



WILEY

Intl. Trans. in Op. Res. 30 (2023) 3801–3832
DOI: 10.1111/itor.13294INTERNATIONAL
TRANSACTIONS
IN OPERATIONAL
RESEARCH

Effective metaheuristics for the latency location routing problem

Alan Osorio-Mora^a , Carlos Rey^{a,b}, Paolo Toth^{a,*}  and Daniele Vigo^a^aDEI, University of Bologna, Viale Risorgimento 2, Bologna I-40136, Italy^bDepartment of Industrial Engineering, Universidad del Bío-Bío, Concepción 4030000, ChileE-mail: alan.osorio2@unibo.it [Osorio-Mora]; crey@ubiobio.cl [Rey]; paolo.toth@unibo.it [Toth];
daniele.vigo@unibo.it [Vigo]

Received 16 July 2022; received in revised form 13 March 2023; accepted 15 March 2023

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 paper proposes three novel metaheuristic algorithms to solve the LLRP. They use a simulated annealing framework, which after each temperature reduction is intensified through a variable neighborhood descent procedure. Each algorithm uses a different search strategy as intensification. Results on 76 benchmark instances indicate that the proposed metaheuristics outperform the state-of-the-art algorithms, finding new best solutions for all the large-sized instances (over 100 customers), or the currently known optimal ones for most of the small- and medium-sized 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

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-known 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

*Corresponding author.

© 2023 The Authors.

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

traditional approach has been to minimize the total cost, which includes the transportation cost, the fixed costs to 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 Moshref-Javadi and Lee (2016) began studying the latency LRP (LLRP), which is a combination of the FLP and the cumulative capacitated VRP (CCVRP) (see Ngueveu et al., 2010). 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 postdisaster planning operations and humanitarian logistics problems (Nguveu et al., 2010; Moshref-Javadi and Lee, 2016). Since the LLRPs are closely related to humanitarian logistic, a bad planning may lead to losses of human lives. As pointed out in Nucamendi-Guillén et al. (2022), 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 LRPs minimizing the global travel time are not appropriate for latency problems (Moshref-Javadi and Lee, 2016; Sze et al., 2017). 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. Also 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 suboptimal solutions (Salhi and Rand, 1989). This consideration strongly supports the importance of solving efficiently LLRPs.

In this paper, 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 Kirkpatrick et al., 1983) and variable neighborhood descent (VND) procedures (Duarte et al., 2018). 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, a formal description of the problem is provided and the related literature is analyzed. Section 3 describes the proposed metaheuristics. In Section 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 prove optimality by the previously proposed exact methods. Finally, in Section 5, the main conclusions are drawn and future directions are proposed.

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 nonnegative demand q_i . Each edge $(i, j) \in E$, with $i \neq j$, has an associated nonnegative 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 (Ngueveu et al., 2010). A mixed integer linear programming (MILP) model of the LLRP has been proposed in Moshref-Javadi and Lee (2016). Although the MILP model considers a vehicle capacity potentially different for each vehicle, the algorithm proposed in this paper and the instances studied were designed for homogeneous fleets of vehicles.

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 (e.g., 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 Moshref-Javadi and Lee (2016), 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, Nucamendi-Guillén et al. (2022) proposed two MILP models, three enumerative algorithms, and a GRASP-based iterated local search (GBILS) algorithm 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 algorithms RGA and MA in small computing times. The computational results reported in Nucamendi-Guillén et al. (2022) 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 Dukkanci et al. (2019), the authors studied the green LRP (GLRP), addressing it as a cumulative LRP. Despite the similarities in the names of the problems, the GLRP corresponds to the family of the cumulative VRPs (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 CuVRP idea is included in the fuel consumption expression but it is not the only component, since the speed of the vehicle and the technical components (engine) are also 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 Corona-Gutiérrez et al. (2022), where a review of the different cumulative routing

problems is presented. 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 LRPs the reader is referred to the following works in chronological order: Nagy and Salhi (2007), Lopes et al. (2013), Prodhon and Prins (2014), Cuda et al. (2015), Drexl and Schneider (2015), Schneider and Drexl (2017), Albareda-Sambola and Rodríguez-Pereira (2019), and Mara et al. (2021).

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 that 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 that 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 with respect to 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 3.2). Then an SA 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/intraroute 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 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: (a) 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 (b) 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. Note that the algorithms

Algorithm 1. Main scheme

```

1: Input:  $t_0, t_f, GP, NC_{\max}, it_{SA}, \alpha, pen, s_0$ 
2: Output:  $s_{bf}$  (Final Solution)
3:  $temp = t_0, NC = 0, s_c = s_0, s_{bf} = s_0$ 
4: while ( $temp > t_f$  and  $NC < NC_{\max}$ ) do
5:    $it = 0, s_{temp} = s_c$ 
6:   while ( $it < it_{SA}$ ) do
7:      $s_p = \text{RandomMove}(GP, s_c)$ 
8:     if ( $\text{AcceptanceCriteria}(temp, s_p, s_c)$ ) then
9:        $s_c = s_p$ 
10:    if ( $f(s_c) < f(s_{bf})$  and  $\text{IsFeasible}(s_c)$ ) then
11:       $s_{bf} = s_c$ 
12:    end if
13:  end if
14:   $it = it + 1$ 
15: end while
16: if ( $f(s_{temp}) = f(s_c)$ ) then
17:    $NC = NC + 1$ 
18:    $temp = \alpha * temp$ 
19: else
20:    $NC = 0$ 
21:    $s_c = \text{VNDX}(s_c)$ 
22:   if ( $f(s_c) < f(s_{bf})$  and  $\text{IsFeasible}(s_c)$ ) then
23:      $s_{bf} = s_c$ 
24:   end if
25:    $temp = \alpha * temp$ 
26: end if
27: end while
28:  $s_{LKH}$  = 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  $s_{bf}$ .
29: if ( $f(s_{LKH}) < f(s_{bf})$ ) then
30:    $s_{bf} = s_{LKH}$ 
31: end if
32: Return:  $s_{bf}$ 

```

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 3.1 for details. If none of the it_{SA} moves generated were accepted, that is, the current solution s_c remains same as the solution s_{temp} at the beginning of the current temperature, the counter of nonchanges NC is augmented by 1, otherwise it is set equal to 0. After the random phase, a VND procedure ($\text{VNDX}(s_c)$) 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 3.4. Then, the value of $temp$ is reduced according to a cooling factor α . 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 (LKH-3) heuristic proposed in Helsgaun (2017) 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.

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 (Helsgaun, 2000). The LKH solvers are based on the Lin–Kernighan algorithm (Lin and Kernighan, 1973), 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 Helsgaun, 2017).

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:

$$f(s_c) = \bar{f}(s_c) + pen\Delta Q, \quad (1)$$

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. Note that for feasible solutions, the second term of the formula is equal to 0. 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.

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 (MacQueen et al., 1967) 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 Fig. 1a, where the triangles are the customers and the squares are the depots. The k -means algorithm is used to define N_v clusters of customers and the corresponding centroids (Fig. 1b; the clusters are represented by big circles, and their centroids are represented by 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 (Fig. 1c). Then, the p depots with the highest scores are opened (depots D2 and D3 in Fig. 1d), and the clusters that were assigned to depots not selected are allocated to the closest one among the open depots (Fig. 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 (Fig. 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

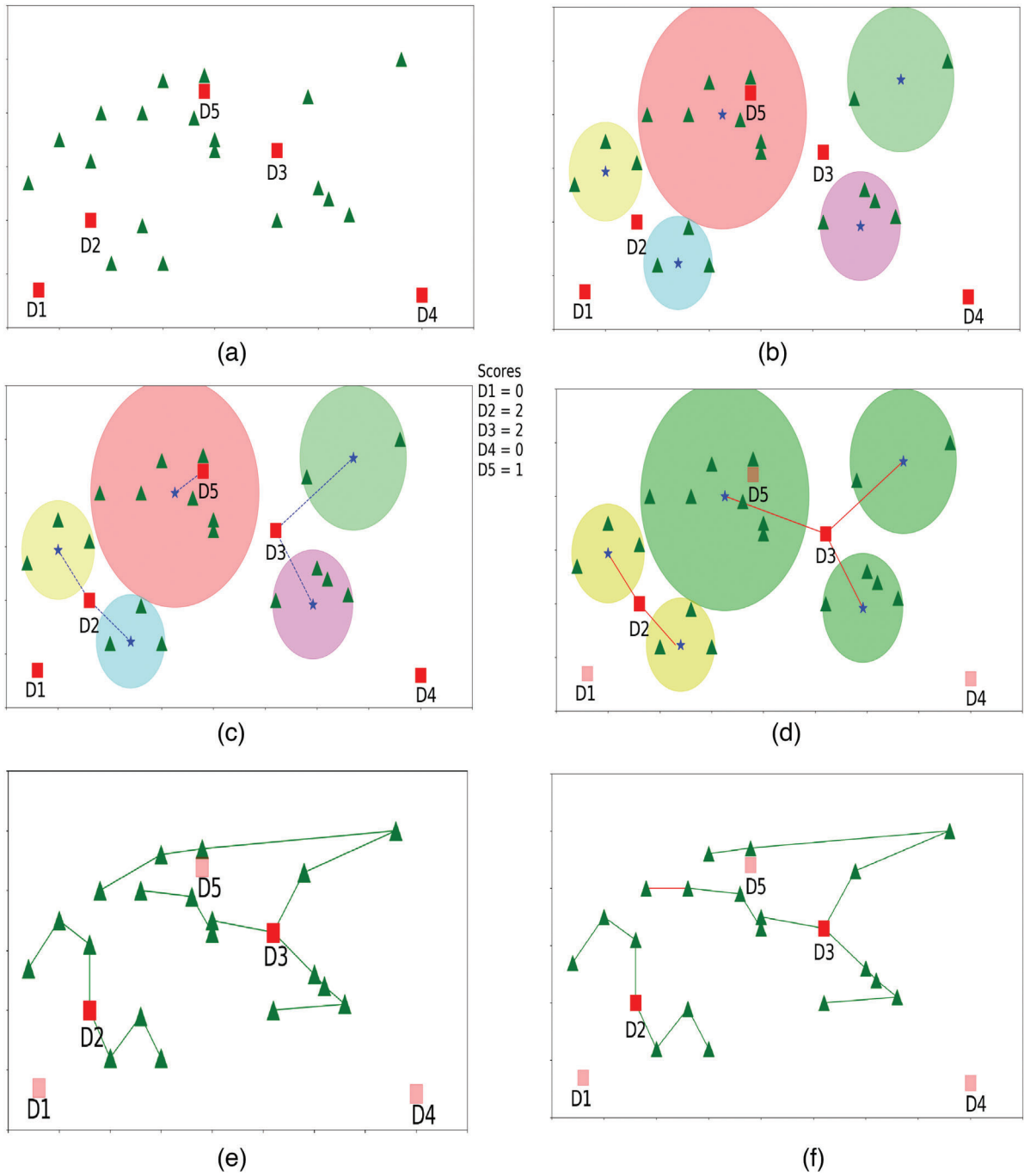


Fig. 1. Initial solution procedure. (a) An LLRP instance, (b) K-means procedure, (c) scoring, (d) allocation process, (e) LKH-3 procedure, and (f) repair procedure.

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 $(NV_j Q - dem_j)$, that is, the depot that 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 that has the largest value of $(NV_j Q - dem_j)$, that is, 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 that 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 interroute local search procedure on the *insertion*, *swap*, and 2-opt neighborhoods (further details are given in Section 3.3). The move that minimizes the sum of the penalizations associated with the infeasibilities, plus the global latency, is applied (Fig. 1f). Other approaches combining clustering and LKH procedures have been successfully used to solve the CLRP in Escobar et al. (2013). Nevertheless, the proposed clustering and the location/allocation procedures are totally different from the one presented in this paper. The mentioned work proposed an iterative procedure that 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 an MILP model.

3.3. Local search

The whole framework of the algorithms uses eight neighborhoods, and the random moves are selected from six 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. Note that the neighborhoods in Group (i) are used also in the VND procedure. 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 intraroute and the interroute cases. For the interroute case, they can be applied for routes starting from the same depot or from different depots. The neighborhoods are as follows.

- *insertion*: This operator selects customer i and 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 Nguveu et al. (2010), where the

Algorithm 2. IntraLS procedure

```

1: Input:  $r_c$ ,  $NeighIntra = \{insertion, swap, 2-opt\}$ 
2: Output:  $r_p$  (IntraLS route)
3:  $r_p = r_c$ 
4: for (each  $ng \in NeighIntra$ ) do
5:    $flag = true$ 
6:   while( $flag = true$ ) do
7:      $r'_p = route(ng, r_p)$ 
8:     if ( $f(r'_p) < f(r_p)$ ) then
9:        $r_p = r'_p$ 
10:       $flag = true$ 
11:     else
12:        $flag = false$ 
13:     end if
14:   end while
15: end for
16: Return:  $r_p$ 

```

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 Nguveu et al. (2010).

The neighborhoods in Group (ii) correspond to operators that change the depot/vehicle relationship. These operators are applied only in the random phase, and are described as follows.

- *DepotOpenClose*: This operator selects two depots: i (open) and j (closed). All the routes assigned to i are reassigned to j . Since the current routes may not be good for the new depot, an intraroute 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.

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. Note 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 LRPs, as done in Vincent et al. (2010) for the CLRPP. The main difference between the operators described in the mentioned paper and those presented

Algorithm 3. Random move selection (*RandomMove*)

```

1: Input:  $GP, s_c$ 
2: Output:  $s_p$ 
3:  $m = \text{Rand}[1, 3GP + 3]$ 
4: if  $(m \leq GP)$  then
5:    $s_p = \text{RandomInsertion}(s_c)$ 
6: end if
7: if  $(GP < m \leq 2GP)$  then
8:    $s_p = \text{RandomSwap}(s_c)$ 
9: end if
10: if  $(2GP < m \leq 3GP)$  then
11:    $s_p = \text{RandomTwoOpt}(s_c)$ 
12: end if
13: if  $(m = 3GP + 1)$  then
14:    $(s_p) = \text{DepotOpenClose}(s_c)$ 
15: end if
16: if  $(m = 3GP + 2)$  then
17:    $(s_p) = \text{RouteSwap}(s_c)$ 
18: end if
19: if  $(m = 3GP + 3)$  then
20:    $(s_p) = \text{RouteRelocation}(s_c)$ 
21: end if
22: Return:  $s_p$ 

```

here is the local search procedure applied at the end of the move. Furthermore, Vincent et al. (2010) 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).

Note that the presented SA scheme is also known in the literature as multineighborhood SA (Bellio et al., 2021; Rosati et al., 2022).

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 VRPs, and have been used successfully for the solution of the CCVRPs. Information about how to compute them in constant time can be found in Kyriakakis et al. (2021):

- *arc-swap*: Two pairs of consecutive customers (i, j) and (k, l) exchange their position. This operator can be applied both for the intraroute and the interroute cases.
- *shift₂₋₁*: 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.

3.4. Variable neighborhood descent strategies

Denote by $Neigh = \{insertion, swap, 2\text{-opt}, arc - swap, shift_{2-1}\}$ 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 .

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 @ 3.70 GHz 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 algorithms. 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 Moshref-Javadi and Lee (2016), 30 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 Moshref-Javadi and Lee (2016) to test algorithms MA and RGA correspond to the 36 instances of the Tuzun and Burke data set (Tuzun and Burke, 1999), the 30 instances of the Prodhon data set proposed by Prins et al. (2004), and 10 of the 19 instances of the Barreto data set (Barreto, 2004). Only the results corresponding to the 30 instances of the Prodhon data set and to 8 of the 10 instances of the Barreto data set are reported in Nucamendi-Guillén et al. (2022).

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 López-Ibáñez et al., 2016) 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.

Table 1
Best configuration of parameters for each algorithm

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

¹As percentage of the value of the initial solution s_0 .

The IRACE software was trained with a set of instances, which correspond to one-third of the instances of each data set, considering different geographical distribution types, sizes, and other features. Globally, 26 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 that 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 algorithms SA-VND0, SA-VND1, and SA-VND2 have different parameter configurations. The values evaluated for each parameter were the following: $t_0 = \{500, 800, 1000, 1200, 1500, 2000\}$, $t_f = \{1, 5, 10, 50, 100, 150\}$, $\alpha = [0.90, 0.99]$, $it_{SA} = \{30, 50, 100, 200, 300, 500\}$, $NC_{max} = \{1, 2, 3, 4, 5, 6\}$, $GP = \{1, 2, 3, 4, 5, 6\}$, and $pen = \{1, 3, 7, 10, 15, 20\}$. The selected configurations are presented in Table 1. The calibration process required 67,020.9, 118,441.5, and 122,871.7 seconds for algorithms SA-VND0, SA-VND1, and SA-VND2, respectively.

4.2. Global 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 Supporting Information, the reader can find the detailed results of each run.

Tables 2–4 show the results obtained by algorithms RGA, MA, SA-VND0, SA-VND1, and SA-VND2 by executing 30 runs for each instance of the three data sets. Tables 3 and 4 also show the results obtained on the second and third data sets by algorithm GBILS executed five 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 Nucamendi-Guillén et al. (2022). The experiments in Moshref-Javadi and Lee (2016) were performed on a 3.1 GHz computer with 4 GB 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 algorithms RGA and MA are those reported in Moshref-Javadi and Lee (2016); 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 Moshref-Javadi and Lee (2016). On the other hand, the computing times of the exact methods

Table 2
Detailed results for the first data set (Tuzun–Burke Instances)

Table with columns: Instance, Nc, Nd, Lb, BKS0, BKS, GapLB, Best, GapLB, Avg, time, MA, GapB, Best, GapB, Avg, time, SA-VNDO, GapB, Best, GapB, Avg, time, SA-VNDI, GapB, Best, GapB, Avg, time, SA-VND2, GapB, Best, GapB, Avg, time. The table contains performance metrics for 33 instances across five different algorithms.

Table 3
Detailed results for the second data set (Prodhon instances)

Instance	Exact methods												SA-VND0				SA-VND1				SA-VND2									
	RGA			MA			GBLLS			Exact methods			SA-VND0		SA-VND1		SA-VND2		SA-VND0		SA-VND1		SA-VND2							
	LB	BKS ₀	BKS	GapLB	Avg	time	Best	GapB	Avg	time	Best	GapB	Avg	time	Best	GapB	Avg	time	Best	GapB	Avg	time	Best	GapB	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	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	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	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	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	1345.6	3.9	1477.3	678.0	1330.2	2.8	1379.8	522.0	1293.9	0.0	68.5	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	12,444.0	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	2055.6	1.6	64.8	2030.9	1.3	21,960.0	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	23,966.0	2311.8	0.0	2336.6	2231.6	2312.5	0.0	2337.3	2871.7	2313.7	0.1	2342.8	3126.5
100-5-2	1032.5	1184.7	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	21,960.0	1132.4	0.2	1136.0	2760.8	1129.8	0.0	1135.5	3976.1	1131.3	0.1	1135.7	3882.7
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	21,960.0	1507.9	0.0	1577.1	2530.4	1510.6	0.2	1579.0	3141.3	1510.6	0.2	1579.6	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	21,960.0	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	21,960.0	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	1338	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	21,960.0	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	6645.3	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	21,960.0	1901.3	0.0	1954.0	2130.3	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	9.1	1326.1	2214.0	1175.0	2.9	1236.3	1326.0	1287.5	12.7	75.8	1175.0	2.9	21,960.0	1143.3	0.1	1152.8	2899.7	1142.3	0.0	1155.5	4132.2	1145.9	0.3	1153.3	3739.8
100-10-2b	437.9	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	21,960.0	1566.5	0.2	1583.7	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	21,960.0	1209.2	0.0	1221.5	1415.2	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	1808.0	11.9	2031.9	1848.0	1799.0	8.2	1809.2	1287.0	1745.1	5.0	70.1	1799.0	8.2	21,960.0	1662.4	0.0	1705.6	1466.0	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	2952.3	5.6	3125.1	5757.0	2920.7	4.4	3092.4	3414.0	2893.9	2.3	278.2	2920.7	4.4	21,960.0	2798.6	0.0	2854.0	2130.3	2855.7	0.9	2863.3	26,673.1	2803.6	0.2	2860.0	33,819.4
200-10-1b	2688.4	3322.2	3327.1	32.8	3262.0	9.0	3992.6	4896.0	3532.2	6.2	3809.3	3240.0	3535.8	6.9	304.1	3532.2	6.2	21,960.0	3368.7	1.3	3477.1	11341.6	3477.9	0.9	3477.1	61,783.6	3477.9	0.0	3472.8	61,734.9
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	346.5	2049.2	3.2	21,960.0	1985.0	0.0	2002.0	17,483.5	1986.6	0.1	2004.2	5906.3	1988.3	0.2	2002.5	49,595.6
200-10-2b	236.9	2473.2	2366.1	14.7	2598.9	11.2	2729.9	5382.0	2516.4	7.7	2684.5	3414.0	2473.2	5.9	273.7	2516.4	7.7	21,960.0	2366.1	0.0	2379.0	12,347.0	2355.2	0.8	2380.2	19,120.6	2368.9	0.1	2380.1	18,699.6
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	21,960.0	2741.2	0.0	2758.1	14,786.4	2744.7	0.0	2757.4	27,900.5	2751.2	0.4	2763.8	21,623.8
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	21,960.0	3242.2	0.5	3274.6	9511.1	3233.9	0.3	3267.8	16,534.2	3225.8	0.0	3273.7	14,933.5
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	14,596.4	1503.0	0.1	1521.2	3975.5	1503.9	0.2	1522.3	6248.6	1503.8	0.2	1521.8	5692.4

and algorithm GBILS are multiplied by a “scaling factor” equal to 0.61, since the computer used in this paper is faster than that used in Nucamendi-Guillén et al. (2022). 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.cpbenchmark.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* (which are the lower bounds proposed in Moshref-Javadi and Lee, 2016), the optimal solution value of the linear relaxation found after implementing the first MILP model (Model 1) presented in Nucamendi-Guillén et al. (2022) for the LLRP, and solving it using CPLEX 20.1, and the optimal solution value/best lower bound obtained after implementing and solving (with a time limit of 216,000 seconds) the first MILP model (Model 1) presented in Bruni et al. (2022) for the multidepot 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. Note that new tighter lower bounds are reported in italic in the Tables 2–4. Furthermore, note that the value (331.9) of *LB2* reported in Moshref-Javadi and Lee (2016) for the instance 20-5-1 of the Prodhon data set is larger than the proved optimal solution value 330.0 (see Nucamendi-Guillén et al., 2022). Hence, we implemented a correct procedure and computed again the value of *LB2* for all the instances.
- BKS_0 : Best known solution value considering algorithms RGA, MA, the five exact methods, and 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 Nucamendi-Guillén et al. (2022) is presented in boldface.
- BKS : Best known solution value considering all the algorithms.
- Gap_{LB} : Percentage gap between BKS and LB , computed as $Gap_{LB} = 100 \frac{(BKS-LB)}{LB}$.

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 Nucamendi-Guillén et al. (2022) 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 two 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 two 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 $0.61 \times (2 \times 2000 + 3 \times 7200) = 21,960.00$ 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.

4.2.1. The Tuzun–Burke data set

The results for these instances are given in Table 2. No results for the exact algorithms and algorithm GBILS are presented on this data set in Nucamendi-Guillén et al. (2022). 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%. Note that for each 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 among the three proposed algorithms is SA-VND0. For all but six 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 Moshref-Javadi and Lee (2016). 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. Note that these values are very close to the optimal solution values (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.

4.2.2. The Prodhon data set

The results of this data set are presented in Table 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 . Note that for these nine instances, the solutions found by the exact methods were proved to be optimal by Nucamendi-Guillén et al. (2022). The best solution values found by the three proposed metaheuristics are equal to those found by 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 MD k -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, 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.

4.2.3. 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 4. It is to point out that for instance Christ-50-5 the best solution values found by the proposed algorithms ($Best = 1661.6$), and also by algorithm MA ($Best = 1690.8$), are smaller than the optimal solution value ($Best = 1719.9$) reported in Nucamendi-Guillén et al. (2022). After implementing and executing (using CPLEX 20.1) the first MILP model (Model 1) presented in Nucamendi-Guillén et al. (2022) we proved that the solution found by the proposed metaheuristics is optimal (the details are given in Appendix A), 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 Nucamendi-Guillén et al. (2022). Because of the above, these two instances were not considered in the global average results of the exact methods and of algorithm GBILS. For the instances Min-134-8 and Or-117-14 no results are reported in Nucamendi-Guillén et al. (2022). The last line of Table 4 gives the average values (Global avg_{NG}) computed by considering only the six instances whose values are correctly reported in Nucamendi-Guillén et al. (2022). The table shows that each of the proposed algorithms is able to improve the value BKS_0 for 5 of the 10 instances (including the optimal solution found for the instance Christ-50-5); for the remaining 5 instances, the 3 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 four 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 Moshref-Javadi and Lee (2016). For nine 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 nine instances but 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 Nucamendi-Guillén et al. (2022). However, note that for the six 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.

4.2.4. 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 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 algorithm GBILS, since the average values found by this algorithm are not reported in Nucamendi-Guillén et al. (2022). Moreover, it can be noted that algorithm SA-VND0 is able to find or improve the best solution value found by GBILS for all the 36 instances analyzed, while 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 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 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. Although 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 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.

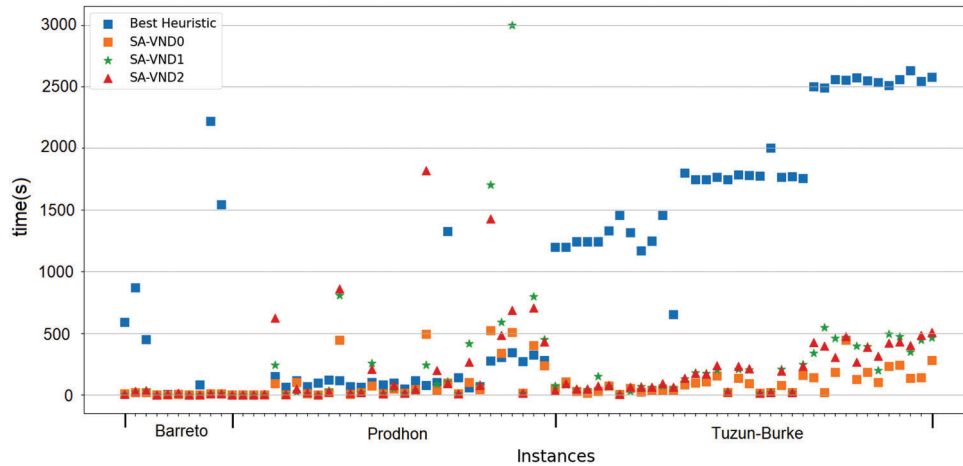


Fig. 2. Computing times (seconds) required by each metaheuristic to reach the target values for the considered instances.

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 algorithms RGA, MA, and GBILS. These experiments considered a time limit for each instance equal to the minimum between 3000 seconds 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 cannot reach the target value within the time limit. The computing times for each instance and each algorithm are drawn in Fig. 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), that is, 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 5. 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 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

Table 5
Average computing times required by each metaheuristic to reach the target values for each data set

Data set	<i>Best</i> heuristic (BH)	SA-VND0		SA-VND1		SA-VND2	
	<i>time</i>	<i>time</i>	# Faster than BH	<i>time</i>	# Faster than BH	<i>time</i>	# 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

Table 6
Average solution values and gaps provided by each part of the algorithms

Data set	Initial		SA-VND0				SA-VND1				SA-VND2			
			<i>Avg</i>	<i>Gap</i>	<i>Avg</i>	<i>Gap</i>	<i>Avg</i>	<i>Gap</i>	<i>Avg</i>	<i>Gap</i>	<i>Avg</i>	<i>Gap</i>	<i>Avg</i>	<i>Gap</i>
	<i>Avg</i>	<i>Gap</i>	<i>SA</i>	<i>SA</i>	<i>LKH</i>	<i>LKH</i>	<i>SA</i>	<i>SA</i>	<i>LKH</i>	<i>LKH</i>	<i>SA</i>	<i>SA</i>	<i>LKH</i>	<i>LKH</i>
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	12,133.5	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

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 algorithm SA-VND0 is slightly faster than the *Best* heuristic, while 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.

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 B.

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 SA frame, and s_{LKH} as the solution found by applying the LKH-3 procedure to s_{SA} . Table 6 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. Note that the initial solution is shared by the three proposed metaheuristics. The largest improvement is achieved by the SA 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

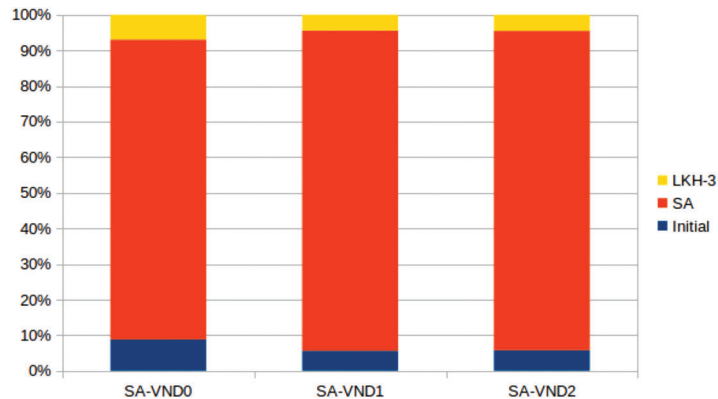


Fig. 3. Relative computing time required by each part of the proposed metaheuristics.

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, 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 algorithms SA-VND0, SA-VND1, and SA-VND2, respectively.

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

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.

As it was previously shown, the combined part of the algorithm (SA-VND) is the one that 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 were carried out considering each procedure independently. Table 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. 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 restart 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

Table 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

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 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. 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 Fig. 4a), 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 algorithms SA-VND0 and SA-VND2, and equal to 2 for 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 Fig. 4b), 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, algorithms SA-VND0 and SA-VND2 allow to explore in a deeper way the other neighborhoods, which can be a possible

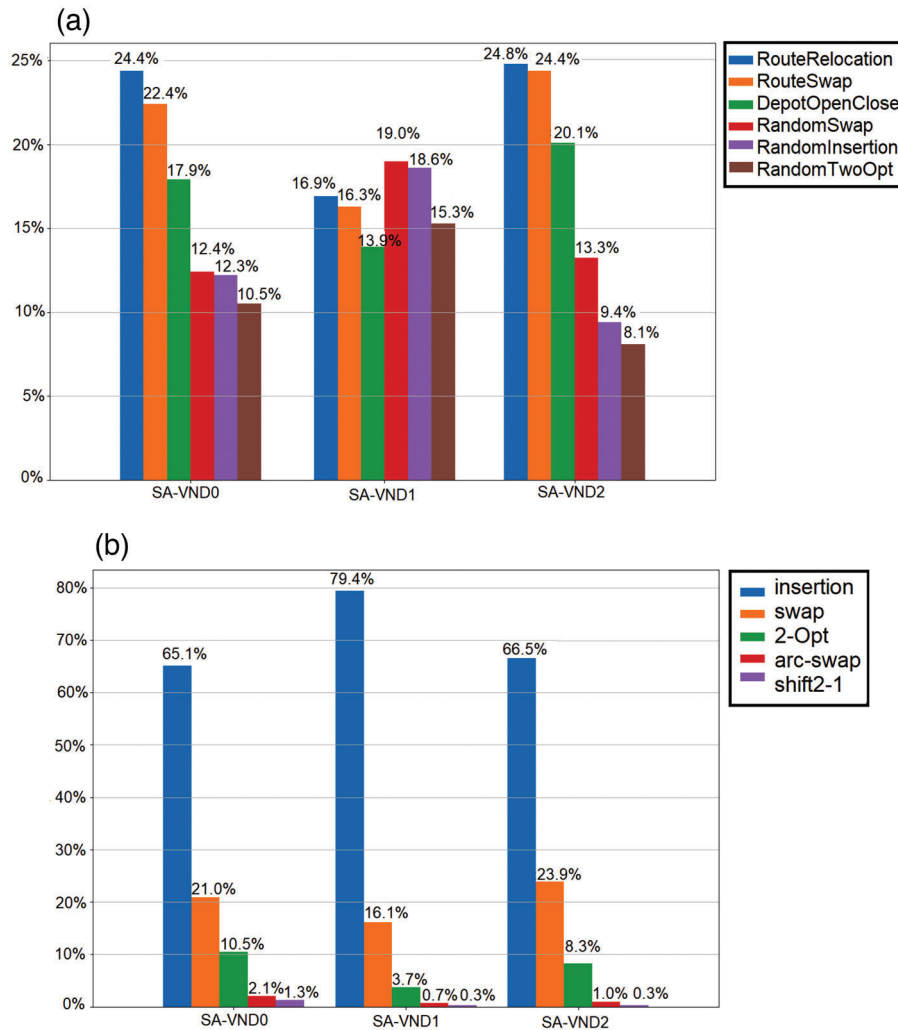


Fig. 4. Percentage average contribution of the neighborhoods: (a) SA and (b) VND.

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 8 presents the global average results of this experiment by considering five 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):

© 2023 The Authors.

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 8
Global average results by removing each neighborhood

Removed neighborhood	<i>Best</i>	<i>Avg</i>	<i>time</i> (avg)	Gap_B	Gap_{BKS_0}	<i>leq BKS</i>	<i>leq BKS₀</i>
None (original algorithms)	3667.8	3702.0	198.4	0.57	−2.58	19.7	72.3
<i>insertion</i>	3709.9	3758.9	157.5	1.36	−1.82	12.7	62.7
<i>swap</i>	3674.9	3710.5	178.8	0.75	−2.40	16.7	70.7
<i>2-opt</i>	3701.8	3742.1	196.0	0.94	−2.23	15.0	70.0
<i>arc-swap</i>	3673.0	3709.4	197.1	0.64	−2.52	18.0	71.3
<i>shift₂₋₁</i>	3672.5	3707.9	199.3	0.62	−2.53	17.7	69.7
<i>RandomInsertion</i>	3667.0	3702.7	205.9	0.61	−2.54	17.3	71.3
<i>RandomSwap</i>	3670.2	3709.5	193.8	0.66	−2.50	16.7	70.7
<i>RandomTwoOpt</i>	3678.0	3709.2	195.8	0.60	−2.56	17.0	71.7
<i>DepotOpenClose</i>	3936.2	3956.2	193.6	9.28	5.79	12.3	37.0
<i>RouteSwap</i>	3671.2	3705.3	204.9	0.65	−2.51	15.3	71.7
<i>RouteRelocation</i>	3721.0	3750.0	215.9	2.15	−1.06	12.7	56.7

- *Best*: Best solution value found.
- *Avg*: Average solution value (computed over five 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 2–4), computed as $Gap_B = 100 \frac{(Best-BKS)}{BKS}$.
- Gap_{BKS_0} : Percentage gap between *Best* and the best known solution value found by the current state-of-the-art algorithms (reported in columns *BKS₀* in Tables 2–4), computed as $Gap_{BKS_0} = 100 \frac{(Best-BKS_0)}{BKS_0}$.
- *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 BKS₀*: The number of instances for which the best solution value *Best* found by the proposed algorithm is better than or equal to *BKS₀*.

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. 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, that is, 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 five of the seven criteria. According to the results presented, there are no reasons for excluding some of the proposed neighborhoods from the SA-VND framework presented in this paper.

As it was mentioned before, the presented neighborhoods have been used in several routing/LRPs 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 *shift₂₋₁*, were also included in the RGA algorithm presented in Moshref-Javadi and Lee (2016), and all those neighborhoods but the *arc-swap*, the *shift₂₋₁* and the interroute 2-opt were included in the GBILS algorithm (Nucamendi-Guillén et al., 2022). Furthermore, the solution space of the MAs (Moshref-Javadi and Lee, 2016) is similar to the solution space of the three proposed metaheuristics. 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 paper, that is, the initial solution, the exploration (combining multineighborhood 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.

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 versus SA-VND1, SA-VND0 versus SA-VND2, and SA-VND1 versus SA-VND2. The *t*-test was chosen in order to perform the same statistical analysis proposed in Moshref-Javadi and Lee (2016). The results indicate that for 60 instances, with a 5% of significance level, there are no 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 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.

Table 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

5. Conclusions and future research

An effective metaheuristic framework for the LLRP was proposed. It combines the well-known SA and VND 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 Moshref-Javadi and Lee, 2016), and GBILS (proposed in Nucamendi-Guillén et al., 2022), and the five exact methods (proposed in Nucamendi-Guillén et al., 2022) 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 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-sized instances the proposed metaheuristics find several proved optimal solutions, and in some cases the average value was equal to the optimal one.

Although the metaheuristics presented in this paper are more time-consuming than 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 multidepot CCVRP, the LLRP with time windows, and other extensions of the CCVRP and the FLP.

Acknowledgments

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

Program/Doctorado Becas Chile/2018-72190600. We also acknowledge Air Force Office of Scientific Research Grant FA8655-20-1-7019, AFOSR Grant no. A9550-17-1-0234, and Grant no. FA8655-21-1-7046. The authors are extremely grateful to the anonymous referees for their very helpful comments.

Open Access Funding provided by Universita degli Studi di Bologna within the CRUI-CARE Agreement.

References

- Albareda-Sambola, M., Rodríguez-Pereira, J., 2019. Location-routing and location-arc routing. In *Location Science*. Springer, Berlin.
- Barreto, S., 2004. Análise e modelização de problemas de localização-distribuição [analysis and modelling of location-routing problems]. PhD thesis, University of Aveiro, Aveiro.
- Bellio, R., Ceschia, S., Di Gaspero, L., Schaerf, A., 2021. Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling. *Computers & Operations Research* 132, 105300.
- Bruni, M.E., Khodaparasti, S., Martínez-Salazar, I., Nucamendi-Guillén, S., 2022. The multi-depot k-traveling repairman problem. *Optimization Letters* 16, 2681–2709.
- Corona-Gutiérrez, K., Nucamendi-Guillén, S., Lalla-Ruiz, E., 2022. Vehicle routing with cumulative objectives: a state of the art and analysis. *Computers & Industrial Engineering* 169. <https://doi.org/10.1016/j.cie.2022.108054>.
- Cuda, R., Guastaroba, G., Speranza, M.G., 2015. A survey on two-echelon routing problems. *Computers & Operations Research* 55, 185–199.
- Drexl, M., Schneider, M., 2015. A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research* 241, 2, 283–308.
- Duarte, A., Sánchez-Oro, J., Mladenović, N., Todosijević, R., 2018. *Variable Neighborhood Descent*. Springer International, Cham. pp. 341–367.
- Dukkanci, O., Kara, B.Y., Bektaş, T., 2019. The green location-routing problem. *Computers & Operations Research* 105, 187–202.
- Escobar, J.W., Linfati, R., Toth, P., 2013. A two-phase hybrid heuristic algorithm for the capacitated location-routing problem. *Computers & Operations Research* 40, 1, 70–79.
- Helsgaun, K., 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research* 126, 1, 106–130.
- Helsgaun, K., 2017. An extension of the Lin–Kernighan–Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical Report, Roskilde University.
- Ke, L., 2018. A brain storm optimization approach for the cumulative capacitated vehicle routing problem. *Memetic Computing* 10, 4, 411–421.
- Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. *Science* 220, 4598, 671–680.
- Kyriakakis, N.A., Marinaki, M., Marinakis, Y., 2021. A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem. *Computers & Operations Research* 134, 105397.
- Lin, S., Kernighan, B.W., 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research* 21, 2, 498–516.
- Lopes, R.B., Ferreira, C., Santos, B.S., Barreto, S., 2013. A taxonomical analysis, current methods and objectives on location-routing problems. *International Transactions in Operational Research* 20, 6, 795–822.
- López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T., 2016. The IRACE package: iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3, 43–58.
- MacQueen, J., et al., 1967. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, Oakland, CA, pp. 281–297.
- Mara, S.T.W., Kuo, R., Asih, A.M.S., 2021. Location-routing problem: a classification of recent research. *International Transactions in Operational Research* 28, 2941–2983.
- Moshref-Javadi, M., Lee, S., 2016. The latency location-routing problem. *European Journal of Operational Research* 255, 2, 604–619.

- Nagy, G., Salhi, S., 2007. Location-routing: issues, models and methods. *European Journal of Operational Research* 177, 2, 649–672.
- Ngueveu, S.U., Prins, C., Wolfler Calvo, R., 2010. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research* 37, 11, 1877–1885.
- Nucamendi-Guillén, S., Martínez-Salazar, I., Khodaparasti, S., Bruni, M.E., 2022. New formulations and solution approaches for the latency location routing problem. *Computers & Operations Research* 143, 105767.
- Prins, C., Prodhon, C., Wolfler Calvo, R., 2004. Nouveaux algorithmes pour le problème de localisation et routage avec contraintes de capacité. MOSIM'04 (4ème Conf. Francophone de Modélisation et Simulation), Nantes, France.
- Prodhon, C., Prins, C., 2014. A survey of recent research on location-routing problems. *European Journal of Operational Research* 238, 1, 1–17.
- Rosati, R.M., Petris, M., Di Gaspero, L., Schaerf, A., 2022. Multi-neighborhood simulated annealing for the sports timetabling competition itc2021. *Journal of Scheduling* 25, 3, 301–319.
- Salhi, S., Rand, G.K., 1989. The effect of ignoring routes when locating depots. *European Journal of Operational Research* 39, 2, 150–156.
- Schneider, M., Drexl, M., 2017. A survey of the standard location-routing problem. *Annals of Operations Research* 259, 1, 389–414.
- Sze, J.F., Salhi, S., Wassan, N., 2017. 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* 101, 162–184.
- Tuzun, D., Burke, L.I., 1999. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research* 116, 1, 87–99.
- Vincent, F.Y., Lin, S.W., Lee, W., Ting, C.J., 2010. A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering* 58, 2, 288–299.

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 Section 4.2.4, the proposed metaheuristics are more time-consuming and much more stable than 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 algorithms RGA and MA by considering 30 runs for each instance, algorithm GBILS

© 2023 The Authors.

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table B.1
Summarized results on #BKS, #Avg-BKS₀, and #Best-BKS₀ for the complete data set considering 30, 10, and 5 runs

Data set	30 runs					10 runs					5 runs				
	RGa	MA	GBILS	Exact methods		SA-VND0	SA-VND1	SA-VND2	SA-VND0	SA-VND1	SA-VND2	SA-VND0	SA-VND1	SA-VND2	
Tuzun–Burke (36 instances)															
#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	59	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	

Table B.2
Average results on solution quality and computing times for the complete data set by considering 30, 10, and 5 runs

Data set	RGA			MA			GBILLS			Exact methods			SA-VND0			SA-VND1			SA-VND2																																										
	Best	Avg	time	GapB	Best	Avg	time	GapB	Best	Avg	time	GapB	Best	Avg	time	GapB	Best	Avg	time	GapB	Best	Avg	time	GapB	Best	Avg	time	GapB	Best	Avg	time	GapB	Best	Avg	time																										
Tuzun-Burke (36 instances)																																																													
30 runs	3826.3	4171.7	4544.5	1805.0	8.7	4028.4	4422.8	1861.2	5.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-																				
10 runs	3834.9	-	-	8.4	-	-	-	4.8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-												
5 runs	3839.1	-	-	8.3	-	-	-	4.7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Prodton (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	14,596.4	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	1503.8	1521.8	5692.4	0.2	-0.6	1503.8	1521.8	5692.4	0.2	-0.6	1503.8	1521.8	5692.4	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	1507.3	1522.1	1900.4	0.2	-0.6	1507.3	1522.1	1900.4	0.2	-0.6	1507.3	1522.1	1900.4	0.2	-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	1509.4	1521.1	949.7	0.3	-0.6	1509.4	1521.1	949.7	0.3	-0.6	1509.4	1521.1	949.7	0.3	-0.6																
Barreto (10 instances)																																																													
30 runs	9390.8	10,533.2	12,081.7	879.3	7.8	9841.9	10,712.6	9000.0	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	9433.8	9551.1	4164.4	0.1	-1.8	9456.5	9522.7	206.3	0.1	-1.8	9456.5	9522.7	206.3	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	4164.4	0.1	-1.8	9456.5	9522.7	206.3	0.1	-1.8	9456.5	9522.7	206.3	0.1	-1.8																					
5 runs	9406.8	-	-	7.6	-	-	-	3.2	-	-	-	-	-	-	-	9390.1	9592.1	140.7	0.2	-1.7	9416.1	9598.9	152.2	0.1	-1.9	9456.5	9522.7	206.3	0.1	-1.8	9456.5	9522.7	206.3	0.1	-1.8																										
Barrete-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	2370.0	2374.6	272.1	0.0	-0.7	2370.0	2374.6	272.1	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	2374.6	2374.6	272.1	0.0	-0.7	2374.6	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	2374.6	272.1	0.0	-0.7	2374.6	2374.6	272.1	0.0	-0.7																										
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	3660.5	3705.0	2216.9	0.3	-1.9	3660.5	3705.0	2216.9	0.3	-1.9	3660.5	3705.0	2216.9	0.3	-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	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	3660.5	3705.0	2216.9	0.3	-1.9	3660.5	3705.0	2216.9	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	12,838.2	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	1651.1	1664.2	1620.0	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	1620.0	0.2	-0.6	1651.1	1664.2	1620.0	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	1653.6	1663.7	813.9	0.3	-0.6																										

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 B.1 and B.2 by considering the data sets: Tuzun–Burke (36 instances), Prodhon (30 instances), Barreto (10 instances), Barreto-NG (containing the 6 instances reported in Nucamendi-Guillén et al., 2022), Total (containing the 76 instances of the overall data set), and Total-NG (containing the 36 instances reported in Nucamendi-Guillén et al., 2022). The rows in Table B.1 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 .

The columns in Table B.2 have the same meaning as in Tables 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. 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, 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 three times, which implies that 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 the number of runs to 5, algorithm SA-VND0 is still able to improve or find the value BKS_0 for 73 instances, while 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 six times, which leads to conclude that all the proposed algorithms require smaller computing times than algorithms RGA and MA to find better solutions. Despite the above, 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, algorithm SA-VND0 exhibits, among the three proposed metaheuristics, the most stable performance in terms of solution quality and the smallest computing times.