.29	112.0	16.02
50-5-3	12	831.57	848.10	928.90	612.0	1.99	832.15	119.2	0.07	831.97	835.10	712.3	0.05	833.01	834.91	810.3	0.17	833.59	835.17	863.8	0.24	**831.57**	834.22	1084.5	0.00	831.83	838.35	123.4	0.03
50-5-3b	6	1101.57	1163.90	1198.80	531.0	5.66	1106.57	69.4	0.45	**1101.57**	1103.15	538.4	0.00	**1101.57**	1103.95	541.3	0.00	**1101.57**	1103.53	794.5	0.00	**1101.57**	1102.76	467.3	0.00	1140.23	1140.67	70.0	3.51
100-5-1	24	2000.80	2030.90	2044.30	891.0	1.50	2035.60	64.8	1.74	2004.33	2023.35	3039.4	0.18	2010.49	2023.78	4791.3	0.48	2008.95	2026.45	3777.3	0.41	**2000.80**	2012.93	2474.4	0.00	2007.31	2024.59	914.3	0.33
100-5-1b	12	2311.21	2374.90	2507.80	744.0	2.76	2357.87	102.8	2.02	2311.84	2336.64	2221.6	0.03	2312.53	2337.27	2871.7	0.06	2313.70	2342.80	3126.5	0.11	**2311.21**	2346.56	2164.6	0.00	2315.96	2361.28	735.9	0.21
100-5-2	24	**1126.93**	1226.10	1500.90	852.0	8.80	1144.70	81.1	1.58	1132.36	1135.99	2760.8	0.48	1129.83	1135.49	3976.1	0.26	1131.28	1135.70	3582.7	0.39	1128.43	1133.53	2718.1	0.13	**1126.93**	1135.78	668.3	0.00
100-5-2b	11	**1506.79**	1622.90	1701.00	855.0	7.71	1567.44	96.8	4.03	1507.88	1517.11	2530.4	0.07	1510.57	1519.04	3141.3	0.25	1510.57	1519.56	3448.8	0.25	1507.88	1512.55	1540.1	0.07	**1506.79**	1528.38	595.7	0.00
100-5-3	24	1572.61	1710.40	1726.20	903.0	8.76	1596.77	49.2	1.54	1581.93	1587.20	2784.3	0.59	1581.93	1586.49	4072.3	0.59	1581.93	1587.20	3595.7	0.59	**1572.61**	1581.49	2943.9	0.00	1576.45	1585.75	663.3	0.24
100-5-3b	11	1933.00	2054.80	2190.50	870.0	6.30	2032.13	114.9	5.13	1933.70	1950.89	2315.8	0.04	1935.70	1953.85	2932.6	0.14	**1933.00**	1954.97	3247.9	0.00	1934.93	1954.27	2140.7	0.10	1934.22	1946.30	576.2	0.06
100-10-1	26	1458.80	1524.10	1589.90	1215.0	4.48	1481.56	80.3	1.56	1472.85	1511.00	2994.7	0.96	1470.71	1503.92	4143.4	0.82	1478.01	1510.98	3908.8	1.32	**1458.80**	1465.01	2593.2	0.00	1461.09	1475.82	726.2	0.16
100-10-1b	12	1894.92	1960.70	2138.60	1185.0	3.47	1984.91	100.8	4.75	1901.27	1953.96	2120.3	0.34	1915.77	1972.26	2723.1	1.10	1908.63	1963.53	2999.3	0.72	**1894.92**	1916.90	2211.6	0.00	1895.28	1924.43	524.9	0.02
100-10-2	24	1137.59	1175.00	1236.30	1326.0	3.29	1287.50	75.8	13.18	1143.30	1152.81	2899.7	0.50	1142.31	1155.47	4132.2	0.41	1145.93	1155.29	3739.8	0.73	**1137.59**	1145.91	3167.4	0.00	1144.57	1153.32	862.1	0.61
100-10-2b	11	**1555.71**	1625.80	1724.20	1230.0	4.51	1645.07	139.1	5.74	1566.48	1585.67	2208.4	0.69	1566.48	1588.24	2889.5	0.69	1563.99	1578.90	3115.3	0.53	1561.40	1570.81	2425.8	0.37	**1555.71**	1577.88	728.7	0.00
100-10-3	25	1204.64	1246.80	1288.30	1332.0	3.50	1216.20	57.2	0.96	1209.20	1221.52	3145.2	0.38	1209.86	1225.98	4290.2	0.43	1211.49	1224.72	4060.6	0.57	**1204.64**	1209.10	1919.8	0.00	**1204.64**	1209.31	838.2	0.00
100-10-3b	11	**1651.06**	1799.00	1890.20	1287.0	8.96	1745.05	70.1	5.69	1662.43	1705.63	2146.0	0.69	1665.69	1706.68	2831.1	0.89	1670.17	1707.11	3058.5	1.16	1653.83	1673.19	2334.1	0.17	**1651.06**	1680.56	760.1	0.00
200-10-1	49	**2777.35**	2920.70	3092.40	3414.0	5.16	2861.85	278.2	3.04	2798.58	2854.10	16846.7	0.76	2797.86	2863.27	26673.1	0.74	2803.57	2860.64	23819.4	0.94	2780.03	2788.66	6664.8	0.10	**2777.35**	2792.17	3217.8	0.00
200-10-1b	22	3290.73	3532.20	3809.30	3240.0	7.34	3557.96	304.1	8.12	3368.71	3477.07	11341.6	2.37	3355.70	3478.91	17983.6	1.97	3327.08	3452.84	17334.9	1.10	**3290.73**	3334.98	6272.8	0.00	3302.42	3382.20	3191.5	0.36
200-10-2	49	1972.33	2064.20	2153.30	3411.0	4.66	1997.01	342.5	1.25	1984.96	2001.97	17482.5	0.64	1986.55	2004.51	29590.3	0.72	1988.31	2002.48	24959.6	0.81	**1972.33**	1981.41	5560.2	0.00	1973.92	1983.77	3986.0	0.08
200-10-2b	23	2325.43	2516.40	2684.50	3141.0	8.21	2473.24	273.7	6.36	2336.11	2379.01	12347.0	0.46	2355.15	2378.21	19720.6	1.28	2368.88	2380.09	18699.6	1.87	**2325.43**	2358.02	5160.5	0.00	2327.37	2375.83	3316.8	0.08
200-10-3	48	2727.15	2805.90	3066.1	3084.0	2.89	2783.20	323.8	2.06	2741.16	2758.09	14786.4	0.51	2744.67	2757.42	27900.5	0.64	2751.23	2763.79	21623.8	0.88	**2727.15**	2736.49	5479.3	0.00	2732.22	2740.97	4217.4	0.19
200-10-3b	22	3190.34	3347.00	3454.60	3267.0	4.91	3413.34	281.4	6.99	3242.18	3274.58	9511.1	1.62	3233.89	3267.81	16534.2	1.37	3225.75	3273.68	14933.5	1.11	**3190.34**	3218.10	3924.1	0.00	3227.38	3249.94	3608.7	1.16
Global avg.		1494.04	1564.42	1610.33	1272.6	4.06	1546.35	**121.7**	2.62	1502.98	1521.25	3975.5	0.40	1503.92	1522.28	6248.6	0.46	1503.77	1521.76	5692.4	0.47	**1494.82**	**1504.82**	2292.4	**0.07**	1504.20	1519.69	1040.6	1.14

Table 4.10: Detailed results for the LLRP Barreto data set

Instance	K	BKS	MA				GBILS			SA-VND0				SA-VND1				SA-VND2				M-ILS				MS-VNS			
			Best	Avg	time	gap_B	Best	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B	Best	Avg	time	gap_B
Christ-50-5	6	1661.64	1690.80	1782.40	591.0	1.75	–	–	–	**1661.64**	1662.07	541.5	0.00	**1661.64**	1662.13	528.7	0.00	**1661.64**	1662.35	813.7	0.00	**1661.64**	1669.05	614.2	0.00	1711.04	1720.18	87.4	2.97
Christ-75-10	9	**2362.48**	2590.30	2689.80	873.0	9.64	–	–	–	2403.79	2459.03	1159.1	1.75	2383.04	2457.45	1384.4	0.87	2408.92	2454.95	1588.4	1.97	2370.73	2409.75	1612.4	0.35	**2362.48**	2473.49	373.2	0.00
Christ-100-10	8	3791.98	4058.20	4194.90	1023.0	7.02	3984.05	451.9	5.07	**3791.98**	3831.18	1958.7	0.00	3806.39	3838.89	2293.6	0.38	3795.15	3825.37	2876.3	0.08	3803.50	3845.85	2110.8	0.30	3812.56	3841.74	536.3	0.54
Gaskell-21-5	4	653.48	658.40	741.10	441.0	0.75	**653.48**	0.9	0.00	**653.48**	**653.48**	116.4	0.00	**653.48**	**653.48**	102.2	0.00	**653.48**	**653.48**	169.5	0.00	**653.48**	**653.48**	143.0	0.00	**653.48**	653.48	15.6	0.00
Gaskell-29-5	4	1199.33	1224.50	1296.30	468.0	2.10	**1199.33**	5.2	0.00	**1199.33**	**1199.33**	311.0	0.00	**1199.33**	**1199.33**	255.4	0.00	**1199.33**	**1199.33**	485.9	0.00	**1199.33**	**1199.33**	283.2	0.00	1249.88	1249.88	26.6	4.21
Gaskell-32-5b	3	1552.84	1571.00	1668.40	483.0	1.17	**1552.84**	4.9	0.00	**1552.84**	1553.29	417.8	0.00	**1552.84**	1553.29	337.6	0.00	**1552.84**	1553.29	650.7	0.00	**1552.84**	1556.58	284.4	0.00	1630.04	1630.17	23.4	4.97
Gaskell-36-5	4	1627.17	1642.40	1647.00	522.0	0.94	**1627.17**	3.2	0.00	**1627.17**	**1627.17**	308.3	0.00	**1627.17**	**1627.17**	274.0	0.00	**1627.17**	**1627.17**	465.8	0.00	**1627.17**	1628.12	369.1	0.00	**1627.17**	1634.92	42.3	0.00
Min-27-5	4	5387.55	**5387.55**	5697.00	834.0	0.00	**5387.55**	84.0	0.00	**5387.55**	**5387.55**	176.3	0.00	**5387.55**	**5387.55**	147.4	0.00	**5387.55**	**5387.55**	267.0	0.00	**5387.55**	**5387.55**	134.0	0.00	**5387.55**	5387.55	25.3	0.00
Min-134-8	11	21752.00	23387.00	26012.50	2220.0	7.52	–	–	–	21852.40	22307.28	2225.7	0.46	21910.50	22309.20	2624.0	0.73	21881.80	22278.05	3250.7	0.60	**21752.00**	22442.41	2127.0	0.00	22679.40	22877.83	1306.1	4.26
Or-117-14	7	53798.50	56209.00	61396.20	1545.0	4.48	–	–	–	**53798.50**	54866.72	1263.1	0.00	53859.10	54805.88	1265.5	0.11	**53798.50**	54905.77	1958.4	0.00	54413.90	56794.12	1598.2	1.14	54917.00	56080.80	783.1	2.08
Global avg.		9378.70	9841.92	10712.56	900.0	3.54	–	–	–	**9392.87**	9554.71	847.8	0.22	9404.10	**9549.44**	921.3	0.21	9396.64	9554.73	1252.6	0.26	9442.21	9758.62	927.6	**0.18**	9603.06	9755.00	**322.5**	1.90
Global avg GBILS		2368.72	2423.68	2540.78	628.5	2.00	2400.74	**91.7**	0.84	**2368.72**	2375.33	548.1	**0.00**	2371.13	2376.62	568.4	0.06	2369.25	**2374.36**	819.2	0.01	2370.64	2378.48	554.1	0.05	2393.45	2399.62	111.1	1.62

Bibliography

[1] C. Prodhon and C. Prins, "A survey of recent research on location-routing problems," European Journal of Operational Research, vol. 238, pp. 1–17, 2014.

[2] R. Cuda, G. Guastaroba, and M. G. Speranza, "A survey on two-echelon routing problems," Computers & Operations Research, vol. 55, pp. 185–199, 2015.

[3] M. Drexl and M. Schneider, "A survey of variants and extensions of the location-routing problem," European Journal of Operational Research, vol. 241, pp. 283–308, 2015.

[4] M. Schneider and M. Drexl, "A survey of the standard location-routing problem," Annals of Operations Research, vol. 259, pp. 389–414, 2017.

[5] S. T. W. Mara, R. Kuo, and A. M. S. Asih, "Location-routing problem: a classification of recent research," International Transactions in Operational Research, vol. 28, pp. 2941–2983, 2021.

[6] I. Correia and F. Saldanha-da-Gama, "Facility location under uncertainty," in Location Science (G. Laporte, S. Nickel, and F. Saldanha-da-Gama, eds.), ch. 8, pp. 185–213, Springer, 2019.

[7] M. Gendreau, O. Jabali, and W. Rei, "Stochastic vehicle routing problems," in Vehicle Routing: Problems, Methods, and Applications (P. Toth and D. Vigo, eds.), ch. 11, pp. 213–239, SIAM, Second Edition, 2014.

[8] J. Oyola, H. Arntzen, and D. L. Woodruff, "The stochastic vehicle routing problem, a literature review, part II: solution methods," EURO Journal on Transportation and Logistics, vol. 6, pp. 349–388, 2017.

[9] J. Oyola, H. Arntzen, and D. L. Woodruff, "The stochastic vehicle routing problem, a literature review, part I: models," EURO Journal on Transportation and Logistics, vol. 7, pp. 193–221, 2018.

[10] K. Corona-Gutiérrez, S. Nucamendi-Guillén, and E. Lalla-Ruiz, "Vehicle routing with cumulative objectives: A state of the art and analysis," Computers & Industrial Engineering, vol. 159, p. 108054, 2022.

[11] G. Laporte, F. Louveaux, and H. Mercure, "Models and exact solutions for a class of stochastic location-routing problems," European Journal of Operational Research, vol. 39, pp. 71–78, 1989.

[12] M. Albareda-Sambola, E. Fernández, and G. Laporte, "Heuristic and lower bound for a stochastic location-routing problem," European Journal of Operational Research, vol. 179, pp. 940–955, 2007.

[13] Y. Marinakis, "An improved particle swarm optimization algorithm for the capacitated location routing problem and for the location routing problem with stochastic demands," Applied Soft Computing, vol. 37, pp. 680–701, 2015.

[14] C. L. Quintero-Araujo, D. Guimarans, and A. A. Juan, "A simheuristic algorithm for the capacitated location routing problem with stochastic demands," Journal of Simulation, vol. 15, no. 3, pp. 217–234, 2021.

[15] Y. Z. Mehrjerdi and A. Nadizadeh, "Using greedy clustering method to solve capacitated location-routing problem with fuzzy demands," European Journal of Operational Research, vol. 229, pp. 75–84, 2013.

[16] A. Nadizadeh and H. H. Nasab, "Solving the dynamic capacitated location-routing problem with fuzzy demands by hybrid heuristic algorithm," European Journal of Operational Research, vol. 238, pp. 458–470, 2014.

[17] M. H. F. Zarandi, A. Hemmati, S. Davari, and I. B. Turksen, "Capacitated location-routing problem with time windows under uncertainty," Knowledge-Based Systems, vol. 37, pp. 480–489, 2013.

[18] E. Pekel and S. S. Kara, "Solving fuzzy capacitated location routing problem using hybrid variable neighborhood search and evolutionary local search," Applied Soft Computing, vol. 83, p. 105665, 2019.

[19] S. Zhong, R. Cheng, Y. Jiang, Z. Wang, A. Larsen, and O. A. Nielsen, "Risk-averse optimization of disaster relief facility location and vehicle routing under stochastic demand," Transportation Research Part E: Logistics and Transportation Review, vol. 141, p. 102015, 2020.

[20] A. Ahmadi-Javid and A. H. Seddighi, "A location-routing problem with disruption risk," Transportation Research Part E: Logistics and Transportation Review, vol. 53, pp. 63–82, 2013.

[21] W. Xie, Y. Ouyang, and S. C. Wong, "Reliable location-routing design under probabilistic facility disruptions," Transportation Science, vol. 50, pp. 1128–1138, 2016.

[22] Y. Zhang, M. Qi, W.-H. Lin, and L. Miao, "A metaheuristic approach to the reliable location routing problem under disruptions," Transportation Research Part E: Logistics and Transportation Review, vol. 83, pp. 90–110, 2015.

[23] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," SIAM Journal on Optimization, vol. 12, pp. 479–502, 2002.

[24] M. Schiffer and G. Walther, "Strategic planning of electric logistics fleet networks: A robust location-routing approach," Omega, vol. 80, pp. 31–42, 2018.

[25] S. Zhang, M. Chen, and W. Zhang, "A novel location-routing problem in electric vehicle transportation with stochastic demands," Journal of Cleaner Production, vol. 221, pp. 567–581, 2019.

[26] M. E. Bruni, S. Khodaparasti, and G. Perboli, "The drone latency location routing problem under uncertainty," Transportation Research Part C: Emerging Technologies, vol. 156, p. 104322, 2023.

[27] S.-H. Huang, "Solving the multi-compartment capacitated location routing problem with pickup–delivery routes and stochastic demands," Computers & Industrial Engineering, vol. 87, pp. 104–113, 2015.

[28] Y. Chan, W. B. Carter, and M. D. Burnes, "A multiple-depot, multiple-vehicle, location-routing problem with stochastically processed demands," Computers & Operations Research, vol. 28, pp. 803–826, 2001.

[29] S. T. Hassanpour, G. Y. Ke, J. Zhao, and D. M. Tulett, "Infectious waste management during a pandemic: A stochastic location-routing problem with chance-constrained time windows," Computers & Industrial Engineering, vol. 177, p. 109066, 2023.

[30] I. B. Mohamed, W. Klibi, R. Sadykov, H. Şen, and F. Vanderbeck, "The two-echelon stochastic multi-period capacitated location-routing problem," European Journal of Operational Research, vol. 306, pp. 645–667, 2023.

[31] S. J. Rennemo, K. F. Rø, L. M. Hvattum, and G. Tirado, "A three-stage stochastic facility routing model for disaster response planning," Transportation Research Part E: Logistics and Transportation Review, vol. 62, pp. 116–135, 2014.

[32] S. Fazayeli, A. Eydi, and I. N. Kamalabadi, "Location-routing problem in multimodal transportation network with time windows and fuzzy demands: Presenting a two-part genetic algorithm," Computers & Industrial Engineering, vol. 119, pp. 233–246, 2018.

[33] S. Mohammadi, S. A. Darestani, B. Vahdani, and A. Alinezhad, "A robust neutrosophic fuzzy-based approach to integrate reliable facility location and routing decisions for disaster relief under fairness and aftershocks concerns," Computers & Industrial Engineering, vol. 148, p. 106734, 2020.

[34] A. M. Caunhye, Y. Zhang, M. Li, and X. Nie, "A location-routing model for prepositioning and distributing emergency supplies," Transportation Research Part E: Logistics and Transportation Review, vol. 90, pp. 161–176, 2016.

[35] Z.-J. M. Shen and L. Qi, "Incorporating inventory and routing costs in strategic location models," European Journal of Operational Research, vol. 179, pp. 372–389, 2007.

[36] A. A. Javid and N. Azad, "Incorporating location, routing and inventory decisions in supply chain network design," Transportation Research Part E: Logistics and Transportation Review, vol. 46, pp. 582–597, 2010.

[37] N. Nekooghadirli, R. Tavakkoli-Moghaddam, V. Ghezavati, and S. Javanmard, "Solving a new bi-objective location-routing-inventory problem in a distribution network by meta-heuristics," Computers & Industrial Engineering, vol. 76, pp. 204–221, 2014.

[38] M. Zhalechian, R. Tavakkoli-Moghaddam, B. Zahiri, and M. Mohammadi, "Sustainable design of a closed-loop location-routing-inventory supply chain network under mixed uncertainty," Transportation Research Part E: Logistics and Transportation Review, vol. 89, pp. 182–214, 2016.

[39] M. Rabbani, R. Heidari, and R. Yazdanparast, "A stochastic multi-period industrial hazardous waste location-routing problem: Integrating NSGA-II and Monte Carlo simulation," European Journal of Operational Research, vol. 272, pp. 945–961, 2019.

[40] M. Biuki, A. Kazemi, and A. Alinezhad, "An integrated location-routing-inventory model for sustainable design of a perishable products supply chain network," Journal of Cleaner Production, vol. 260, p. 120842, 2020.

[41] X. Shang, G. Zhang, B. Jia, and M. Almanaseer, "The healthcare supply location-inventory-routing problem: A robust approach," Transportation Research Part E: Logistics and Transportation Review, vol. 158, p. 102588, 2022.

[42] D. Bertsimas and M. Sim, "The price of robustness," Operations research, vol. 52, pp. 35–53, 2004.

[43] M. M. M. Chavez, Y. Costa, and W. Sarache, "A three-objective stochastic location-inventory-routing model for agricultural waste-based biofuel supply chain," Computers & Industrial Engineering, vol. 162, p. 107759, 2021.

[44] S. Daroudi, H. Kazemipoor, E. Najafi, and M. Fallah, "The minimum latency in location routing fuzzy inventory problem for perishable multi-product materials," Applied Soft Computing, vol. 110, p. 107543, 2021.

[45] W. Klibi, F. Lasalle, A. Martel, and S. Ichoua, "The stochastic multiperiod location transportation problem," Transportation Science, vol. 44, pp. 221–237, 2010.

[46] W. Klibi, A. Martel, and A. Guitouni, "The impact of operations anticipations on the quality of stochastic location-allocation models," Omega, vol. 62, pp. 19–33, 2016.

[47] M. Moshref-Javadi and S. Lee, "The latency location-routing problem," European Journal of Operational Research, vol. 255, pp. 604–619, 2016.

[48] S. Nucamendi-Guillén, I. Martínez-Salazar, S. Khodaparasti, and M. E. Bruni, "New formulations and solution approaches for the latency location routing problem," Computers & Operations Research, vol. 143, p. 105767, 2022.

[49] A. Osorio-Mora, C. Rey, P. Toth, and D. Vigo, "Effective metaheuristics for the latency location routing problem," International Transactions in Operational Research, vol. 30, pp. 3801–3832, 2023.

[50] A. Osorio-Mora, J. W. Escobar, and P. Toth, "An iterated local search algorithm for latency vehicle routing problems with multiple depots," Computers & Operations Research, vol. 158, p. 106293, 2023.

[51] P. Beraldi, M. E. Bruni, D. Laganà, and R. Musmanno, "The risk-averse traveling repairman problem with profits," Soft Computing, vol. 23, pp. 2979–2993, 2019.

[52] M. E. Bruni, P. Beraldi, and S. Khodaparasti, "A hybrid reactive grasp heuristic for the risk-averse k-traveling repairman problem with profits," Computers & Operations Research, vol. 115, p. 104854, 2020.

[53] M. Bruni, S. Khodaparasti, and P. Beraldi, "The selective minimum latency problem under travel time variability: An application to post-disaster assessment operations," Omega, vol. 92, p. 102154, 2020.

[54] R. T. Rockafellar, S. Uryasev, et al., "Optimization of conditional value-at-risk," Journal of risk, vol. 2, pp. 21–42, 2000.

[55] K. Helsgaun, "An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems," tech. rep., Roskilde University, 2017.

[56] J. W. Escobar, R. Linfati, and P. Toth, "A two-phase hybrid heuristic algorithm for the capacitated location-routing problem," Computers & Operations Research, vol. 40, pp. 70–79, 2013.

[57] M. Schneider and M. Löffler, "Large composite neighborhoods for the capacitated location-routing problem," Transportation Science, vol. 53, pp. 301–318, 2019.

[58] A. A. Juan, J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira, "A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems," Operations Research Perspectives, vol. 2, pp. 62–72, 2015.

[59] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, "The IRACE package: Iterated racing for automatic algorithm configuration," Operations Research Perspectives, vol. 3, pp. 43–58, 2016.

[60] C. Prins, C. Prodhon, and R. W. Calvo, "Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking," 4or, vol. 4, pp. 221–238, 2006.

[61] R. W. Floyd, "Algorithm 97: shortest path," Communications of the ACM, vol. 5, no. 6, p. 345, 1962.

[62] N. Noyan, "Risk-averse two-stage stochastic programming with an application to disaster management," Computers & Operations Research, vol. 39, pp. 541–559, 2012.