urban science

MDPI

Article

Scatter Search Algorithm for a Waste Collection Problem in an Argentine Case Study

Diego Rossit [1,2,*], Begoña González Landín [3,*], Mariano Frutos [1,4] and Máximo Méndez Babey [3]

[1] Department of Engineering, Universidad Nacional del Sur, Bahía Blanca 8000, Argentina; mfrutos@uns.edu.ar
[2] Instituto de Matemática, Universidad Nacional del Sur (UNS)—CONICET, Bahía Blanca 8000, Argentina
[3] Instituto Universitario SIANI, Universidad de Las Palmas de Gran Canaria, 35017 Las Palmas de Gran Canaria, Spain; maximo.mendez@ulpgc.es
[4] Instituto de Investigaciones Económicas y Sociales del Sur, Universidad Nacional del Sur (UNS)—CONICET, Bahía Blanca 8000, Argentina
* Correspondence: diego.rossit@uns.edu.ar (D.R.); bego.landin@ulpgc.es (B.G.L.)

Abstract: Increasing urbanization and rising consumption rates are putting pressure on urban systems to efficiently manage Municipal Solid Waste (MSW). Waste collection, in particular, is one of the most challenging aspects of MSW management. Therefore, developing computer-aided tools to support decision-makers is crucial. In this paper, a Scatter Search algorithm is proposed to address the waste collection problem. The literature is relatively scarce in applying this algorithm, which has proven to be efficient in other routing problems, to real waste management problems. Results from real-world instances of an Argentine city demonstrate that the algorithm is competitive, obtaining, in the case of small instances, the same outcomes as those of an exact solver enhanced by valid inequalities, although requiring more computational time (as expected), and significantly improving the results of the latter for the case of larger instances, now requiring much less computational time. Thus, Scatter Search proves to be a competitive algorithm for addressing waste collection problems.

Keywords: municipal solid waste; waste collection; scatter search; mixed-integer programming; valid inequalities; case study

Citation: Rossit, D.; González Landín, B.; Frutos, M.; Méndez Babey, M. Scatter Search Algorithm for a Waste Collection Problem in an Argentine Case Study. *Urban Sci.* **2024**, *8*, 240. https://doi.org/10.3390/urbansci8040240

Academic Editors: Guoray Cai and Thomas W. Sanchez

Received: 29 September 2024
Revised: 13 November 2024
Accepted: 22 November 2024
Published: 2 December 2024

1. Introduction

In modern societies, municipal solid waste (MSW) management has emerged as a key priority for ensuring environmental sustainability, public health and the overall well-being of communities [1]. Among the various stages of the MSW system, the waste collection stage is one of the most tricky since it poses significant challenges due to its logistical complexity [2]. Implementing an efficient waste collection system is essential not only for maintaining the livability and quality of life for residents, but also for environmental protection and economic efficiency. Effective route management can significantly reduce operational costs and improve the overall efficiency of the waste management process [3] and reduce greenhouse emissions [4].

Moreover, route optimization and planning contributes to greater predictability in waste collection services. An efficient system allows residents to know when to expect collection, which helps to reduce waste accumulation in public areas and avoids the need for emergency services or additional collections outside scheduled times. This predictability minimizes the costs associated with emergency management and unforeseen interventions. Furthermore, improved planning can enhance the efficiency of scheduling other municipal services by coordinating routes to avoid conflicts with public transportation or emergency services, ultimately reducing delays and operational costs related to congestion and urban planning.

In Argentina, where the case study is focused, the design of waste collection routes is primarily based on empirical knowledge from decision-makers [5]. While this practical

experience is invaluable, the literature offers extensive insights into cost reduction and environmental impact through computational models that support the decision-making process [6–8]. By providing a solid foundation for decision-making, these computational support tools enable municipal managers to make adjustments and adapt the system to the changing needs of the city. However, waste collection problems are well-known NP-hard challenges, making them computationally difficult to solve [9]. Consequently, the design of efficient computational methods, based mainly on heuristic and metaheuristic approaches that can provide good solutions in reasonable computational times, are commonly employed in the literature.

This study proposes computational models to address the waste collection problem in the Argentine city of Bahía Blanca, focusing specifically on the downtown area. This sector was chosen because the collection points, where citizens will deposit their waste, have already been established. These points are strategically located to maximize waste management efficiency and facilitate access for both citizens and collection trucks [10]. The waste collection problem is mathematically modeled as a vehicle routing problem with time limits and capacity constraints. This problem is solved in two ways: exactly, using mixed-integer programming, and heuristically, using a Scatter Search (SS) algorithm. The exact resolution method is based on mixed-integer programming using different commercial solvers and valid inequalities to enhance the formulation. The metaheuristic approach employs an SS algorithm, which has proven efficient in solving various combinatorial optimization problems. As far as we know, this article is the first work to apply SS to the waste collection problem and to perform with this approach extensive computational experimentation using real-world instances. Considering different combination and improvement methods and different local search sizes, 36 SS configurations were implemented. Both exact and metaheuristic methods were compared in computational experiments based on real-world data from Bahía Blanca. The results show that SS is effective in tackling the problem, producing optimal solutions with small instances, although with longer computational times—as would be expected when comparing an exact method with a metaheuristic on problems easily tackled by the exact method. However, with larger instances, the SS algorithm clearly outperforms the exact method, which—even when enhanced by valid inequalities—struggled to handle the complexity of the instances. For larger instances, the SS algorithm obtains better results in smaller computing times.

In this context, this article contributes to the related literature with the following: (i) the first Scatter Search algorithm to address real-world instances of the waste collection problem in the literature; (ii) a study of the capacity to enhance the exact resolution of the problem using valid inequalities; and (iii) a computational experiment conducted on real-world instances from an Argentine city. This article is an invited extension of our conference work presented at the "VI Ibero-American Congress of Smart Cities" that was held from 13–17 November 2014 in Mexico and Cuernavaca City [11]. Compared to our conference work, the enhancements in this article include addressing a different optimization problem within waste management and introducing a new metaheuristic solution method. Additionally, we have expanded the literature review and computational experiments.

This article is structured as follows: Section 2 presents a review of the main related works. Section 3 develops the exact method for the waste collection problem, including the mathematical formulation and the used of valid inequalities to enhance the resolution process. Section 4 presents the Scatter Search algorithm and its main features. Section 5 describes the computational experimentation, including the description of the realistic instances, the tests performed over the valid inequalities for enhancing the exact resolution, the proposed implementations of the SS algorithm, and the comparison between the implementations of SS and the exact method. Finally, Section 6 outlines the main conclusion and future research lines.

2. Literature Review

The literature extensively addresses waste collection problems using various computational methods [12]. For a recent review on this topic, interested readers may refer to Hess et al. [2]. Waste collection problems are well-known variants of traditional vehicle routing problems, which are generally considered NP-hard [13]—that is, problems for which no efficient algorithms exist that can solve them in polynomial time with respect to the problem size. This class characterizes problems that are computationally difficult to solve [9]. As a result, heuristic and metaheuristic approaches have been widely used to tackle the computational complexity of waste collection problems [14]. While Scatter Search has been successfully employed to address several combinatorial optimization problems, including routing problems [15], it remains relatively underexplored in the context of waste management, in general, and waste collection problems, in particular [16].

According to a search performed in Scopus with the criteria schema TITLE-ABS-KEY (list of terms) using as keyword terms: ('waste' OR 'waste management') AND ('scatter search'), only five works related to the application of SS in waste management were found. Moreover, if the focus is put on waste collection, only two works are mentioned. One is the work of Zhang et al. [17]. Although the conceptual model in this work refers to the collection of recyclable waste, the SS algorithm was tested on synthetic instances built from the procedure developed in Dethloff [18]. The proposed SS outperformed a genetic algorithm in terms of computational efficiency, achieving similar results in significantly less time. Another related work is that of Chu et al. [19], who, while not applying SS directly to waste management, proposed waste collection as a potential application of their algorithm. Mainly they present an SS to address the periodic capacitated arc routing problem (PCARP). They compared SS with an insertion heuristic using PCARP instances derived from the benchmarks of Golden et al. [20] and Belenguer and Benavent [21], with SS outperforming the insertion heuristic in most cases.

Although no other applications of SS in waste collection routing problems were found, a few works have applied SS to other waste management-related problems. Yu et al. [22] used SS to optimize the transport of industrial waste from plants to intermediate recycling facilities and, eventually, to landfills using a transport-like mathematical model. They compared two SS versions: the typical SS and a modified SS where the update method is altered to maintain population diversity, rather than replacing the worst solution as in the traditional method. The modified version proved more efficient in tests conducted on synthetic, randomly generated instances.

Similarly, Ortega et al. [23] employed an SS hybridized with linear programming to design a multi-period reverse logistics network for MSW, which is mainly a location and transport optimization problem. In this network, the waste generated in towns has to be allocated to transfer centers and then to treatment plants. While the treatment plant location is fixed for the whole planning horizon, the location of the transfer centers may vary. The authors introduced randomness into the problem by varying waste generation based on empirical probabilistic distributions. They compared their algorithm with CPLEX. For small problems, SS reached the optimum in slightly more time than CPLEX, but for larger problems, SS was faster in reaching the optimum when CPLEX succeeded. The computational experiments were conducted on synthetic instances created by the authors, as well as on an instance partly based on the region of Gipuzkoa, Spain, and partly synthetic (mainly regarding waste generation).

Thus, the application of SS to real-world waste management problems is very limited. Most studies utilize synthetic instances that resemble waste management problems. Only the work of Ortega et al. [23] applied SS to a realistic instance for the network design of an actual region, although many key parameters were still synthetically generated. Concerning the specific problem of waste collection routing, the literature is even scarcer. Only two works address routing problems potentially representing waste collection, both of which rely on synthetic instances from benchmarks that are not related to waste management. Therefore, we consider that there is still a gap in the literature regarding the

implementation of Scatter Search—a metaheuristic that has proven efficient in solving other combinatorial optimization problems—specifically, for the waste collection problem, and its testing in realistic scenarios. As far as we know, this is the first work to propose an SS algorithm to the waste collection problem and perform an extensive computational experiment in real-world instances.

3. Exact Resolution

This Section presents the exact resolution approach used for the waste collection problem of Bahía Blanca. In particular, the basic mathematical formulation as a typical routing problem and a discussion about some valid cuts to improve the formulation are presented.

3.1. Mathematical Formulation

The typical waste collection problem consists in a set of collection points $I = \{1, 2, \ldots, n_I\}$ distributed in an area and a set of collection vehicles or trucks $V = \{1, 2, \ldots, n_V\}$. In addition, if 0 represents the depot where vehicles start and end their routes, and where collected waste is deposited, the set $I^0 = I \cup \{0\}$ is defined. The trucks have to depart from the depot, visit every collection point to obtain the waste and return to the depot to unload while respecting certain operative restrictions (truck capacity and travel time). The objective of the problem is to minimize the traveled distance, which is typically associated with reduced expenses, particularly in terms of fuel consumption and greenhouse gas emissions for diesel trucks like those used in the city of the case study. In order to present the model that is used in the exact solver, the following parameters are defined:

- Q: Truck capacity.
- c_{ig}: Travel time between points $i \in I^0$ and $g \in I^0$.
- se_i: Service time of collection point $i \in I^0$. At the depot, the service time corresponds to the time spent unloading the collected waste.
- b_i: Amount of waste generated daily at collection point $i \in I$.
- α: Cost per minute for collection trucks.
- T_L: Maximum time a vehicle can spend performing a route including unloading waste at the depot.

And the model uses the following variables:

- x_{igl}: (binary) 1 if truck $l \in L$ travels between collection points $i \in I^0$ and $g \in I^0$, and 0 otherwise.
- v_{igl}: (continuous non-negative) load of truck $l \in L$ when going from collection point $i \in I$ to collection point $g \in I$.

Taking all these elements into account, the following mathematical model based on Mixed-Integer Linear Programming (MILP) is proposed:

Minimize

$$f(x) = \alpha \sum_{l \in L} \sum_{i \in I^0} \sum_{g \in I^0} \Big(x_{iglt} \big(c_{ig} + se_i \big) \Big) \tag{1a}$$

subject to

$$\sum_{\substack{i \in I^0 \\ i \neq g}} \sum_{l \in L} x_{igl} = 1, \quad \forall\, g \in I, \tag{1b}$$

$$\sum_{\substack{i \in I^0 \\ i \neq g}} x_{igl} - \sum_{\substack{i \in I^0 \\ i \neq g}} x_{gil} = 0, \quad \forall\, g \in I^0, l \in L, \tag{1c}$$

$$\sum_{i \in I} x_{0il} \leq 1, \quad \forall\, l \in L, \tag{1d}$$

$$\sum_{\substack{i,g\in I^0\\ i\neq g}} x_{igl}\left(c_{ig}+se_i\right) \leq T_L, \quad \forall\, l \in L, \tag{1e}$$

$$v_{igl} \leq Q x_{igl}, \quad \forall\, i \in I^0, g \in I^0, l \in L, \tag{1f}$$

$$\sum_{\substack{i\in I^0\\ i\neq g}} v_{igl} + h_g \leq \sum_{\substack{i\in I^0\\ i\neq g}} v_{gil} + Q\left(1 - \sum_{\substack{i\in I^0\\ i\neq g}} x_{igl}\right), \quad \forall\, g \in I, l \in L, \tag{1g}$$

$$x_{igl} \in \{0,1\}, v_{igl} \geq 0, \quad \forall\, i \in I^0, g \in I^0, l \in L.$$

Equation (1a) represents the objective of minimizing the routing cost, which is based on the sum of the travel times and the service times (time to empty the collection points) for each route performed by the fleet of trucks. Equation (1b) presents the requirement that every collection point is visited by a collection truck. Equation (1c) requires that if a collection point is visited by a truck, this collection point is also left by the same truck. Equation (1d) requires that each collection truck only performs at most one route. Equation (1e) checks that the route does not exceed the maximum time that the vehicle can spend performing a route, which includes traveling between collection points, collecting waste of the collection pint (service time), and unloading waste at the depot in the end of the route. Equation (1f) ensures that the capacity of the trucks is not exceeded. Finally, Equation (1g) is the subtour elimination constraint, which also keeps track of the accumulated waste throughout the route.

3.2. Valid Cuts

The routing problems are computationally challenging problems [13], primarily due to the presence of many symmetric solutions. Two solutions are considered symmetric if they result in the same objective function value but differ in variable assignments [24]. For instance, one route can be performed by any truck of the homogeneous fleet and the solution incurs the same cost. Thus, the resolution solver has no mechanism to exclude one of them and this may have an impact in the efficiency of the resolution process.

To address this issue, Valid Inequalities (VIs) can be introduced into the model. A VI is a constraint that tightens the feasible region of the problem without eliminating any optimal solutions. For a thorough review of VIs in routing problems, the interested reader is referred to the work of Dror et al. [25]. In this paper, the performance for improving the mathematical formulation of Section 3.1 is tested with four typical VIs that were used in Mahéo et al. [24].

3.2.1. Furthest Visit

Without loss of generality, since each collection point can only be visited by one of the trucks in the fleet, the first truck is forced to be the one that visits the collection point furthest from the depot. These symmetric solutions are avoided using VI (2).

$$\sum_{i\in I} x_{igl} = 0, \quad \forall\, l \in L, l > 0, g \in I, g = \arg\max_{g\in I}\{c_{0g}\}. \tag{2}$$

3.2.2. Vehicles Are Ordered

It is mandated that a vehicle or truck with index l can only depart from the depot if the truck with index $l-1$ has already done so. In scenarios where not all available trucks are utilized, an unused truck could be swapped for one that is in use. These symmetric solutions are prevented by means of VI (3).

$$\sum_{\substack{g\in I\\ g\neq 0}} x_{0gl} \leq \sum_{\substack{g\in I\\ g\neq 0}} x_{0gl-1}, \quad \forall\, l \in L, l > 0. \tag{3}$$

3.2.3. Vehicle Has to Start Empty

A vehicle or truck must begin its route with an empty load. This avoids solutions with varying delivery plans. That is, if a truck completes its collection tour without reaching full capacity, an alternative solution could be considered where the truck starts with any load less than the remaining unused capacity. These symmetric solutions are avoided using VI (4).

$$\sum_{\substack{g \in I \\ g \neq 0}} v_{0gl} = 0, \quad \forall\, l \in L. \tag{4}$$

3.3. Solvers and Implementation Details

Exact solvers play a pivotal role in addressing complex mathematical combinatorial optimization problems, such as routing problems. In recent years, commercial solvers have significantly enhanced their capabilities to handle computationally intensive tasks, driven by a competitive landscape aimed at providing decision-makers with the most effective options.

There are several solvers available for mixed-integer programming problems. For a comprehensive list of solvers, refer to the NEOS Server website [26]. Among open-source solvers are CLP, CBC, LP_Solve, and GLPK [27]. In general, noncommercial MILP software tools may lack the speed and robustness of their commercial counterparts but provide a practical alternative for users who cannot afford commercial options [28]. For commercial solvers, a wide range is available, including CPLEX, Gurobi, XPRESS, and LINDO [29], with Gurobi and CPLEX among the most commonly used [30,31].

In line with this, the present study compares the performance of two renowned, state-of-the-art solvers: Gurobi version 11.0.3 [32] and CPLEX version 22.1.1 [33], both versions released in 2024. By evaluating these solvers, we aim to provide valuable insights for researchers and practitioners seeking to optimize complex combinatorial optimization problems.

The mathematical model and VIs presented in the previous section were implemented in Python using Pyomo [34] as a modeling environment. Pyomo provides a flexible platform for defining optimization problems and enables connections to various commercial solvers, including Gurobi and CPLEX, which were utilized in this study.

4. Scatter Search Algorithm

The Scatter Search algorithm—proposed by Glover in 1977 [35]—is a metaheuristic optimization technique aimed at solving complex combinatorial and continuous optimization problems. A core feature of Scatter Search is its focus on maintaining a reference set of diverse solutions, which are combined to generate new candidate solutions. This process enhances the likelihood of finding high-quality solutions [36]. In addition, the flexibility of SS allows adaptation to various problem domains, making it applicable in areas such as scheduling, routing, and resource allocation. By effectively balancing exploration and exploitation, SS aims to provide high-quality solutions efficiently. Its successful application across fields like logistics, telecommunications, finance, and engineering demonstrates its capability in addressing real-world optimization challenges [15]. As shown in Figure 1, it consists of five methods [37]:

- A diversification generation method used to generate a set of diverse solutions that are the basis for initializing the search. The outcome consists of a set of solutions that is referred to as the population.
- An improvement method for modifying solutions to improve their quality (in terms of the objective function) or feasibility.
- A reference set update method used in the main iterative loop of any scatter search implementation. The output is a set of solutions known as the reference set. Typically, the number of reference solutions is of the order of 10% of the population size.

- A subset generation method that produces subsets of reference solutions which become the input to the combination method.
- A combination method that uses the output from the subset generation method to create new solutions.

The main iterative loop is executed as long as at least one New Reference Solution (NRS) is obtained. The process continues until the Stopping Criterion is Satisfied (SCS).

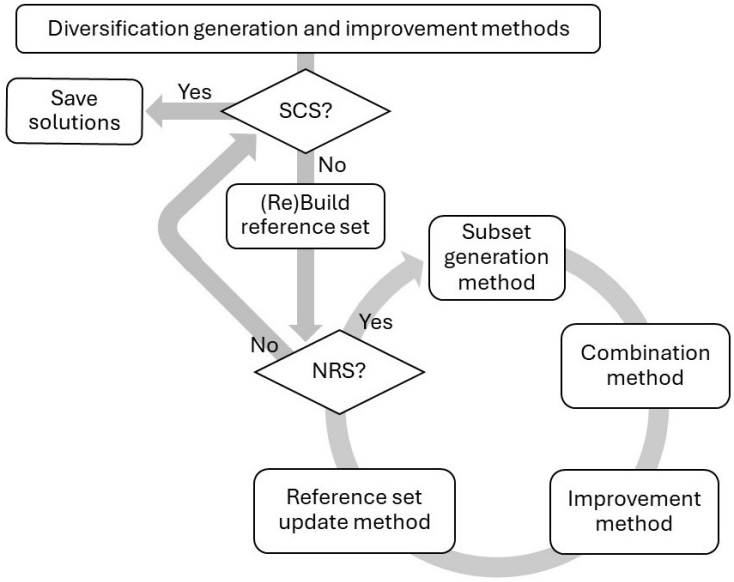


Figure 1. Scatter Search algorithm flowchart.

In this paper, the SS algorithm will be applied using various combinations of some key elements of the algorithm in order to evaluate the effectiveness of each configuration in addressing the waste collection problem. Specifically, different combination and improvement methods and different sizes of the local search, in the sense of the number of times the improvement method is applied to a solution, will be tested. This approach aims to identify the most effective configurations to optimize waste collection routes and improve overall system performance. The SS was also coded using Python as the programming language.

5. Computational Experimentation

This Section presents the description of instances considered, the tests performed on the VIs with the exact method, the proposed implementations of the SS algorithm and the comparison between the SS and the exact method.

5.1. Description of Instances and Resolution Platform

To carry out the computational experimentation in this work, 10 realistic instances were constructed with varying numbers of collection points: three instances with 15, three with 30, three with 50, and one with 100 collection points. All the instances are based on information gathered on field studies in Bahía Blanca [10]. These instances can be retrieved from Github (https://github.com/diegorossit/Urban_Science_waste_collection_BBCA.git, accessed on 27 October 2024). The naming convention follows the format: $n - id$, where n is the number of collection points, and id is a reference number to distinguish between instances with the same number of collection points.

Each instance consists of two files: "times.txt", which contains the matrix of travel times (minutes) from the depot to the collection points, between collection points, and from the collection points to the depot; and "waste.txt", which contains the geographic coordinates of each collection point, as well as the waste (m^3) generated daily by the urban area assigned to each collection point. Moreover, the service time of a collection point was set at 0.78 min based on the field work of Carlos et al. [38] for a homogeneous

collection vehicles fleet; the truck unloading time was set at 8 min [39], and the cost per minute of truck operation (α) was estimated at USD 0.5764 per minute [40]. Also, as in Mahéo et al. [24], given that the instances are smaller than the actual collection zones in the city, the capacity and size of the fleet were adjusted to ensure that the problem does not become trivial, where a single truck can collect all the waste in a single trip (see Table 1). In all cases, the maximum time to complete a trip or route was set at 6 h (the working hours of an employee in the sector in Bahía Blanca).

Table 1. Problem parameters.

Instance Size (Number of Collection Points)	Truck Fleet Size	Truck Capacity (m^3)
15	8	10
30	16	20
50	20	21
100	20	21

The resolution platform that was used for the SS and the exact method are a personal computer with a Processor Intel(R) Xeon(R) Silver 4216 CPU @ 2.10 GHz, 2095 Mhz (2 threads) and 64 GB of RAM within a Windows 11 operative system environment and Python compiler version 3.10.11.

5.2. Tests over VIs and Solver of Exact Method

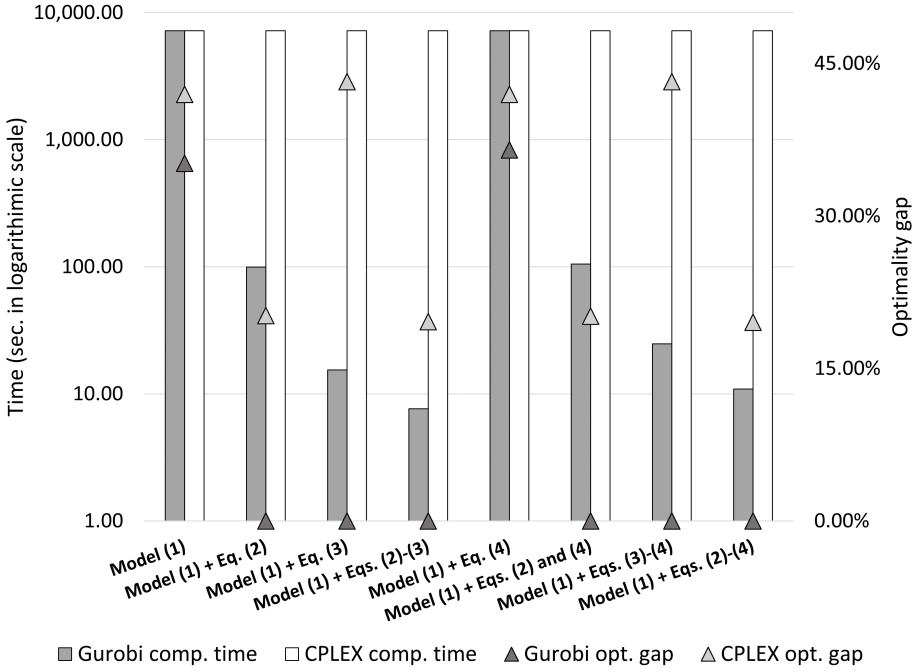
This section presents the results of tests analyzing the effect of VIs in enhancing the MILP model introduced in Section 3.1. As aforementioned, the tests were conducted using two state-of-the-art solvers, CPLEX and Gurobi, on the three smallest instances involving 15 collection points. Considering that there were three different VIs and all combinations between VIs and the MILP model were tested, eight runs were performed for each solver and each instance. A maximum allowable solving time of 7200 s was used for all cases.

Figure 2 illustrates the key results for the three instances. The bars show computing times (left x-axis), while triangle markers indicate optimality gaps (right x-axis), which are non-zero when the solver fails to converge within the allotted time (7200 s). For the sake of clarity, computing times are displayed on a logarithmic scale. The results of the detailed computing times and optimality gaps are presented in Appendix A. As aforementioned, each instance was tested with various mathematical formulations that resulted from the combination of Model (1) and the VIs from Equations (2)–(4). For example, the tag "Model (1) + Equations (2) and (4)" indicates that VIs (2) and (4) were added to Model (1). In particular, the tag "Model (1)" represents the base model presented in Section 3.1 without any VIs.

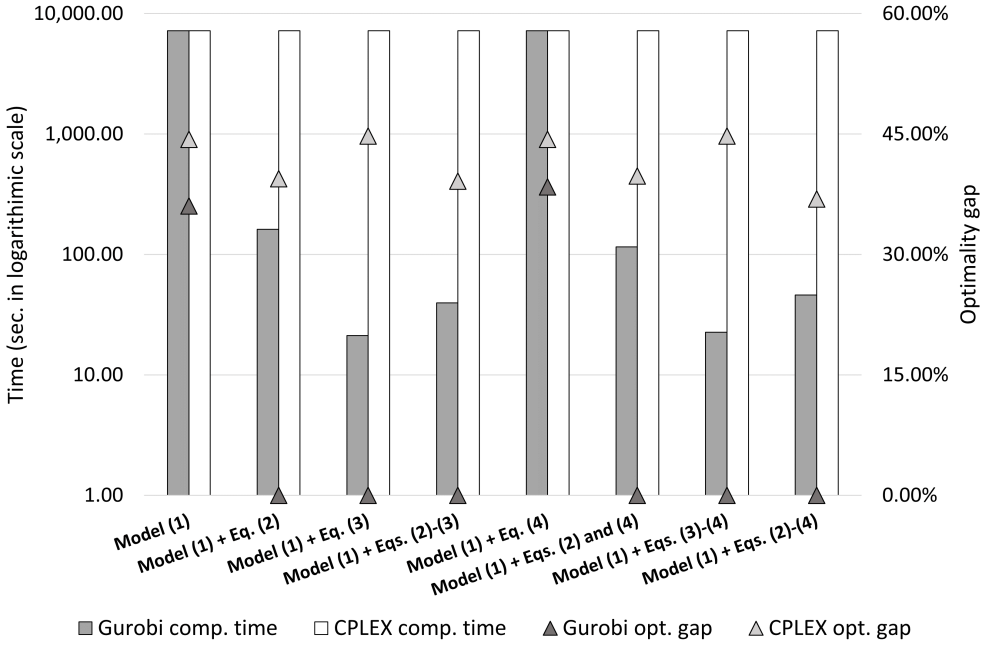
The comparison between the solvers reveals that Gurobi consistently outperforms CPLEX in these instances. In some cases, the difference is significant, such as in instance 15-1 with the formulation "Model (1) + Equations (3)–(4)", where Gurobi finds the optimal solution in under 8 s, while CPLEX only achieves a suboptimal solution with a 20% gap after reaching the time limit. Moreover, CPLEX fails to converge to the optimal solution for any instance or formulation. Additionally, in the formulations in which both solvers cannot obtain the optimal solution (e.g., "Model (1)" and "Model (1) + Equation (4)" across all three instances), Gurobi performs better, showing a smaller optimality gap.

Although the internal mechanisms of commercial solvers are not publicly accessible, these results align with recent findings where Gurobi outperformed CPLEX in complex integer problems. The detailed results are depicted in Appendix A, where it is shown that while CPLEX often finds the optimal solution, it typically fails to prove optimality within the time limit due to a weak lower bound. Nonetheless, as the literature emphasizes, Gurobi's performance advantage is problem-specific; different types of problems can lead to cases where either solver may outperform the other [30,41–43]. Thus, the recommendation for

Gurobi in this study is derived specifically from the computational experiments conducted here, without implying its superiority across all problem contexts.

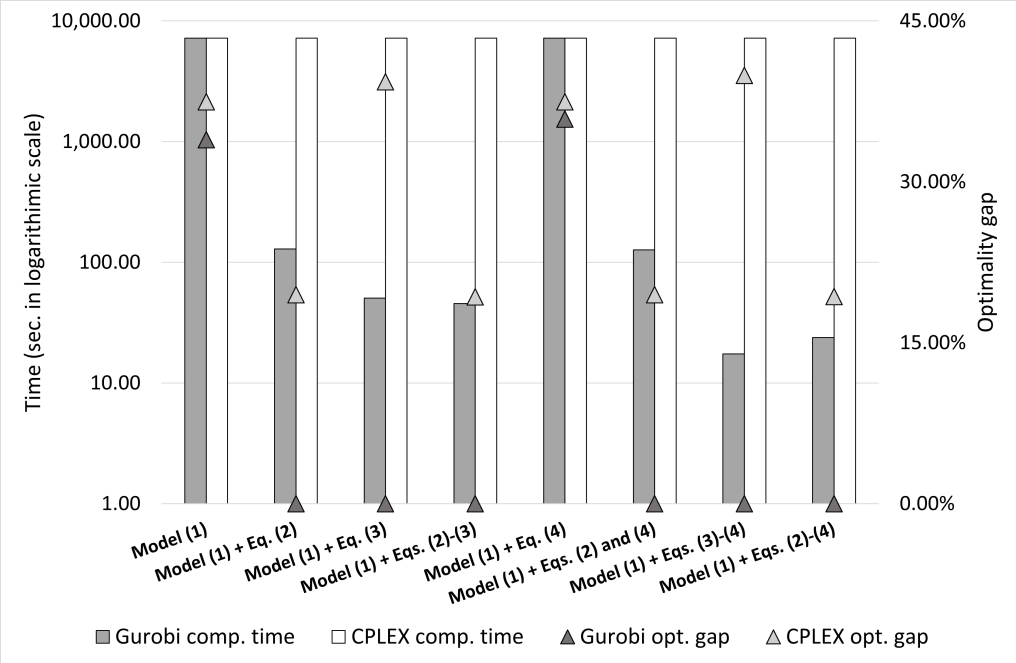


(**a**) Instance 15-1.



(**b**) Instance 15-2.

Figure 2. *Cont.*



(**c**) Instance 15-3.

Figure 2. Comparison of solvers and VIs for the exact method.

Regarding the effectiveness of VIs, in instance 15-1, the best result with Gurobi is achieved using formulation "Model (1) + Equation (2) and (3)" with a computing time of 7.63 s. For instances 15-2 and 15-3, the best results are obtained with the formulation "Model (1) + Equations (3) and (4)" yielding times of 22.63 s and 17.45 s, respectively. For the comparison with the Scatter Search algorithm, the Gurobi solver was used, applying the formulation "Model (1) + Equations (2) and (3)" as it achieved the smallest average computing time across the three tested instances.

5.3. Implementations of the SS Algorithm

This section presents the SS algorithms implemented to solve the problem described in Section 3.1. A permutation coding was considered for the representation of the solutions. Thus, the variable x is replaced by the chromosome $y = [y_i]$, with a permutation of the order of collection of waste from the n_I collection points and the variable v is determined when the fitness function is calculated. Thus, by handling permutations, constraint (1c) is always satisfied. The remaining constraints are imposed when evaluating the fitness function. On the one hand, restrictions (1b), (1d), (1f), and (1g) are met by adding trucks to the overall route (defined by the chromosome) once the previous truck has filled up or does not have the capacity to load all the waste deposited at the next collection point. On the other hand, solutions are penalized in the event they do not meet restriction (1e) or the size of the fleet (n_L) is exceeded, as shown in Equation (5).

$$f_{fitness}(x) = f(x) - \lambda\, min\{0, n_L - n(x)\} - \gamma \sum_{l \in L} min\{0, T_L - TT_l\} \quad (5)$$

where $f(x)$ is the objective function (1a), λ and γ are positive real numbers, $n(x)$ is the number of trucks required to collect all the accumulated waste, one for each route of the solution x, and TT_l is the summation of the left-hand side of the constraint (1e).

The following are the different methods that were considered:

- Diversification generation method: more_intertools.random_permutation() (https://pypi.org/project/more-itertools/, accessed on 27 October 2024).
- Improvement methods:

1 Exchange (EXC). Two alleles are chosen randomly and exchanged with each other.
2 Insertion (INS). Two alleles are chosen randomly and the second allele is placed just after the first one.
3 Inversion (INV). Two alleles are chosen at random and the order of the alleles between them is reversed.

- Reference set update method: the first half of the solutions in the reference set are chosen to be the best solutions (in terms of the fitness function value) in the population. The other half are chosen to be the most diverse (in terms of the Hamming distance) with respect to the solutions already incorporated in the reference set.
- Subset generation method: all possible pairs of solutions of the reference set are generated.
- Combination method:

1 Partially Mapped Crossover (PMX) [44].
2 Order Crossover (OX) [45]).
3 Circle Crossover (CX) [46].
4 Modified Circle Crossover (CX2) [47].

In addition, three sizes for local search were considered for the improvement methods: (1) 10, (2) 20, and (3) 30. Thus, in total, 36 configurations of the SS algorithm were made. Since SS is based on stochastic procedures, for each instance and SS configuration, 31 runs were made to obtain a proper sample of the distribution of the results for statistical analysis. A maximum number of evaluations of the fitness function was set as a stopping criterion for the algorithms. Table 2 shows the size of the reference set, the size of the population and the maximum number of fitness function evaluations considered according to the instance size.

Table 2. Algorithm parameters.

Instance Size (Number of Collection Points)	Reference Set Size	Population Size	Maximum Number of Fitness Function Evaluations
15	10	90	100,000
30	10	90	250,000
50	12	132	500,000
100	14	182	1,000,000

For instances of 15 and 30 collection points, a Kruskal–Wallis Rank Sum Test [48] was performed to determine whether there are significant differences between the four combination methods considered, the three improvement methods considered, and the three sizes for local search considered. When the value of a Kruskal–Wallis test is significant (p-value < 0.05), a multiple comparison test after Kruskal–Wallis [49] between treatments was performed to determine which levels are different, with pairwise comparisons adjusted appropriately for multiple comparisons. The tests were performed using the available libraries in the programming language R [50].

Figures 3 and 4 show the box-and-whisker plots associated with the results of the instances of 15 and 30 collection points, respectively, grouped by combination methods, improvement methods, and local search sizes. In both cases, it can be seen that the PMX and OX combination methods show the best performance, as well as the EXC improvement method. The size of the local search does not seem to be relevant. These conclusions are supported by the hypothesis tests performed (see Tables 3 and 4). A detailed statistical summary of the computational executions can be found in Appendix B.

Table 3. Kruskal–Wallis Rank Sum Test: *p*-values.

Instance	Combination Method	Improvement Method	Local Search Sizes
15-1	2.2×10^{16}	1.281×10^{08}	0.9543
15-2	2.2×10^{16}	6.430×10^{-11}	0.0799
15-3	2.2×10^{16}	5.504×10^{11}	0.8066
30-1	2.2×10^{16}	2.2×10^{16}	0.7939
30-2	2.2×10^{16}	2.2×10^{16}	0.2394
30-3	2.2×10^{16}	2.2×10^{16}	0.3931

Table 4. Outputs of the multiple comparison test after Kruskal–Wallis: if TRUE, then statistically significant differences are found between the compared levels.

Combination Method		Improvement Method
1–2 FALSE	2–3 TRUE	1–2 TRUE
1–3 TRUE	2–4 TRUE	1–3 TRUE
1–4 TRUE	3–4 TRUE	2–3 TRUE

Note: The outputs are the same for all instances of 15 and 30 collection points.

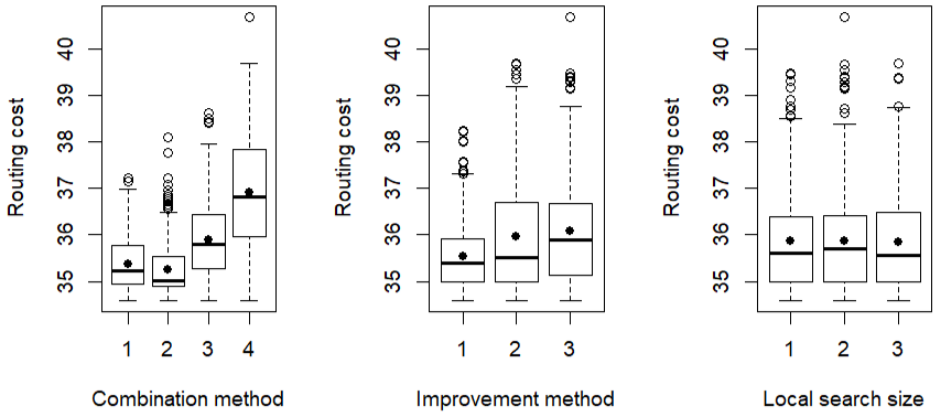
Taking into account the conclusions obtained with the instances of 15 and 30 collection points, the larger instances with 50 and 100 collection points were run with only the configurations that were more efficient, i.e., the configurations that use PMX or OX as the combination method and EXC as the improvement method, with a local search size of 20. Table 5 shows the statistical summary of the configurations executed on instances 50-1, 50-2, 50-3, and 100-1. From left to right, the table reports: the minimum (the best value of the objective function found), the first quartile, the median, the second quartile, the (pseudo) median, the lower bound and the upper bound of a 95% confidence interval for the (pseudo) median, the total runtime (in seconds), of the 31 runs performed in each case, and the mean runtime of each run (in seconds).

Table 5. Statistical summary (runtimes in seconds) of the configurations executed with the 50-1, 50-2, 50-3, and 100-1 instances.

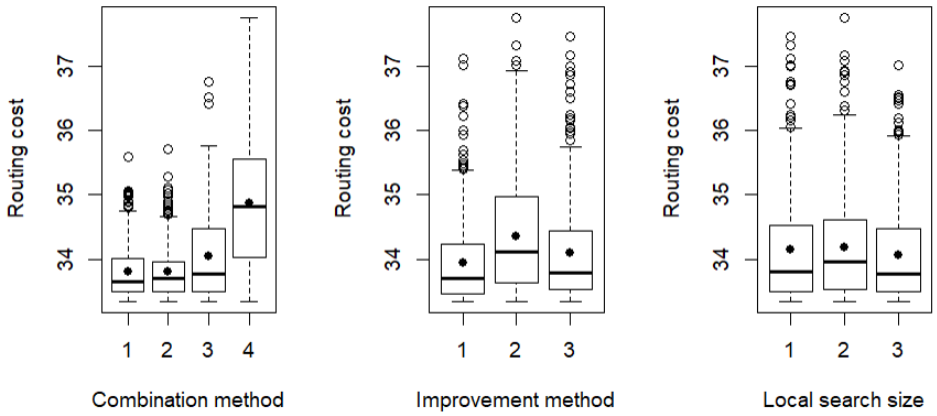
Instance	Comb. Method	min	Q1	Median	Q2	(Pseudo) Median	lb	ub	Total Runtimes	Mean Runtimes
50-1	PMX	79.36150	83.38203	84.75679	86.69644	84.96863	84.06797	85.97016	2222.0133	71.6778
	OX	82.68168	83.97286	85.37932	86.59845	85.30007	84.68186	85.97304	2285.4383	73.7238
50-2	PMX	83.75382	86.82326	87.46020	88.33636	87.44579	86.90684	88.11156	2217.8015	71.5420
	OX	83.06212	86.43129	87.73112	88.63898	87.62016	86.86361	88.21819	2296.5859	74.0834
50-3	PMX	82.61251	84.89225	85.82317	87.49190	86.16902	85.42832	86.90107	2219.7335	71.6043
	OX	81.53460	83.37627	85.11994	87.16335	85.19631	84.36194	86.14020	2290.5435	73.8885
100-1	PMX	151.6964	158.0572	159.8413	162.7925	160.0127	158.6192	161.2967	4863.0509	156.8726
	OX	156.4116	158.7144	159.7951	161.5936	159.9803	159.1870	160.8702	5184.1772	167.2315

Note: The 100-1 instance was run in a Processor 13th Gen Intel(R) Core(TM) i9-13900K with 128 GB of RAM.

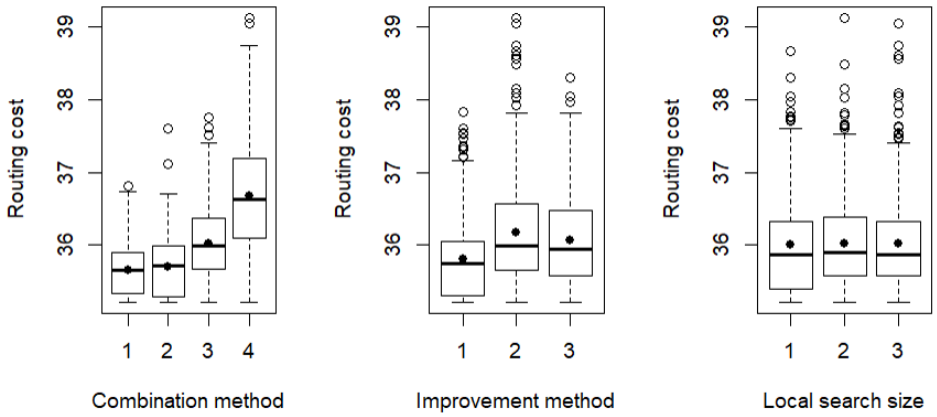
The results of Table 5 show that the mean and total runtime increase approximately linearly with the size of the problem, which is a good result of an NP-hard problem.



(**a**) Instance 15-1.

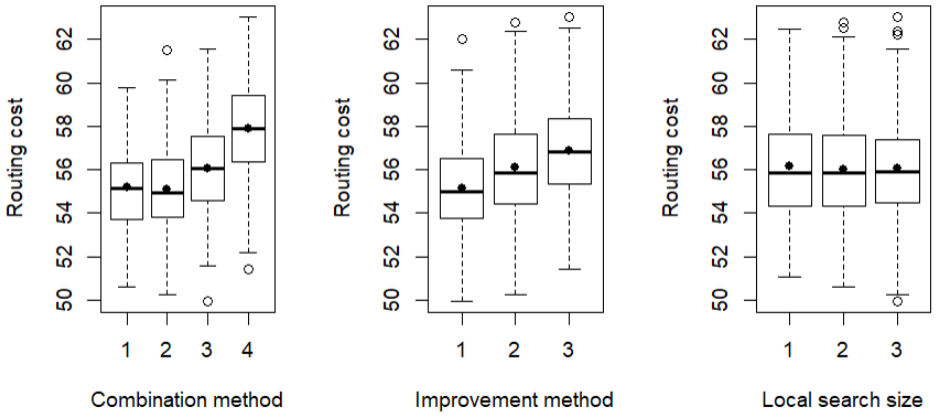


(**b**) Instance 15-2.

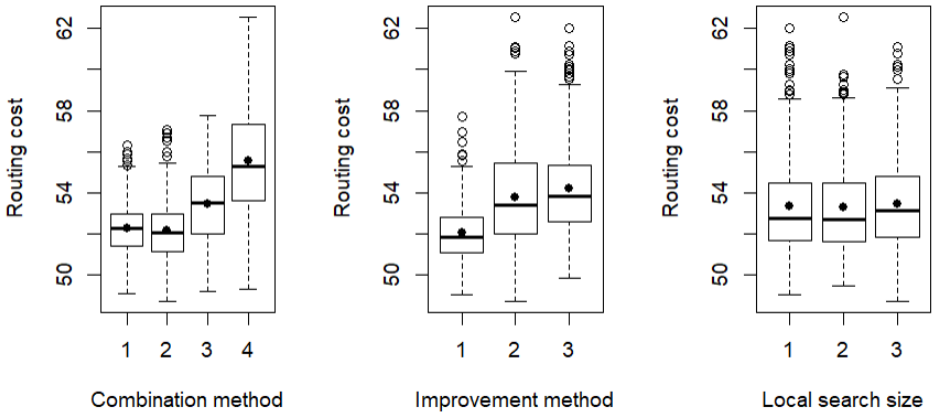


(**c**) Instance 15-3.

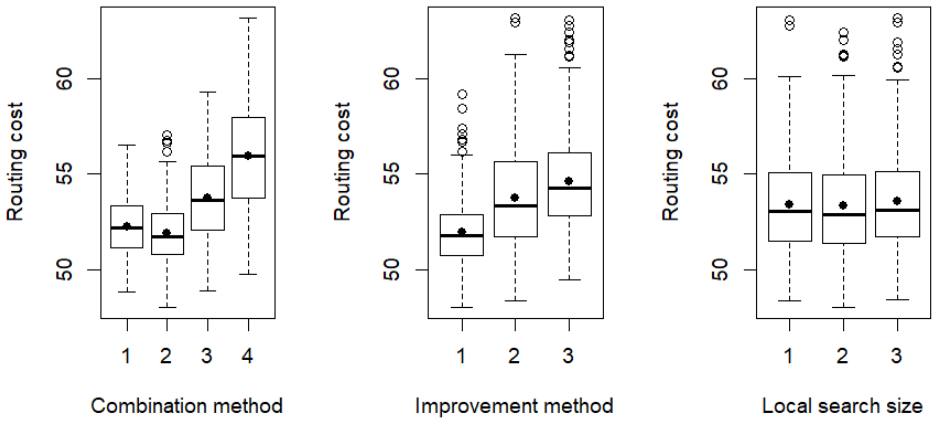
Figure 3. Box-and-whisker plots associated with the results of the three instances of 15 collection points grouped by combination methods, improvement methods, and local search sizes.



(**a**) Instance 30-1.



(**b**) Instance 30-2.



(**c**) Instance 30-3.

Figure 4. Box-and-whisker plots associated with the results of the three instances of 30 collection points grouped by combination methods, improvement methods, and local search sizes.

5.4. Comparison Between the Scatter Search and the Exact Method

In this section, we compare the results obtained with the Gurobi solver, applying the formulation "Model (1) + Equations (3) and (4)", with the SS algorithm configuration that uses OX as the combination method, EXC as the execution method, and 20 as the

size of the local search, since it is statistically one of the most efficient among the tested SS configurations. Table 6 shows the routing cost, lower bound, computing time in seconds, and optimality gap of the exact method; and the percentage difference, calculated with Equation (6), between the routing cost obtained with the exact method and the minimum and (pseudo) median values obtained with the (OX, EXC, 20) configuration of the SS algorithm.

$$\%\ \text{dif.} = \left(\frac{C_*^{SS} - C_*^{exact}}{C_*^{exact}} \right) \% \tag{6}$$

In Equation (6), C_*^{SS} is the routing cost obtained with the SS and C_*^{exact} is the routing cost obtained with the exact method. Thus, a negative value of % dif. means that the SS algorithm obtained a smaller (better) solution. In Table 6, the Equation (6) was calculated using two different values for C_*^{SS}: the minimal (best) value and the median value obtained for the 31 runs for each instance.

From the results presented in Table 6, it can be concluded that for the instances with 15 collection points, the SS algorithm was able to consistently obtain optimal solutions. Moreover, the median result across the 31 runs of the SS algorithm showed less than a 2% deviation from the optimal value for all three instances with 15 collection points. This validates the SS algorithm for small instances, as it is capable of producing near-optimal solutions in most runs. The total runtimes of the SS are much larger (around 250 s; see Tables A2–A4) than the exact method (around 20 s) as is expected for small instances in which the stochastic search of metaheuristics is overrun by the systematic search of the exact method. However, the mean runtime of the SS is competitive against the exact method (around 7 s, see Tables A2–A4).

Table 6. Comparison of the results obtained with the exact method and the SS algorithm.

Instance	Exact Method				Exact Method vs. SS	
	Routing Cost (USD)	Comp. Time (s)	Lower Bound	Opt. Gap (%)	min (%dif)	(Pseudo) Median (%dif)
15-1	34.59	24.80	34.59	0.00%	0.00%	1.99%
15-2	33.35	22.63	33.35	0.00%	0.00%	1.80%
15-3	35.22	17.45	35.22	0.00%	0.00%	1.00%
30-1	67.57	7201.82	34.78	94.27%	−23.41%	−19.37%
30-2	80.61	7200.75	35.16	129.27%	−38.49%	−36.05%
30-3	48.16	7201.73	35.11	37.17%	−00.25%	5.56%
50-1	123.57	7201.58	49.87	147.78%	−33.09%	−30.97%
50-2	-	7200.03	51.81	-	-	-
50-3	-	7200.02	49.96	-	-	-

Note: The minimum and (pseudo) median values are taken from the configuration (OX, EXC, 20).

For instances with 30 collection points, the SS algorithm clearly outperforms the exact method in terms of finding the best minimal solution. The routing plans generated by the SS algorithm were more than 20% better than those produced by the exact method for instances 30-1 and 30-2. However, in instance 30-3, the percentage difference between the SS and the exact method was much smaller, which aligns with the exact method's ability to achieve a relatively small optimality gap for this particular instance size. The median results follow a similar pattern, with the SS algorithm significantly outperforming the exact method in instances 30-1 and 30-2, while in instance 30-3, the SS median result was about 5% worse than the exact method's solution. In these instances, the total computing times of the SS are much smaller (around 700 s, see Tables A5–A7) than for the exact method, which used the maximum allowable time of 7200 s for all three instances. The exponential increase in computational time for the exact method as the instance size doubled from 15 to 30 collection points is typical of NP-hard problems.

For the larger instances with 50 collection points, only the instance 50-1 could be compared, as the exact method failed to find a solution within the allowable time (7200 s)

for the other two instances. In instance 50-1, the SS algorithm improved the exact solution by 33.09% when considering the best solution and by 30.97% when considering the median solution.

These results highlight the robustness of the SS algorithm in larger instances, where it clearly outperforms the exact method in both solution quality and computational efficiency.

6. Conclusions

Municipal Solid Waste (MSW) systems are critical components of modern societies, playing a crucial role in enhancing the sustainability and livability of cities. To achieve these objectives, various stages of the system must be carefully planned. This article addresses the waste collection problem, a complex logistics challenge that is often computationally expensive to solve to optimality. Waste collection is one of the most costly stages of the MSW system. Therefore, implementing computational intelligence methods to optimize this stage, which is typically difficult to solve manually, can reduce costs and significantly impact the system's overall expenses. In this line, this work proposes a Scatter Search algorithm to tackle the problem efficiently, being the first work in the literature to apply this approach to a real-world case study of the waste collection problem. The case study focuses on the medium-sized Argentine city of Bahía Blanca. Additionally, an exact resolution of the problem by means of mathematical programming is proposed.

Computational experimentation was performed over scenarios of different sizes to study the impact of the computational complexity. The exact method was tested using different valid inequalities to enhance the formulation and two different state-of-the-art commercial solvers. The results showed that Gurobi was significantly more efficient than CPLEX for this problem, and that the valid inequalities greatly strengthened the formulation, allowing the model to obtain the optimal solution in small instances, something that was not possible for the plain mathematical model without the valid inequalities. The Scatter Search algorithm was tested using different configurations for the improvement method, the combination method, and the size in the local search operator, giving a total of 36 different configurations of the Scatter Search algorithm. The results showed that the configurations using partially mapped crossover or order crossover as the combination method and exchange as the improvement method are the most efficient to address this problem. Moreover, the results showed that the size of the local search is not so relevant for the target problem. Finally, the comparison between the exact method with the best formulation and the Scatter Search revealed that the heuristic approach consistently found near-optimal solutions in smaller instances. Additionally, in larger instances, the SS was able to find solutions that outperformed the exact method with much smaller computing times. Overall, the experimentation demonstrated the competitiveness of the Scatter Search algorithm in addressing large real-world scenarios in the waste collection problem. Additionally, it confirmed the ability of valid inequalities to enhance the formulation model, facilitating its application to more complex and realistic instances of waste collection challenges.

Future research lines could explore hybrid approaches that combine Scatter Search with other metaheuristics and perform a comparative analysis between Scatter Search and other computational intelligent mataheuristics. Additionally, dynamic models can be applied for real-time adjustments to planned routes based on smart bins information, integrating smart city technologies like IoT to enhance decision-making. Another critical research line is to incorporate sustainability assessments to evaluate the environmental impacts of collection routes alongside travel costs, thereby addressing the waste collection problem in a multi-objective manner. Finally, engaging with practitioners to discuss these solutions and gather feedback will be essential to ensure the practicality and effectiveness of the proposed strategies.

Author Contributions: Conceptualization, D.R., B.G.L., M.F. and M.M.B.; methodology, D.R. and B.G.L.; software, D.R. and B.G.L.; validation, D.R. and B.G.L.; formal analysis, D.R. and B.G.L.; investigation, D.R. and B.G.L.; resources, D.R. and B.G.L.; data curation, D.R. and B.G.L.; writing—original draft preparation, D.R., B.G.L. and M.F.; writing—review and editing, D.R., B.G.L., M.F. and M.M.B.; visualization, D.R. and B.G.L.; supervision, D.R., B.G.L., M.F. and M.M.B.; project administration, D.R., B.G.L., M.F. and M.M.B.; funding acquisition, D.R., B.G.L. and M.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Consejería de Economía, Industria, Comercio y Conocimiento of the Government of the Canary Islands through the direct grant awarded to the ULPGC called "Support for R+D+i activity. Campus of International Excellence CEI CANARIAS-ULPGC", by CONICET under research project PIBAA 0466CO, by the Agencia I+D+i of Argentina under research project PICT-2021-I-INVI00217, and by 319RT0574—Red iberoamericana Industria 4.0 of CYTED.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Instances can be retrieved from Github (https://github.com/diegorossit/Urban_Science_waste_collection_BBCA.git, accessed on 27 October 2024). Other datasets are available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Comparison Between Solvers and Valid Inequalities for Exact Method

This Appendix presents the detailed results of the comparison between the commercial solvers Gurobi and CPLEX and valid inequalities for the exact method. In Table A1, the results for each instance and solver are presented. For each instance, the following data are outlined: Upper Bound (UB) (i.e., the incumbent integer feasible solution obtained by the solver), computing time in seconds, optimality gap, and the lower bound (LB) (i.e., solution of the relaxed problem). The optimality gap is estimated as in Gurobi Optimization, LLC [32] with the information of the UB and LB with Equation (A1).

$$opt.\ gap = \left(\frac{UB - LB}{LB}\right)\% \tag{A1}$$

From the detailed results, it can be observed that CPLEX occasionally reaches the optimal solution but lacks sufficient information to confirm this due to a weak lower bound. In contrast, a key strength of Gurobi lies in its ability to generate competitive lower bounds, which significantly reduces computing times. This capability enables Gurobi to converge more effectively on optimal solutions, enhancing its performance relative to CPLEX.

Table A1. Detailed results for the comparison between solvers and VIs for the exact method.

Gurobi												
	15_1				**15_2**				**15_3**			
Formulations	**UB**	**Comp. Time (sec)**	**opt. gap**	**LB**	**UB**	**Comp. Time (sec)**	**opt. gap**	**LB**	**UB**	**Comp. Time (sec)**	**opt. gap**	**LB**
Model (1)	34.93	7203.08	35%	22.64	33.37	7204.15	36%	21.35	35.31	7203.75	34%	23.33
Model (1) + Equation (2)	34.59	99.20	0%	34.59	33.35	161.67	0%	33.35	35.22	129.31	0%	35.22
Model (1) + Equation (3)	34.59	15.40	0%	34.59	33.35	21.23	0%	33.35	35.22	50.60	0%	35.22
Model (1) + Equations (2) and (3)	34.59	7.63	0%	34.59	33.35	39.64	0%	33.35	35.22	45.59	0%	35.22
Model (1) + Equation (4)	34.93	7203.70	36%	22.19	33.52	7204.27	38%	20.65	35.71	7203.84	36%	22.91
Model (1) + Equations (2) and (4)	34.59	104.81	0%	34.59	33.35	115.65	0%	33.35	35.22	126.38	0%	35.22
Model (1) + Equations (3) and (4)	34.59	24.80	0%	34.59	33.35	22.63	0%	33.35	35.22	17.45	0%	35.22
Model (1) + Equations (2) and (4)	34.59	10.91	0%	34.59	33.35	46.10	0%	33.35	35.22	23.84	0%	35.22
CPLEX												
Formulations	**UB**	**Comp. Time (sec)**	**opt. gap**	**LB**	**UB**	**Comp. Time (sec)**	**opt. gap**	**LB**	**UB**	**Comp. Time (sec)**	**opt. gap**	**LB**
Model (1)	34.59	7203.66	42%	20.08	33.35	7203.32	44%	18.57	35.22	7209.65	37%	22.03
Model (1) + Equation (2)	34.59	7208.26	20%	27.60	33.35	7203.53	39%	20.21	35.22	7205.12	19%	28.36
Model (1) + Equation (3)	35.10	7203.38	43%	19.87	33.49	7206.26	45%	18.44	35.22	7206.83	39%	21.37
Model (1) + Equations (2) and (3)	35.10	7210.45	20%	28.12	35.16	7205.62	39%	21.34	35.22	7213.45	19%	28.43
Model (1) + Equation (4)	34.59	7203.27	42%	20.08	33.35	7204.08	44%	18.57	35.22	7203.04	37%	22.03
Model (1) + Equations (2) and (4)	34.59	7208.77	20%	27.62	33.64	7203.77	40%	20.20	35.22	7210.53	19%	28.36
Model (1) + Equations (3) and (4)	35.10	7202.95	43%	19.87	33.49	7205.99	45%	18.45	35.69	7209.10	40%	21.38
Model (1) + Equations (2) and (4)	35.10	7210.04	20%	28.14	33.93	7205.62	37%	21.33	35.22	7205.89	19%	28.42

Appendix B. Scatter Search. Statistical Summary of the Computational Experiments

This appendix contains the box-and-whisker plots of the 36 SS algorithm configurations considered with the instances of 15 (see Figures A1–A3) and 30 (see Figures A4–A6) collection points, and the corresponding statistical summary tables. Thus, Tables A2–A4 show the statistical summary of the experiments performed with the 15-1, 15-2, and 15-3 instances, respectively, and Tables A5–A7 show the statistical summary of the experiments performed with the 30-1, 30-2, and 30-3 instances, respectively. In each table, from left to right is reported, for each configuration of the SS algorithm: the minimum (the best value of the objective function found), the first quartile, the median, the second quartile, the (pseudo) median, the lower bound and the upper bound of a 95% confidence interval for the (pseudo) median, the total runtime (in seconds), of the 31 runs performed in each case, and the mean runtime of each run (in seconds). The pseudomedian of a distribution F is the median of the distribution of $(u + v)/2$, where u and v are independent, each with [48].

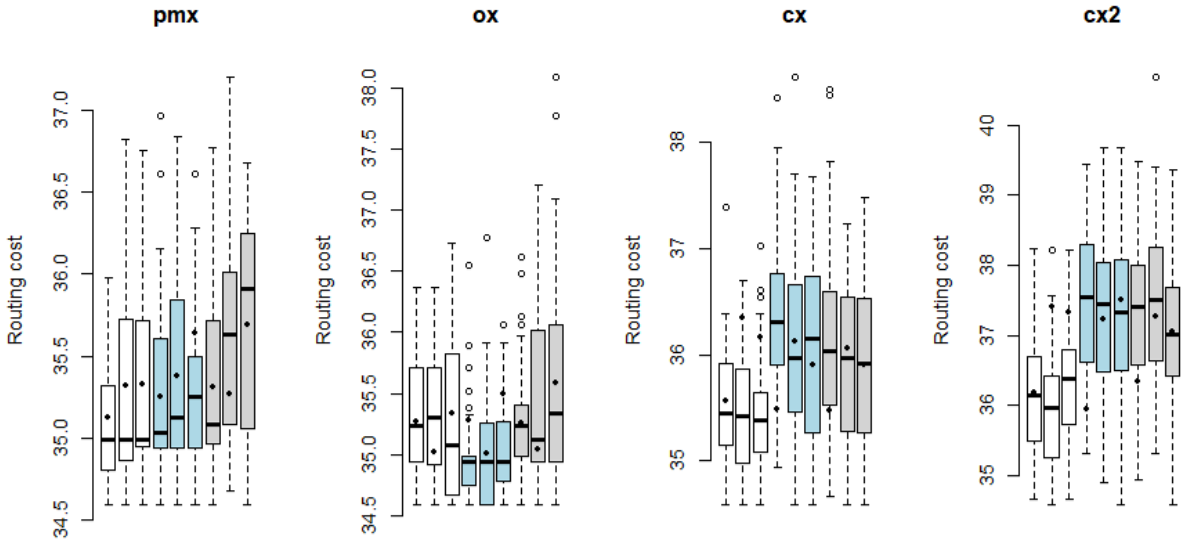


Figure A1. Instance 15-1. Box-and-whisker plots of the results grouped by combination methods, improvement method: EXC (white), INS (blue), and INV (gray), and local search sizes: 10, 20, and 30, from left to right.

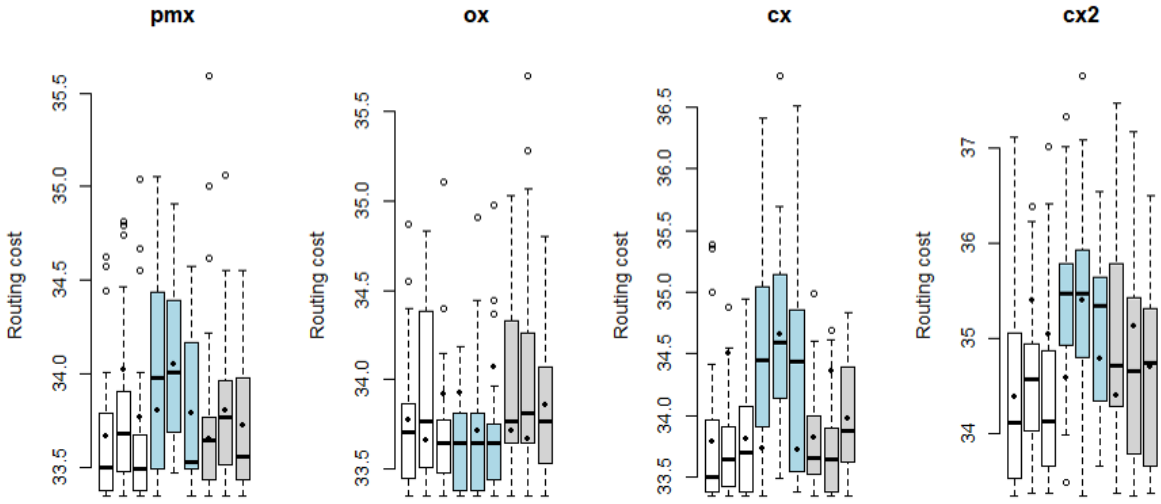


Figure A2. Instance 15-2. Box-and-whisker plots of the results grouped by combination methods, improvement method: EXC (white), INS (blue), and INV (gray), and local search sizes: 10, 20, and 30, from left to right.

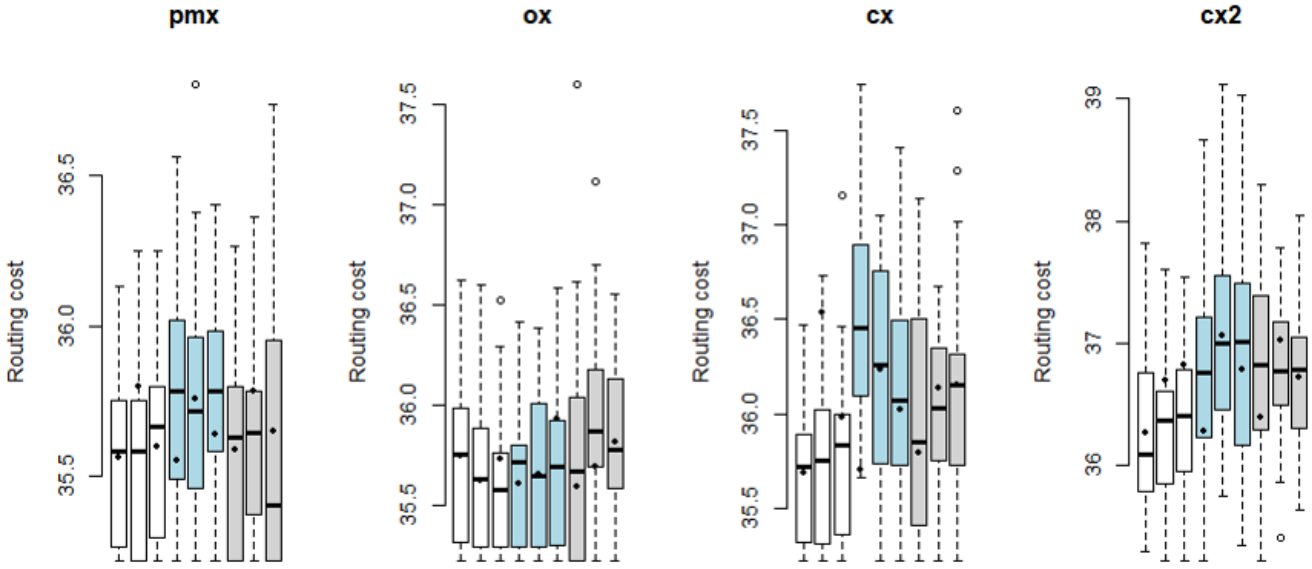


Figure A3. Instance 15-3. Box-and-whisker plots of the results grouped by combination methods, improvement method: EXC (white), INS (blue), and INV (gray), and local search sizes: 10, 20, and 30, from left to right.

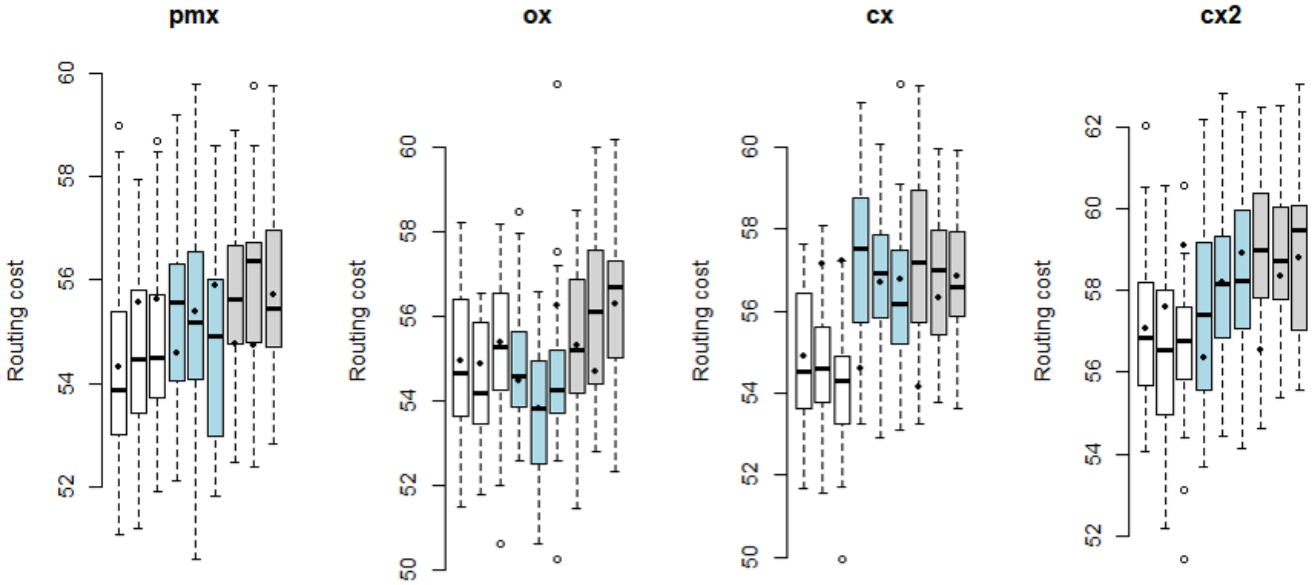


Figure A4. Instance 30-1. Box-and-whisker plots of the results grouped by combination methods, improvement method: EXC (white), INS (blue), and INV (gray), and local search sizes: 10, 20, and 30, from left to right.

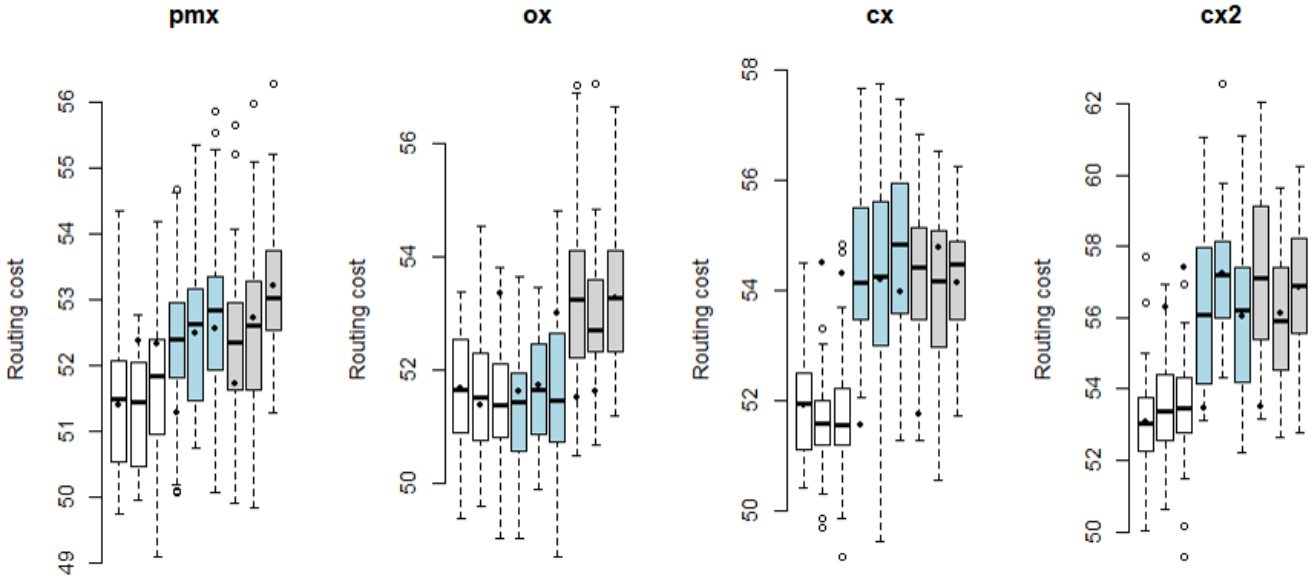


Figure A5. Instance 30-2. Box-and-whisker plots of the results grouped by combination methods, improvement method: EXC (white), INS (blue), and INV (gray), and local search sizes: 10, 20, and 30, from left to right.

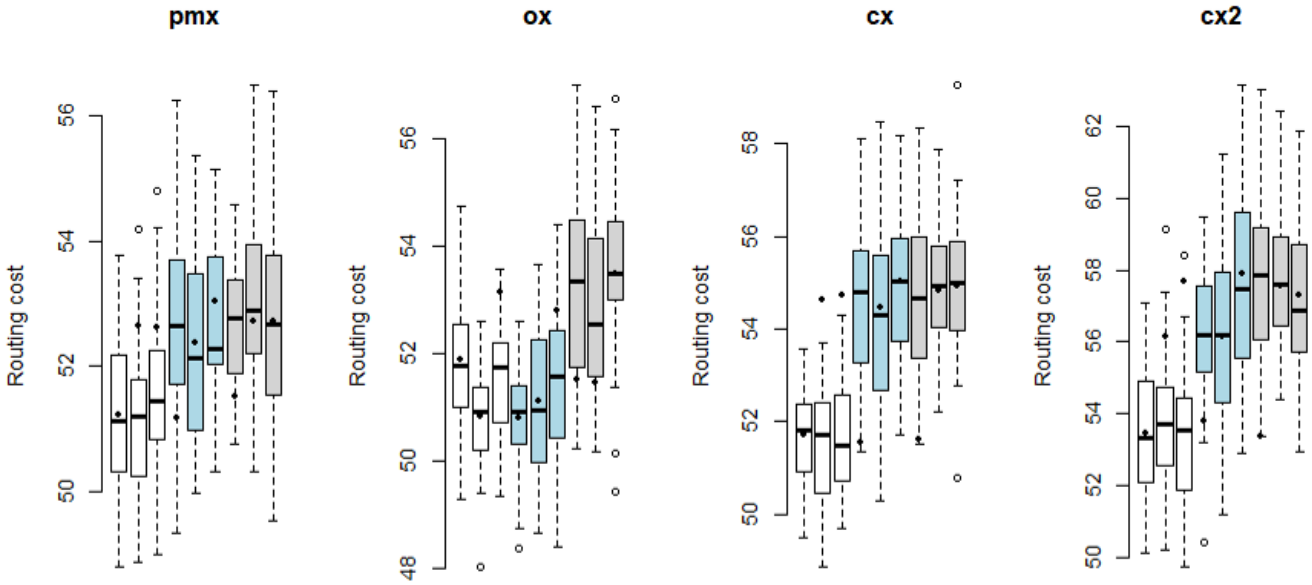


Figure A6. Instance 30-3. Box-and-whisker plots of the results grouped by combination methods, improvement method: EXC (white), INS (blue), and INV (gray), and local search sizes: 10, 20, and 30, from left to right.

Table A2. Instance 15-1. Statistical summary (runtimes in seconds).

Combin. Method	Improv. Method	Local Search Size	min	Q1	Median	Q2	(Pseudo) Median	lb	ub	Total Runtimes	Mean Runtimes
PMX	EXC	10	**34.59096**	34.80424	34.98869	35.31725	35.10107	34.95119	35.31720	243.68339	7.86075
PMX	EXC	20	**34.59096**	34.86188	34.98869	35.72651	35.21642	34.95989	35.40944	207.04524	6.67888
PMX	EXC	30	**34.59096**	34.94258	34.98869	35.71786	35.24804	34.98868	35.49298	191.32109	6.17165
PMX	INS	10	**34.59096**	34.93681	35.02904	35.60258	35.23934	35.00881	35.49601	263.45977	8.49870
PMX	INS	20	**34.59096**	34.93681	35.12703	35.83891	35.35184	35.08669	35.59676	229.96111	7.41810
PMX	INS	30	**34.59096**	34.93681	35.24808	35.49594	35.23083	35.04350	35.43840	217.88484	7.02854
PMX	INV	10	**34.59096**	34.95987	35.08092	35.71498	35.30537	35.03196	35.52474	267.14046	8.61743
PMX	INV	20	34.67166	35.08092	35.62852	36.01472	35.59175	35.37194	35.85050	231.33146	7.46231
PMX	INV	30	**34.59096**	35.05786	35.91096	36.25105	35.67753	35.44991	35.97443	219.01970	7.06515
OX	EXC	10	**34.59096**	34.93681	35.23079	35.70922	35.25384	35.07809	35.44986	231.14924	7.45643
OX	EXC	20	**34.59096**	34.91952	35.30572	35.71498	35.27983	35.08385	35.48148	194.77790	6.28316
OX	EXC	30	**34.59096**	34.67166	35.08092	35.82450	35.23946	34.95993	35.48443	182.80096	5.89681
OX	INS	10	**34.59096**	34.74660	34.93681	34.98581	34.95990	34.80425	35.13281	264.53177	8.53328
OX	INS	20	**34.59096**	**34.59096**	34.93681	35.25673	**34.93688**	**34.78975**	**35.10975**	229.01960	7.38773
OX	INS	30	**34.59096**	34.78694	34.93681	35.26826	34.96274	34.90220	35.15868	216.53589	6.98503
OX	INV	10	**34.59096**	34.98581	35.23079	35.40371	35.24462	35.10683	35.52475	267.98669	8.64473
OX	INV	20	**34.59096**	34.93681	35.12703	36.02048	35.45007	35.04638	35.79576	232.05914	7.48578
OX	INV	30	**34.59096**	34.93681	35.33454	36.06372	35.46722	35.15582	35.83898	219.81882	7.09093
CX	EXC	10	**34.59096**	35.15585	35.44983	35.91961	35.52714	35.32585	35.74099	222.05486	7.16306
CX	EXC	20	**34.59096**	34.98293	35.42101	35.86773	35.45504	35.22783	35.72363	189.66683	6.11828
CX	EXC	30	**34.59096**	35.08092	35.38642	35.64005	35.41524	35.22788	35.62569	178.91450	5.77144
CX	INS	10	34.93681	35.90520	36.30869	36.76118	36.33454	36.02624	36.65452	258.99309	8.35462
CX	INS	20	**34.59096**	35.45847	35.97437	36.66031	36.05195	35.73220	36.42977	224.35371	7.23722
CX	INS	30	**34.59096**	35.26249	36.15306	36.73236	36.07049	35.68336	36.42406	212.69208	6.86103
CX	INV	10	34.67166	35.52476	36.03201	36.59114	36.08384	35.78712	36.43552	259.40189	8.36780
CX	INV	20	**34.59096**	35.27402	35.96861	36.54503	35.91387	35.59974	36.23950	226.78883	7.31577
CX	INV	30	**34.59096**	35.26826	35.91673	36.52773	35.90070	35.57952	36.19918	216.42255	6.98137
CX2	EXC	10	34.67166	35.49594	36.15306	36.69489	36.18476	35.85039	36.49609	232.64759	7.50476
CX2	EXC	20	**34.59096**	35.26537	35.96861	36.42974	35.89079	35.65446	36.24241	196.65155	6.34360
CX2	EXC	30	34.67166	35.72939	36.38939	36.79577	36.28275	36.00893	36.66612	185.13147	5.97198
CX2	INS	10	35.32878	36.62572	37.54800	38.30599	37.45871	36.94277	37.86217	266.94226	8.61104
CX2	INS	20	34.90223	36.48162	37.44424	38.03219	37.31848	36.66893	37.76419	230.21344	7.42624
CX2	INS	30	**34.59096**	36.49315	37.32896	38.08983	37.28032	36.84764	37.73815	217.89116	7.02875
CX2	INV	10	34.93681	36.59114	37.40389	37.99760	37.31742	36.89372	37.74691	269.12641	8.68150
CX2	INV	20	35.32878	36.64014	37.51341	38.25411	37.47034	36.96007	38.00915	232.88722	7.51249
CX2	INV	30	**34.59096**	36.42398	37.02345	37.68057	37.10931	36.65164	37.57973	220.83009	7.12355

Table A3. Instance 15-2. Statistical summary (runtimes in seconds).

Combin. Method	Improv. Method	Local Search Size	min	Q1	Median	Q2	(Pseudo) Median	lb	ub	Total Runtimes	Mean Runtimes
PMX	EXC	10	**33.34589**	33.37472	33.49576	33.78974	33.60532	33.49580	33.73214	227.75893	7.34706
PMX	EXC	20	**33.34589**	33.47847	33.68022	33.90214	33.72628	33.57073	33.96260	193.70328	6.24849
PMX	EXC	30	**33.34589**	33.37472	33.49000	33.67157	**33.50802**	**33.43241**	**33.72633**	182.34684	5.88216
PMX	INS	10	**33.34589**	33.49288	33.97996	34.43533	33.98722	33.74071	34.24511	263.04966	8.48547
PMX	INS	20	33.46694	33.68886	34.00878	34.39210	34.05206	33.87327	34.21623	229.88504	7.41565
PMX	INS	30	**33.34589**	33.49000	33.52459	34.17018	33.79832	33.56488	33.95395	217.47901	7.01545
PMX	INV	10	**33.34589**	33.43236	33.64563	33.76668	33.66869	33.55622	33.78966	267.02398	8.61368
PMX	INV	20	**33.34589**	33.51017	33.76668	33.95978	33.74652	33.65138	33.87620	230.50383	7.43561
PMX	INV	30	**33.34589**	33.43236	33.55917	33.97707	33.70549	33.52462	33.85884	219.22672	7.07183
OX	EXC	10	**33.34589**	33.44389	33.70328	33.86179	33.71767	33.57075	33.88770	229.91375	7.41657
OX	EXC	20	**33.34589**	33.50729	33.76668	34.38345	33.94542	33.67742	34.10389	195.04656	6.29182
OX	EXC	30	**33.34589**	33.47271	33.64563	33.77245	33.64563	33.56782	33.76668	182.48869	5.88673
OX	INS	10	**33.34589**	33.37472	33.64563	33.81280	33.66285	33.53901	33.76086	264.40133	8.52908
OX	INS	20	**33.34589**	33.37472	33.64563	33.81280	33.66868	33.55627	33.80125	229.29051	7.39647
OX	INS	30	**33.34589**	33.43812	33.63987	33.74939	33.59379	33.51877	33.70334	217.31168	7.01005
OX	INV	10	**33.34589**	33.64275	33.76668	34.32869	33.88645	33.70616	34.06637	268.06627	8.64730
OX	INV	20	**33.34589**	33.64275	33.81280	34.25664	33.91238	33.74359	34.29691	232.63654	7.50440
OX	INV	30	**33.34589**	33.52747	33.76668	34.06930	33.81849	33.66584	34.01446	220.26735	7.10540
CX	EXC	10	**33.34589**	33.37472	33.49576	33.95978	33.66584	33.51021	33.89634	222.08047	7.16389
CX	EXC	20	**33.34589**	33.42083	33.63987	33.91079	33.68013	33.55628	33.83583	189.45881	6.11157
CX	EXC	30	**33.34589**	33.37472	33.70328	34.07507	33.74260	33.57067	34.00014	177.81074	5.73583
CX	INS	10	**33.34589**	33.90502	34.45262	35.04057	34.46412	34.17891	34.76682	259.44712	8.36926
CX	INS	20	33.49000	34.14135	34.59673	35.14144	34.64280	34.35470	34.94826	225.07482	7.26048
CX	INS	30	33.37472	33.54188	34.43533	34.85323	34.30566	34.03188	34.62557	212.64264	6.85944
CX	INV	10	**33.34589**	33.52459	33.65140	34.00013	33.76089	33.61400	33.91940	260.00214	8.38717
CX	INV	20	**33.34589**	33.37472	33.63987	33.89926	33.66286	33.51309	33.88201	227.02476	7.32338
CX	INV	30	**33.34589**	33.62546	33.87620	34.39786	33.97136	33.76384	34.12980	216.41019	6.98097
CX2	EXC	10	**33.34589**	33.53611	34.11830	35.05786	34.35471	33.93387	34.68322	231.42931	7.46546
CX2	EXC	20	33.37472	34.02607	34.57367	34.93681	34.56799	34.28830	34.84753	196.16718	6.32797
CX2	EXC	30	33.37472	33.66004	34.13559	34.86476	34.28402	33.97416	34.69187	183.38571	5.91567
CX2	INS	10	33.49000	34.93393	35.46712	35.78992	35.38606	35.08090	35.68042	265.78683	8.57377
CX2	INS	20	**33.34589**	34.79847	35.46712	35.92826	35.38210	35.03192	35.78415	230.63251	7.43976
CX2	INS	30	33.65140	34.34310	35.34031	35.64293	35.13424	34.79847	35.45847	218.36783	7.04412
CX2	INV	10	33.37472	34.28834	34.71777	35.78415	35.01603	34.58802	35.43836	268.83379	8.67206
CX2	INV	20	**33.34589**	33.79262	34.66013	35.42677	34.72070	34.28255	35.20772	233.39165	7.52876
CX2	INV	30	33.37472	33.66293	34.74660	35.31437	34.72647	34.33444	35.06652	221.32340	7.13946

Table A4. Instance 15-3. Statistical summary (runtimes in seconds).

Combin. Method	Improv. Method	Local Search Size	min	Q1	Median	Q2	(Pseudo) Median	lb	ub	Total Runtimes	Mean Runtimes
PMX	EXC	10	**35.21926**	35.26249	35.58240	35.74957	**35.54210**	**35.45850**	**35.67456**	227.71527	7.34565
PMX	EXC	20	**35.21926**	**35.21926**	35.58240	35.74957	35.54489	35.43258	35.66306	194.81453	6.28434
PMX	EXC	30	**35.21926**	35.29708	35.66310	35.79568	35.55641	35.48434	35.72358	183.07657	5.90570
PMX	INS	10	**35.21926**	35.49306	35.78415	36.02048	35.79617	35.63136	35.94266	266.05218	8.58233
PMX	INS	20	**35.21926**	35.45847	35.71498	35.95996	35.74837	35.58237	35.90234	232.48314	7.49946
PMX	INS	30	**35.21926**	35.58240	35.78415	35.98302	35.78413	35.64009	35.90807	219.23577	7.07212
PMX	INV	10	**35.21926**	**35.21926**	35.62852	35.79568	35.58237	35.46714	35.72358	268.63915	8.66578
PMX	INV	20	**35.21926**	35.37201	35.64581	35.78415	35.62559	35.50167	35.74958	231.71591	7.47471
PMX	INV	30	**35.21926**	**35.21926**	35.40371	35.95420	35.60547	35.44114	35.79568	219.80032	7.09033
OX	EXC	10	**35.21926**	35.31437	35.74957	35.98302	35.74375	35.56508	35.89941	242.16606	7.81181
OX	EXC	20	**35.21926**	35.28843	35.62852	35.88214	35.57376	35.46711	35.76973	194.91102	6.28745
OX	EXC	30	**35.21926**	35.28843	35.57664	35.76109	35.55928	35.44405	35.70923	181.91904	5.86836
OX	INS	10	**35.21926**	35.28843	35.71498	35.79568	35.57664	35.48444	35.74958	264.82807	8.54284
OX	INS	20	**35.21926**	35.28843	35.64581	36.00896	35.64977	35.48437	35.79566	229.24999	7.39516
OX	INS	30	**35.21926**	35.29708	35.68616	35.91961	35.68610	35.50164	35.82444	217.21216	7.00684
OX	INV	10	**35.21926**	**35.21926**	35.66310	36.04066	35.69477	35.47583	35.87061	269.37530	8.68953
OX	INV	20	**35.21926**	35.68904	35.86485	36.17612	35.90231	35.74958	36.09830	233.15601	7.52116
OX	INV	30	**35.21926**	35.57952	35.77262	36.13289	35.80140	35.66313	35.97141	220.73601	7.12052
CX	EXC	10	**35.21926**	35.31437	35.71498	35.88791	35.68137	35.53626	35.80724	223.12399	7.19755
CX	EXC	20	**35.21926**	35.30572	35.74957	36.02048	35.68054	35.53921	35.85622	190.05782	6.13090
CX	EXC	30	**35.21926**	35.35472	35.83603	35.99743	35.79851	35.59677	35.92537	178.46300	5.75687
CX	INS	10	35.66310	36.09254	36.45856	36.89664	36.51188	36.30293	36.73813	258.61783	8.34251
CX	INS	20	**35.21926**	35.73227	36.25682	36.76118	36.24549	36.01479	36.44419	222.54077	7.17873
CX	INS	30	**35.21926**	35.72939	36.06660	36.49891	36.09080	35.90226	36.30580	211.01199	6.80684
CX	INV	10	**35.21926**	35.40371	35.84756	36.50468	35.97941	35.72075	36.19922	259.19600	8.36116
CX	INV	20	**35.21926**	35.74957	36.02625	36.35193	36.03029	35.87642	36.17616	226.44557	7.30470
CX	INV	30	**35.21926**	35.72363	36.15306	36.31734	36.12424	35.93112	36.32315	215.50162	6.95167
CX2	EXC	10	35.28843	35.78415	36.08965	36.75254	36.23662	35.97145	36.47011	231.04708	7.45313
CX2	EXC	20	**35.21926**	35.84468	36.36057	36.60843	36.27787	36.02330	36.49895	194.29054	6.26744
CX2	EXC	30	**35.21926**	35.94843	36.40668	36.78424	36.37367	36.13575	36.64306	182.08140	5.87359
CX2	INS	10	**35.21926**	36.23088	36.75830	37.21079	36.69778	36.38939	37.00328	264.41698	8.52958
CX2	INS	20	35.74957	36.45856	37.00040	37.55952	37.02864	36.72077	37.32900	228.83163	7.38167
CX2	INS	30	35.34031	36.16171	37.00616	37.49612	37.01976	36.59396	37.36073	216.29497	6.97726
CX2	INV	10	**35.21926**	36.28852	36.82747	37.39236	36.82747	36.53062	37.11280	267.69497	8.63532
CX2	INV	20	35.40371	36.49891	36.76983	37.17332	36.81017	36.59400	37.00332	231.97497	7.48306
CX2	INV	30	35.62852	36.30005	36.78136	37.04363	36.70892	36.48733	36.93411	219.82824	7.09123

Table A5. Instance 30-1. Statistical summary (runtimes in seconds).

Combin. Method	Improv. Method	Local Search Size	min	Q1	Median	Q2	(Pseudo) Median	lb	ub	Total Runtimes	Mean Runtimes
PMX	EXC	10	51.06504	53.01911	53.87221	55.39396	54.18930	53.52346	54.83772	881.08183	28.42199
PMX	EXC	20	51.18033	53.42548	54.44863	55.79457	54.54374	53.92409	55.20086	737.33660	23.78505
PMX	EXC	30	51.91238	53.73099	54.49474	55.71675	54.66767	54.02208	55.32479	691.54074	22.30777
PMX	INS	10	52.11989	54.03937	55.56688	56.29606	55.58562	54.85789	56.19518	1037.05853	33.45350
PMX	INS	20	50.59814	54.06531	55.16339	56.55256	55.33776	54.59850	56.14619	894.97129	28.87004
PMX	INS	30	51.80286	52.98452	54.90400	56.01361	54.74001	53.99612	55.47184	846.55571	27.30825
PMX	INV	10	52.45998	54.74837	55.63029	56.65344	55.62599	55.00486	56.22979	1049.03119	33.83972
PMX	INV	20	52.38505	54.78295	56.35658	56.71108	55.94052	55.19506	56.55546	907.41383	29.27141
PMX	INV	30	52.83465	54.71378	55.44007	56.96470	55.59566	55.01924	56.40270	861.09960	27.77741
OX	EXC	10	51.47430	53.61859	54.62732	56.40558	54.83195	54.15177	55.68505	905.71830	29.21672
OX	EXC	20	51.75675	53.43413	54.18348	55.82916	54.48335	53.87801	54.98472	754.31409	24.33271
OX	EXC	30	50.59814	54.25841	55.27291	56.51798	55.40981	54.77719	55.96461	700.88260	22.60912
OX	INS	10	52.56950	53.86068	54.55239	55.61876	54.76566	54.20653	55.36514	1057.72009	34.12000
OX	INS	20	50.60391	52.50321	53.79151	54.93282	**53.78863**	**53.18339**	**54.39675**	902.89111	29.12552
OX	INS	30	50.24653	53.71081	54.22959	55.17780	54.48466	53.95291	55.12304	851.61238	27.47137
OX	INV	10	51.42819	54.16618	55.20374	56.85518	55.38387	54.74837	56.03378	1064.09214	34.32555
OX	INV	20	52.77125	54.39099	56.09719	57.55553	56.22526	55.43138	57.07703	908.57995	29.30903
OX	INV	30	52.29858	54.99911	56.69667	57.31055	56.27156	55.67352	56.98488	859.53252	27.72686
CX	EXC	10	51.66452	53.61570	54.53509	56.41422	54.93138	54.20077	55.50636	871.22559	28.10405
CX	EXC	20	51.57806	53.76557	54.58121	55.61876	54.59850	54.04802	55.12304	725.63137	23.40746
CX	EXC	30	**49.95832**	53.23815	54.28147	54.88671	54.19499	53.60122	54.70222	678.97770	21.90251
CX	INS	10	53.24968	55.71964	57.51518	58.73719	57.18807	56.39117	57.99361	1034.47365	33.37012
CX	INS	20	52.91535	55.81475	56.93300	57.86392	56.76440	56.18077	57.33073	882.38903	28.46416
CX	INS	30	53.11134	55.20086	56.17789	57.46907	56.22405	55.56981	56.95313	832.18669	26.84473
CX	INV	10	53.23815	55.70811	57.17510	58.92453	57.26012	56.43151	57.99938	1031.70938	33.28095
CX	INV	20	53.77422	55.40837	56.99064	57.98208	56.79170	56.10001	57.48638	893.67188	28.82813
CX	INV	30	53.61282	55.86662	56.58715	57.94462	56.78169	56.25282	57.35955	842.95357	27.19205
CX2	EXC	10	54.06819	55.67352	56.84654	58.18960	56.98344	56.27012	57.71981	936.78281	30.21880
CX2	EXC	20	52.18330	54.98182	56.52951	58.01955	56.34582	55.46310	57.23273	768.74164	24.79812
CX2	EXC	30	51.43972	55.82916	56.77160	57.58724	56.66455	55.92714	57.22404	711.29197	22.94490
CX2	INS	10	53.67623	55.57841	57.40566	59.18680	57.41143	56.59291	58.46916	1092.76569	35.25051
CX2	INS	20	54.44287	56.83501	58.17230	59.30209	58.09449	57.32208	58.91300	921.59958	29.72902
CX2	INS	30	54.12007	57.07422	58.22418	59.97073	58.26309	57.54977	59.12628	862.70223	27.82910
CX2	INV	10	54.64461	57.81780	58.97929	60.37134	59.11907	58.35099	59.77475	1097.34856	35.39834
CX2	INV	20	55.37090	57.79475	58.73143	60.02549	58.88416	58.22412	59.56438	931.48914	30.04804
CX2	INV	30	55.55536	57.01370	59.48654	60.08313	58.82654	57.99361	59.69982	873.67884	28.18319

Table A6. Instance 30-2. Statistical summary (runtimes in seconds).

Combin. Method	Improv. Method	Local Search Size	min	Q1	Median	Q2	(Pseudo) Median	lb	ub	Total Runtimes	Mean Runtimes
PMX	EXC	10	49.74504	50.53474	51.48583	52.05937	51.35325	50.99587	51.75098	866.66081	27.95680
PMX	EXC	20	49.94679	50.46845	51.42819	52.03919	**51.25815**	**50.92378**	**51.57232**	722.85330	23.31785
PMX	EXC	30	49.08792	50.93535	51.83168	52.38793	51.71461	51.29557	52.11989	675.78662	21.79957
PMX	INS	10	50.06207	51.80286	52.39657	52.93841	52.38352	51.93826	52.82593	1017.67223	32.82814
PMX	INS	20	50.73649	51.46566	52.61561	53.16321	52.45854	52.03055	52.92112	876.12927	28.26223
PMX	INS	30	50.06207	51.92103	52.83465	53.33614	52.76837	52.19771	53.14304	827.92987	26.70742
PMX	INV	10	49.89491	51.62994	52.33317	52.94706	52.26113	51.81724	52.71078	1028.13709	33.16571
PMX	INV	20	49.83727	51.60976	52.59256	53.28138	52.50612	52.08530	52.94996	886.08724	28.58346
PMX	INV	30	51.27832	52.52627	53.02487	53.73387	53.17474	52.71361	53.61570	837.62361	27.02012
OX	EXC	10	49.37037	50.88635	51.65876	52.55221	51.67893	51.26391	52.10836	895.56372	28.88915
OX	EXC	20	49.58941	50.75954	51.50312	52.30435	51.55788	51.14574	52.01902	740.58223	23.88975
OX	EXC	30	49.01299	50.79989	51.38208	52.10836	51.49444	51.06503	51.97289	683.61961	22.05225
OX	INS	10	49.02452	50.57221	51.41666	51.94985	51.35324	50.95548	51.73083	1038.97999	33.51548
OX	INS	20	49.88338	50.86618	51.64723	52.45710	51.71784	51.26103	52.12566	885.74781	28.57251
OX	INS	30	**48.69596**	50.71919	51.44548	52.63579	51.64291	51.09963	52.13142	834.56697	26.92152
OX	INV	10	50.48286	52.20636	53.24968	54.12295	53.29435	52.62714	53.97597	1062.18632	34.26407
OX	INV	20	50.67308	52.33029	52.69055	53.60417	52.87072	52.53198	53.26700	901.22178	29.07167
OX	INV	30	51.19186	52.32164	53.26120	54.10278	53.19059	52.66749	53.75693	840.90345	27.12592
CX	EXC	10	50.41945	51.11692	51.92967	52.49457	51.88065	51.52046	52.26403	862.95733	27.83733
CX	EXC	20	49.67587	51.18321	51.57806	52.00173	51.57229	51.26679	51.86627	714.57964	23.05096
CX	EXC	30	49.16286	51.17745	51.53771	52.20924	51.63646	51.30712	52.05644	667.59255	21.53524
CX	INS	10	52.06225	53.46872	54.12007	55.50060	54.42113	53.91260	55.06257	1018.25457	32.84692
CX	INS	20	49.43378	52.99605	54.24688	55.59282	54.28946	53.56378	54.96741	870.52058	28.08131
CX	INS	30	51.26679	53.56959	54.81754	55.93291	54.81466	54.18348	55.38243	818.01102	26.38745
CX	INV	10	51.26679	53.46007	54.40252	55.13169	54.28145	53.79733	54.85797	1031.82092	33.28455
CX	INV	20	50.55203	52.97299	54.14889	55.07116	53.98365	53.41391	54.56969	878.18085	28.32841
CX	INV	30	51.72216	53.48024	54.45439	54.87518	54.18346	53.72225	54.63600	829.36701	26.75377
CX2	EXC	10	50.03902	52.26400	53.00758	53.74828	53.02915	52.52047	53.52927	926.92980	29.90096
CX2	EXC	20	50.64426	52.53492	53.39378	54.41693	53.42838	52.90093	53.94423	753.77314	24.31526
CX2	EXC	30	49.29544	52.77989	53.46295	54.33335	53.53701	53.03640	54.04514	694.61395	22.40690
CX2	INS	10	53.13439	54.16042	56.07990	57.96191	56.13610	55.41990	57.08287	1075.41473	34.69080
CX2	INS	20	54.31029	55.98191	57.20392	58.14060	57.16501	56.58138	57.78034	901.75945	29.08901
CX2	INS	30	52.19483	54.17771	56.18942	57.41719	56.06261	55.23833	56.99064	844.93842	27.25608
CX2	INV	10	53.14016	55.37955	57.09440	59.13492	57.36532	56.37963	58.25307	1078.62075	34.79422
CX2	INV	20	52.62138	54.54662	55.90121	57.41719	55.97614	55.17492	56.76584	908.90025	29.31936
CX2	INV	30	52.78278	55.54959	56.89841	58.22418	56.89121	56.17213	57.59300	852.43323	27.49785

Table A7. Instance 30-3. Statistical summary (runtimes in seconds).

Combin. Method	Improv. Method	Local Search Size	min	Q1	Median	Q2	(Pseudo) Median	lb	ub	Total Runtimes	Mean Runtimes
PMX	EXC	10	48.80548	50.31282	51.12269	52.18906	51.19762	50.66731	51.71063	878.12475	28.32660
PMX	EXC	20	48.86888	50.24653	51.19762	51.78557	51.08802	50.69614	51.61267	737.05760	23.77605
PMX	EXC	30	49.00146	50.82871	51.43972	52.25535	51.50893	51.06798	51.91244	687.98596	22.19310
PMX	INS	10	49.34155	51.71928	52.65020	53.71081	52.75107	51.96426	53.35920	1034.64337	33.37559
PMX	INS	20	49.98138	50.97858	52.12566	53.48024	52.35316	51.73365	52.97304	889.41709	28.69087
PMX	INS	30	50.32146	52.02190	52.27553	53.74540	52.69156	52.18037	53.16895	839.41356	27.07786
PMX	INV	10	50.75378	51.88068	52.77701	53.37937	52.60985	52.18330	53.03928	1044.30079	33.68712
PMX	INV	20	50.32146	52.21212	52.89230	53.93562	52.99758	52.47434	53.56091	898.29205	28.97716
PMX	INV	30	49.53177	51.54924	52.67902	53.77422	52.72513	52.09107	53.35055	849.39456	27.39982
OX	EXC	10	49.30696	50.99876	51.77981	52.55509	51.83583	51.31862	52.39653	902.29349	29.10624
OX	EXC	20	**48.04460**	50.21194	50.92094	51.39360	**50.84456**	**50.44251**	**51.18033**	746.78063	24.08970
OX	EXC	30	49.35884	50.72784	51.75675	52.20636	51.53190	51.08230	51.93259	693.01672	22.35538
OX	INS	10	48.37316	50.33587	50.93247	51.41378	50.88780	50.50015	51.25238	1047.95678	33.80506
OX	INS	20	48.67290	49.97561	50.95552	52.28129	51.12845	50.58085	51.65011	898.56316	28.98591
OX	INS	30	48.41928	50.44251	51.58959	52.44845	51.47142	50.92670	52.00749	843.49820	27.20962
OX	INV	10	50.23500	51.75387	53.35920	54.48898	53.15313	52.50898	53.77134	1059.09986	34.16451
OX	INV	20	50.18312	51.58670	52.55797	54.14025	52.80439	52.11989	53.38225	902.16492	29.10209
OX	INV	30	49.44530	53.01046	53.50907	54.46016	53.54265	53.01046	54.09119	850.08570	27.42212
CX	EXC	10	49.48565	50.92382	51.81439	52.36487	51.75675	51.25526	52.14295	868.52623	28.01698
CX	EXC	20	48.88041	50.47421	51.70487	52.40234	51.55848	51.08527	52.07383	723.63057	23.34292
CX	EXC	30	49.69893	50.71055	51.46854	52.58679	51.56590	51.08810	52.04494	674.05238	21.74363
CX	INS	10	51.36478	53.25832	54.80025	55.67640	54.65565	53.92409	55.41418	1034.39624	33.36762
CX	INS	20	50.30417	52.67326	54.29876	55.59859	54.42990	53.52059	55.25562	876.13563	28.26244
CX	INS	30	51.71063	53.73387	55.01352	55.96750	54.90544	54.22383	55.49483	825.90955	26.64224
CX	INV	10	51.52042	53.36208	54.66767	55.97614	54.69505	54.03649	55.37667	1040.66017	33.56968
CX	INV	20	52.21788	54.01920	54.92706	55.78592	55.00488	54.51204	55.53518	885.35162	28.55973
CX	INV	30	50.79989	53.96155	54.99046	55.87527	54.94723	54.32758	55.52077	836.78412	26.99304
CX2	EXC	10	50.10242	52.08531	53.34190	54.89536	53.43557	52.66749	54.18924	937.06006	30.22774
CX2	EXC	20	50.20041	52.57238	53.72234	54.74549	53.70649	53.03064	54.48321	764.68281	24.66719
CX2	EXC	30	49.74504	51.88933	53.54365	54.41981	53.26409	52.64732	54.01055	705.87219	22.77007
CX2	INS	10	50.43675	55.15763	56.20095	57.53824	56.22256	55.56400	56.93588	1071.65137	34.56940
CX2	INS	20	51.17456	54.32470	56.20671	57.95038	56.17357	55.27579	57.12034	904.03375	29.16238
CX2	INS	30	52.90959	55.56688	57.45754	59.60759	57.51518	56.47475	58.59597	845.40639	27.27117
CX2	INV	10	53.34767	56.06549	57.84951	59.19833	57.70828	56.74855	58.57580	1076.21986	34.71677
CX2	INV	20	54.40828	56.43440	57.59588	58.92165	57.78786	56.99640	58.69108	910.16465	29.36015
CX2	INV	30	52.94417	55.71099	56.89265	58.72278	56.89121	56.17213	57.59300	853.29243	27.52556

References

1. Szpilko, D.; De la Torre, A.; Jimenez Naharro, F.; Rzepka, A.; Remiszewska, A. Waste management in the smart city: Current practices and future directions. *Resources* **2023**, *12*, 115. [CrossRef]
2. Hess, C.; Dragomir, A.G.; Doerner, K.F.; Vigo, D. Waste collection routing: A survey on problems and methods. *Cent. Eur. J. Oper. Res.* **2024**, *32*, 399–434. [CrossRef]
3. Ghiani, G.; Laganà, D.; Manni, E.; Musmanno, R.; Vigo, D. Operations research in solid waste management: A survey of strategic and tactical issues. *Comput. Oper. Res.* **2014**, *44*, 22–32. [CrossRef]
4. Yaman, C.; Anil, I.; Jaunich, M.; Blaisi, N.; Alagha, O.; Yaman, A.; Gunday, S. Investigation and modelling of greenhouse gas emissions resulting from waste collection and transport activities. *Waste Manag. Res.* **2019**, *37*, 1282–1290. [CrossRef]
5. Rossit, D.; Nesmachnow, S.; Cavallin, A. Municipal Solid waste management systems: Application of SWOT methodology to analyze an Argentinean case study. In Proceedings of the VI Ibero-American Congress of Smart Cities, Mexico City, Mexico, 13–17 November 2023; pp. 668–680.
6. Bonomo, F.; Durán, G.; Larumbe, F.; Marenco, J. A method for optimizing waste collection using mathematical programming: A Buenos Aires case study. *Waste Manag. Res.* **2012**, *30*, 311–324. [CrossRef]
7. Braier, G.; Durán, G.; Marenco, J.; Wesner, F. An integer programming approach to a real-world recyclable waste collection problem in Argentina. *Waste Manag. Res.* **2017**, *35*, 525–533. [CrossRef]
8. Molfese, S.; Rossit, D.; Frutos, M.; Cavallin, A. Optimization of waste collection through the sequencing of micro-routes and transfer station convenience analysis: An Argentinian case study. *Waste Manag. Res.* **2023**, *41*, 1267–1279. [CrossRef]
9. Rossit, D.; Toutouh, J.; Nesmachnow, S. Exact and heuristic approaches for multi-objective garbage accumulation points location in real scenarios. *Waste Manag.* **2020**, *105*, 467–481. [CrossRef]
10. Cavallin, A.; Rossit, D.; Herran, V.; Rossit, D.; Frutos, M. Application of a methodology to design a municipal waste pre-collection network in real scenarios. *Waste Manag. Res.* **2020**, *38*, 117–129. [CrossRef]
11. Rossit, D.; González Landín, B.; Frutos, M.; Méndez Babey, M. An Allocation-Routing Problem in Waste Management Planning: Exact and Heuristic Resolution Approaches. In *Ibero-American Congress of Smart Cities*; Nesmachnow, S., Hernández, L., Eds.; Springer: Cham, Switzerland, 2023; pp. 92–107. [CrossRef]
12. Hannan, M.; Lipu, M.; Akhtar, M.; Begum, R.; Al Mamun, M.; Hussain, A.; Mia, M.; Basri, H. Solid waste collection optimization objectives, constraints, modeling approaches, and their challenges toward achieving sustainable development goals. *J. Clean. Prod.* **2020**, *277*, 123557. [CrossRef]
13. Toth, P.; Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*; SIAM: Philadelphia, PA, USA, 2014. [CrossRef]
14. Elshaer, R.; Awad, H. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Comput. Ind. Eng.* **2020**, *140*, 106242. [CrossRef]
15. Kalra, M.; Tyagi, S.; Kumar, V.; Kaur, M.; Mashwani, W.; Shah, H.; Shah, K. A comprehensive review on scatter search: Techniques, applications, and challenges. *Math. Probl. Eng.* **2021**, *2021*, 5588486. [CrossRef]
16. Liang, Y.; Minanda, V.; Gunawan, A. Waste collection routing problem: A mini-review of recent heuristic approaches and applications. *Waste Manag. Res.* **2022**, *40*, 519–537. [CrossRef] [PubMed]
17. Zhang, Q.; Wu, L.; Li, J. Application of Improved Scatter Search Algorithm to Reverse Logistics VRP Problem. In Proceedings of the 2023 IEEE 5th International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 14–16 July 2023; pp. 283–288. [CrossRef]
18. Dethloff, J. Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up: Fahrzeugeinsatzplanung und Redistribution: Tourenplanung mit simultaner Auslieferung und Rückholung. *OR-Spektrum* **2001**, *23*, 79–96. [CrossRef]
19. Chu, F.; Labadi, N.; Prins, C. A scatter search for the periodic capacitated arc routing problem. *Eur. J. Oper. Res.* **2006**, *169*, 586–605. [CrossRef]
20. Golden, B.L.; DeArmon, J.S.; Baker, E.K. Computational experiments with algorithms for a class of routing problems. *Comput. Oper. Res.* **1983**, *10*, 47–59. [CrossRef]
21. Belenguer, J.M.; Benavent, E. A cutting plane algorithm for the capacitated arc routing problem. *Comput. Oper. Res.* **2003**, *30*, 705–728. [CrossRef]
22. Yu, H.; Li, X.; Su, S. Material recovery facility location and waste flow assignment problem based on scatter search. In Proceedings of the 2007 International Conference on Machine Learning and Cybernetics, Hong Kong, China, 19–22 August 2007; Volume 4, pp. 2368–2372. [CrossRef]
23. Ortega, M.; Hipólito, J.D.; García, Á. Scatter search for locating a treatment plant and the necessary transfer centers in a reverse network. In *Metaheuristics in the Service Industry*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 63–81. [CrossRef]
24. Mahéo, A.; Rossit, D.; Kilby, P. Solving the integrated bin allocation and collection routing problem for municipal solid waste: A Benders decomposition approach. *Ann. Oper. Res.* **2023**, *322*, 441–465. [CrossRef]
25. Dror, M.; Laporte, G.; Trudeau, P. Vehicle routing with split deliveries. *Discret. Appl. Math.* **1994**, *50*, 239–254. [CrossRef]
26. NOES Server Solvers. Available online: https://neos-server.org/neos/solvers/index.html (accessed on 27 October 2024).
27. Linderoth, J.; Ralphs, T. Noncommercial software for mixed-integer linear programming. In *Integer Programming*; CRC Press: London, UK, 2005; pp. 269–320. [CrossRef]

28. Meindl, B.; Templ, M. *Analysis of Commercial and Free and Open Source Solvers for Linear Optimization Problems*; Technical Report CS-2012-1; Vienna University of Technology: Vienna, Austria, 2012. Available online: http://hdl.handle.net/20.500.12708/37465 (accessed on 27 October 2024).
29. Atamtürk, A.; Savelsbergh, M.W. Integer-programming software systems. *Ann. Oper. Res.* **2005**, *140*, 67–124. [CrossRef]
30. Du, Y.; Kochenberger, G.; Glover, F.; Wang, H.; Lewis, M.; Xie, W.; Tsuyuguchi, T. Solving clique partitioning problems: A comparison of models and commercial solvers. *Int. J. Inf. Technol. Decis. Mak.* **2022**, *21*, 59–81. [CrossRef]
31. González, M.; López, J.; Aparicio, J. A parallel algorithm for matheuristics: A comparison of optimization solvers. *Electronics* **2020**, *9*, 1541. [CrossRef]
32. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. 2024. Available online: https://www.gurobi.com (accessed on 27 October 2024).
33. International Business Machines Corporation. IBM ILOG CPLEX Optimization Studio Reference Manual. 2024. Available online: https://www.ibm.com/docs/en/icos/22.1.1 (accessed on 27 October 2024).
34. Hart, W.; Watson, J.; Woodruff, D. Pyomo: Modeling and solving mathematical programs in Python. *Math. Program. Comput.* **2011**, *3*, 219–260. [CrossRef]
35. Glover, F. Heuristics for integer programming using surrogate constraints. *Decis. Sci.* **1977**, *8*, 156–166. [CrossRef]
36. Martí, R.; Laguna, M.; Glover, F. Principles of scatter search. *Eur. J. Oper. Res.* **2006**, *169*, 359–372. [CrossRef]
37. Laguna, M.; Martí, R. Scatter Search. In *Metaheuristic Procedures for Training Neural Networks*; Operations Research/Computer Science Interfaces Series; Alba, E., Martí, R., Eds.; Springer: Boston, MA, USA, 2006; Volume 36, pp. 139–152. [CrossRef]
38. Carlos, M.; Gallardo, A.; Edo-Alcon, N.; Abaso, J.R. Influence of the municipal solid waste collection system on the time spent at a collection point: A case study. *Sustainability* **2019**, *11*, 6481. [CrossRef]
39. Owusu, F.; Oduro, S.; Essandoh, H.; Wayo, F.; Shamudeen, M. Characteristics and management of landfill solid waste in Kumasi, Ghana. *Sci. Afr.* **2019**, *3*, e00052. 0 052 [CrossRef]
40. D'Onza, G.; Greco, G.; Allegrini, M. Full cost accounting in the analysis of separated waste collection efficiency: A methodological proposal. *J. Environ. Manag.* **2016**, *167*, 59–65. [CrossRef]
41. Pichugina, O.; Matsiy, O.; Skob, Y. Performance Comparison of Unbounded Knapsack Problem Formulations. *CEUR-WS* **2023**, *3403*, 263–272.
42. Saghand, P.G.; Charkhgard, H. Exact solution approaches for integer linear generalized maximum multiplicative programs through the lens of multi-objective optimization. *Comput. Oper. Res.* **2022**, *137*, 105549. [CrossRef]
43. Schulze, P.; Walter, R. A note on the exact solution of the minimum squared load assignment problem. *Comput. Oper. Res.* **2023**, *159*, 106309. [CrossRef]
44. Goldberg, D.; Lingle, R. Alleles, Loci and the traveling salesman problem. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*; Psychology Press: New York, NY, USA, 2014; pp. 154–159. [CrossRef]
45. Davis, L. Applying adaptive algorithms to epistatic domains. In Proceedings of the International Joint Conferences on Artificial Intelligence, San Francisco, CA, USA, 18–23 August 1985; pp. 162–164. [CrossRef]
46. Oliver, I.; Smith, D.; Holland, J. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*; Psychology Press: New York, NY, USA, 1987; pp. 224–230. [CrossRef]
47. Hussain, A.; Muhammad, Y.; Nauman, M.; Hussain, I.; Mohamd, A.; Gani, S. Genetic algorithm for traveling salesman problem with modified cycle crossover operator. *Comput. Intell. Neurosci.* **2017**, *1*, 7430125. [CrossRef]
48. Hollander, M.; Wolfe, D. *Nonparametric Statistical Methods*; John Wiley & Sons: Hoboken, NJ, USA, 1973.
49. Siegel, S. *Non Parametric Statistics for the Behavioural Sciences*; MacGraw Hill Int: Columbus, OH, USA, 1988.
50. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2022. Available online: https://www.R-project.org/ (accessed on 27 October 2024).