EURO Journal on Computational Optimization 11 (2023) 100063

Contents lists available at ScienceDirect

EURO Journal on Computational Optimization

journal homepage: www.elsevier.com/locate/ejco

A conceptually simple algorithm for the capacitated location-routing problem

Maximilian Löffler, Enrico Bartolini, Michael Schneider *

Deutsche Post Chair – Optimization of Distribution Networks, RWTH Aachen University, Germany

ARTICLE INFO

Keywords:
Location
Routing
Location routing
Simple heuristic
Depot configuration refinement

ABSTRACT

Location-routing problems (LRPs) jointly optimize the location of depots and the routing of vehicles. The most studied LRP variant, the capacitated LRP (CLRP), has been addressed by a large number of metaheuristic approaches. These methods often decompose the problem into a location stage to determine a promising depot configuration and a routing stage, in which a vehicle-routing problem is solved to assess the quality of the previously determined depot configuration. Unfortunately, the CLRP literature does not shed much light on the important question which algorithmic features have the biggest influence on the solution quality and runtime of such heuristics. The purpose of this paper is to propose a conceptually simple (yet reasonably effective) heuristic for the CLRP and to provide some insights on the design of successful metaheuristics for this problem. Our algorithm is a hybrid combining (i) a GRASP phase that uses a variable neighborhood descent for local improvement in the location stage, and (ii) a variable neighborhood search in the routing stage. We analyze the impact of the algorithmic components on solution quality and runtime. In addition, we find that the suboptimal routing solutions used to assess the quality of the investigated depot configurations in tendency lead to depot configurations with too many open depots. We propose a depot configuration refinement phase that alleviates this drawback, and we show

* Corresponding author.
E-mail addresses: loeffler@dpo.rwth-aachen.de (M. Löffler), bartolini@dpo.rwth-aachen.de (E. Bartolini), schneider@dpo.rwth-aachen.de (M. Schneider).

https://doi.org/10.1016/j.ejco.2023.100063
2192-4406/© 2023 The Author(s). Published by Elsevier Ltd on behalf of Association of European Operational Research Societies (EURO). This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

that this algorithmic component significantly contributes to the solution quality of our method, enabling it to provide reasonable results in comparison to the state-of-the-art methods from the literature.

© 2023 The Author(s). Published by Elsevier Ltd on behalf of Association of European Operational Research Societies (EURO). This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

In location-routing problems (LRPs), decisions on the location of depots are jointly taken with decisions on the routing of vehicles. Making these types of decisions independently of one another may lead to suboptimal results (see [28]). LRPs have been studied for decades, and the research community has been very active, particularly in the last years. This is witnessed by a large number of surveys on the recent LRP literature that were published in a very short time span (see [16,9,27,8,1,29,18]). For earlier LRP surveys, we refer the reader to Min et al. [19] and Nagy and Salhi [21].

The capacitated LRP (CLRP) is the most studied LRP variant. Given a set of potential capacitated depot locations and a set of customers with known demand, the objective of the CLRP is to determine the depots to open and the routes serving the customers from the opened depots in a way that minimizes total costs, which are composed of fixed costs of depots and vehicles and variable routing costs. The CLRP assumes an unlimited and homogeneous fleet of capacitated vehicles at each depot, and depot capacities limit the total demand served on the routes assigned to each depot. For a detailed mixed-integer programming formulation see Appendix A.

The CLRP is NP-hard because it contains the warehouse-location problem and the capacitated vehicle-routing problem (VRP). Due to its computational complexity and practical relevance, a substantial effort of the research community has been devoted to the development of high-quality heuristic approaches. Many of the proposed heuristics decompose the problem into a location-allocation stage, in which the facilities to be opened and the assignment of customers to facilities are determined, and a routing stage, in which a VRP is solved for each opened facility. Allocation decisions are often also allowed during the routing stage (leading to a multi-depot VRP, MDVRP), and in many cases the two stages are solved iteratively in a feedback loop. The most successful heuristic methods for the CLRP use a variety of paradigms: large neighborhood search [14], simulated annealing (SA, see [33]), ant colony optimization [31], memetic algorithms [17], variable neighborhood search (VNS, see [23,13,12]), greedy randomized adaptive search procedure (GRASP, see [10,6]) and matheuristics integrating integer linear programming techniques [26,23,11,6]. For summaries of the individual works, we refer the reader to the surveys of Lopes et al. [16], Prodhon and Prins [27], and Schneider and Drexl [29].

It has been noted in the literature that the quality of a solution strongly depends on the opened facilities (see, e.g. [25]). Therefore, many successful heuristics intensively search the space of potential depot configurations, employing diversification and intensification techniques. GRASP is a frequently used approach in this context (see, e.g., [25,10,22,6]). Apart from this, the CLRP literature does not shed much light on the important question why a certain heuristic performs better than other approaches with a comparable degree of sophistication, or which components of a presented heuristic have the biggest influence on solution quality and runtime.

In this paper, we develop a metaheuristic hybrid combining GRASP for the location phase with a VNS for the routing phase, called GRASP/VNS, to address the CLRP. Instead of a greedy local search, the GRASP phase uses a variable neighborhood descent (VND) to improve the routing solutions given by the randomized construction heuristic. The same VND component is also used as local search within the VNS. The aims of this paper are:

- to propose a conceptually simple heuristic for the CLRP that achieves a reasonable performance on the benchmark sets from the literature. Developing simpler heuristics without losing too much solution quality has been advocated by the research community in recent years. Thus, the design of GRASP/VNS is kept simple, and several known algorithmic components are combined in a new and successful fashion. This design also enables flexible adaptation to other LRP variants as outlined in Section 2.4.
- to provide some insights on which components make a (GRASP-based) metaheuristic effective for the CLRP. We explain several implementation decisions in more detail and study the effects of the main components of our GRASP/VNS on solution quality and runtime. In the development and testing of our method, we find that determining high-quality routing solutions is not only important for the final solution quality but also for an accurate evaluation of the quality of depot configurations. The suboptimal routing solutions used to assess the quality of the investigated depot configurations in the GRASP phase in tendency lead to depot configurations with too many open depots. To counteract this problem, we use a depot configuration refinement phase that carries out a quick assessment of the feasible configurations with a reduced number of open depots. In numerical studies, we show that this algorithmic component and a thorough final routing with the VNS component significantly contribute to the quality of our method. On the commonly used CLRP benchmarks of Tuzun and Burke [32], Barreto [4], and Prins et al. [24], our method provides reasonable results in comparison to the state-of-the-art.

The remainder of this paper is structured as follows. We describe our solution method in detail in Section 2. Our numerical studies to investigate the effect of different components of GRASP/VNS on the solution quality and runtime and to compare our method

```
{GRASP phase (see Section 2.1)}
S_best ← ∅
for n_grasp iterations do
    s ← RECWA( )
    s ← VND(s)
    update set S_best storing the three best solutions
end for
{route improvement phase (see Section 2.2)}
S_best ← shortVNSRun(S_best) {route improvement with short VNS run}
s* ← bestOf(S_best)
{depot configuration refinement phase (see Section 2.3)}
Ω ← generate η_dcr feasible depot configurations with one depot less than s*
for ω ∈ Ω do
    s ← initialSolution(ω) {assign all customers to their closest depot in ω and serve them by dedicated
    routes}
    s ← VND(s)
    s ← shortVNSRun(s)
    if s improves s* then
        s* ← s
    end if
end for
{finalization phase (see last paragraph of Section 2.3)}
s* ← longVNSRun(s*) {improve with long VNS run}
```

Fig. 1. Overview of the GRASP/VNS algorithm.

to the state-of-the-art are described in Section 3, followed by a short conclusion in Section 4.

2. GRASP/VNS for the CLRP

We implement a hybrid solution method combining GRASP for the location phase with a VNS for the routing phase. Fig. 1 gives a pseudocode overview of our GRASP/VNS, which is described in the following.

GRASP/VNS uses the randomized extended Clarke and Wright savings algorithm (RECWA) proposed by Prins et al. [25] to determine the initial depot configuration and vehicle routes, which are subsequently improved by a greedy local search. We note the following problem of a basic GRASP procedure: Depot configurations are evaluated based on the generally rather low-quality routing solutions returned by the local search. This is likely to lead to wrong decisions when comparing the potential of depot configurations, and a configuration that is dominated by other configurations after a thorough routing phase might be assigned the lowest total cost in this step. This is particularly problematic if the depot configuration is fixed during the routing phase, and thus the best configuration has to be found in the *GRASP phase*. To counteract this problem, our algorithm uses the following two measures: (i) we replace the greedy local search by a VND that is able to handle solutions that are infeasible with respect to vehicle and/or depot capacities and aims at repairing such solutions as quickly as possible (see Section 2.1), and (ii) we do not only keep one solution of the GRASP phase but store the three best solutions found during the GRASP phase. Note that the number of stored solutions is a parameter of the algorithm. Our preliminary experiments show that a value

of three provides a reasonable tradeoff between quality and runtime. The routing of these three solutions is improved by means of a *route improvement phase* that carries out a small number of VNS iterations (see Section 2.2) to solve the underlying MDVRP.

We further improve the best solution found this far as follows: We observe a tendency of the GRASP phase to open too many depots because customers cannot be fitted into the suboptimal routes given by the routing solutions found during the GRASP phase. We design a *depot configuration refinement phase* that carries out a fast assessment of the profitability of depot configurations with a reduced number of open depots (see Section 2.3). In this step, the newly generated depot configurations are also improved with short VNS runs to allow for a fair comparison with the current best solution. In the *finalization phase*, our algorithm uses a long VNS run to further improve the routing of the best solution found until then (see Section 2.3).

2.1. The GRASP phase

The core element of the GRASP phase is the RECWA of Prins et al. [25]. Similar to Prins et al. [25], RECWA is repeatedly performed alternating between a diversification and an intensification phase. RECWA first assigns customers to their closest depot with sufficient capacity, serves each customer on a dedicated route from the depot, and closes the facilities with no assigned customers. Subsequently, all feasible mergers of two routes, i.e., mergers that do not violate vehicle or depot capacity restrictions, are evaluated. RECWA evaluates all possibilities of assigning the merged route to available depots. We store the merge moves with the best savings in a restricted candidate list of size l and randomly select the move to perform from this list with uniform probability. In each iteration of the RECWA, the size l is randomly varied within the interval $[1, l_{max}]$, where l_{max} is a user-defined parameter. This procedure is repeated until no feasible merger that leads to a cost reduction can be found.

Subsequently, we improve the routes of the resulting solution by a VND using the neighborhoods defined by the operators in Table 1 in the given order. The order of the operators is chosen by their structural impact on the solution, and in our computational experiments we show that the proposed order is favorable compared to picking a random order each time the VND restarts (see Section 3.3). In each iteration of the VND, the first improving move is executed. The search stops if no improving move can be found. An infeasible solution s is assigned the objective value $F_g(s) = F(s) + M \cdot V_c(s)$, where $F(s)$ denotes the standard objective value, $V_c(s)$ the amount of capacity violation on vehicles and depots, and M is a very large penalty factor. In this way, the VND will always prefer moves that reduce the amount of capacity violation in the solution over moves that improve the routing distance and thus guide the search towards feasible solutions.

Motivated by the insight that the quality of a solution strongly depends on its depot configuration, we carry out the described GRASP procedure for n_{grasp} iterations, alternating between n_{div} diversification and n_{int} intensification iterations as proposed by Prins et al. [25]. Both phases are based on restricting the depots that can be used for the

Table 1
Neighborhood operators of the VND.

Order	Operator	Description
1	Split-1	A new route containing a single customer is opened at any open depot.
2	Relocate-1	A single customer is extracted from a route and inserted into the same route at another position or into any other route.
3	Exchange-1	Two customers from the same or from different routes are swapped.
4	Split-2	A new route containing two consecutive customers is opened at any open depot.
5	Relocate-2	Same as Relocate-1, but two consecutive customers are relocated, preserving their original order.
6	Exchange-2	Same as Exchange-1, but two consecutive customers are swapped, preserving their original order.
7	Split-b	A new route containing a customer i and all its predecessors is opened at any open depot.
8	Split-f	A new route containing a customer i and all its successors is opened at any open depot.
9	Relocate-b	A customer i and all its predecessors are moved to the same or a different route at any position.
10	Relocate-f	A customer i and all its successors are moved to the same or a different route at any position.
11	2-Opt*	Two routes originating at the same depot are reconnected in such a way that the first part of the first route is connected with the second part of the second route and vice versa.
12	Exchange-b	A customer i and all its predecessors are swapped with a customer j and all its predecessors. Customers i and j are on different routes at different depots.
13	Exchange-f	A customer i and all its successors are swapped with a customer j and all its successors. Customers i and j are on different routes at different depots.

initial customer assignment to a subset I' of the available depots I. In the first iteration of the diversification phase, all depots are available and may potentially be opened. In the next iterations, initially only two depots are available to define I': The first one is iteratively selected so that each depot of I was opened at least once, the second one is randomly drawn from the remaining ones. In this way, the solution space is systematically explored, and each depot is used at least once. All customers are sequentially assigned to their closest depot in I'. If a customer cannot be assigned because of a capacity violation, the depot closest to the customer is opened. Subsequently, mergers are performed as described above, i.e., not only depots in I' but all depots are considered as starting and ending location for the merged routes. In the intensification phase, only the depots in I' that are open in the best solution of the preceding diversification phase may be used. Here, not only the assignment of customers is restricted to this subset but also the merge operations during the RECWA.

2.2. Route improvement phase using variable neighborhood search

In general, the vehicle routes generated by the GRASP are of mediocre quality. We apply a short VNS run to further improve the routing by solving the underlying MDVRP of the current CLRP solution. VNS, originally proposed by Mladenović and Hansen [20], searches in increasingly large neighborhoods in order to escape from local optima. Given a set of pre-selected neighborhoods for the so-called shaking step and a generally different neighborhood for the local search: Starting from the current solution $\mathcal{S}$, VNS randomly

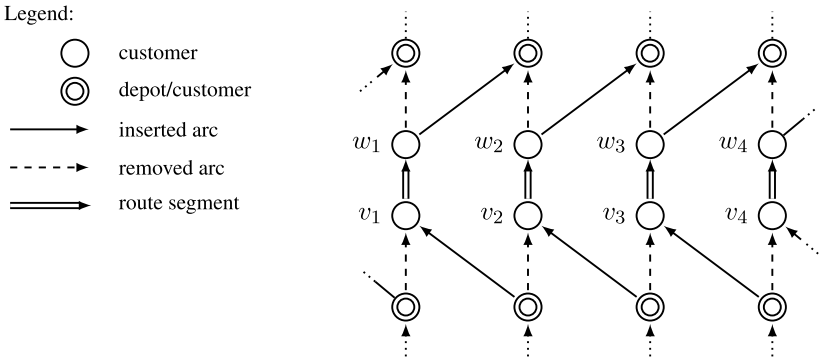


Fig. 2. The cyclic-exchange operator with four routes.

Table 2
Overview of neighborhoods used in the VNS.

Neighborhood	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
# Routes involved	3	3	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	5
Max. sequence length	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6

generates a new solution $\mathcal{S}'$ using the first shaking neighborhood and from there a local search is performed. In our implementation, the VND as described above is used as local search component. If the thus obtained solution $\mathcal{S}''$ has lower quality than the current solution $\mathcal{S}$, the next shaking neighborhood is chosen to repeat the procedure. If a better solution is found or all available shaking neighborhoods have been explored, the search is restarted with the first shaking neighborhood.

We define the shaking neighborhoods by means of a cyclic-exchange operator, which moves customer sequences between routes in a cyclic way [15]. Fig. 2 depicts an example with four routes $i = 1, 2, 3, 4$, in which the customer sequences from v_i to w_i are exchanged between the routes. We distinguish 18 neighborhoods generated by this operator as shown in Table 2. Note that the example in Fig. 2 corresponds to any of the neighborhoods 7–12 in Table 2 because all of these neighborhoods involve exactly four routes.

The routes that are involved in a cyclic exchange are determined in the following way. First, a base route is randomly drawn from the set of all routes. The centroid of a route is given by the average Cartesian coordinates of all customers on the route and of the depot that the route originates from. Second, all routes are sorted in increasing order of the distance between their centroid and the centroid of the base route, and the number of routes to be involved in the cyclic exchange move are taken from the beginning of the list. The number of customers to be exchanged is randomly drawn from the interval from 1 up to the given maximum sequence length. One draw is made for each individual route.

Note that the resulting solution may be infeasible with respect to capacity constraints. This solution is improved (and repaired in case it is infeasible) by applying the local

search, i.e., the VND described in Section 2.1. The short VNS run in the route improvement phase terminates after n_{short} iterations without improvement.

2.3. Depot configuration refinement and finalization phase

According to our computational experience, the depot configurations determined by the GRASP are not optimal in most cases. As described above, the GRASP tends to open too many depots because customers cannot be fitted into the suboptimal routes of the routing solutions determined during the GRASP phase. To remedy this drawback, we use a depot configuration refinement phase that systematically evaluates depot configurations with a reduced number of depots. Starting from the best depot configuration found by the GRASP and route improvement phase, we determine η_{dcr} feasible configurations with exactly one depot less, restricted to those for which the total capacity of open depots is sufficient to serve the total customer demand. More precisely, we sample random configurations with one depot less until η_{dcr} feasible configurations have been found, or all possible configurations have been evaluated.

We randomly select one depot configuration out of the resulting list and assign all customers to their closest depot, from which each customer is initially served by a dedicated route. The resulting solution may be infeasible with respect to depot capacity. First, it is improved (or repaired in case the solution is infeasible) by one run of the VND described in Section 2.1. Next, the routing is further improved with a short VNS run that stops after n_{short} iterations without improvement. This allows for a fair comparison of the resulting solution with the current best solution. If the resulting solution improves upon the current best solution, it becomes the new best solution, otherwise it is discarded. The described procedure is repeated until we have examined η_{dcr} depot configurations.

Note that there may be less than η_{dcr} feasible depot configurations with one depot less than in the best solution determined by GRASP. In this case, we increase the number of open depots by one and randomly generate further configurations until we have investigated η_{dcr} depot configurations. We decided against investigating depot configurations with even less open depots than the best found depot configuration minus one because a configuration with such a small number of depots is unlikely to yield good objective values due to high routing costs.

In the finalization phase, we try to further improve the routing of the best solution found during the depot configuration refinement phase by carrying out a long VNS run that terminates after n_{final} iterations without improvement.

2.4. Flexibility of GRASP/VNS to be adapted to other problem variants

GRASP/VNS uses algorithmic components that allow a straightforward adaptation of the method to other LRP variants that are characterized by additional routing constraints, like for example time windows, simultaneous pickup-and-delivery, or precedence

constraints such as line- and backhauls. This is possible due to the following properties of the components:

- GRASP: the RECWA is based on the principle of a savings heuristic, which can easily be adapted to handle additional routing constraints in the merge phase because mergers resulting in infeasible routes are discarded. The use of the restricted candidate list and the availability of depots in the diversification and intensification phases of the GRASP are not affected by additional routing constraints. The GRASP also contains a VND component, whose flexibility is outlined below.
- VND: Additional routing constraints require an extension of the generalized objective function by additional penalty terms to handle solutions that are infeasible with regard to the new constraints. The computation of the new penalty terms must be implemented for all neighborhood operators. Note that a selection of specific neighborhood operators that have proven effective for the given routing constraints is easily possible.
- VNS: The shaking neighborhood does not require any modification because potentially infeasible solutions with regard to the new constraints can be handled by the VND as described above.

3. Computational studies

This section presents our numerical experiments to (i) investigate the effect of the core components of our algorithm on runtime and solution quality (Sections 3.3 and 3.4), and (ii) compare GRASP/VNS to the state-of-art (Section 3.5). The utilized benchmark sets are introduced in Section 3.1, and the parameter setting of the algorithm is described in Section 3.2.

3.1. Benchmark instances

The benchmark set proposed by Barreto [4], called Barreto set in the following, consists of 19 instances with up to 318 customers and 15 depots, of which most are adapted from classical VRP instances. The comparison methods from the literature only use a subset of 13 instances from the set, and therefore, we only consider these 13 instances in our computational studies to enable a meaningful comparison. Nevertheless, we report the results on the remaining 6 instances in Table 10 of Appendix B. On the Barreto set, vehicle capacities are generally limited, and in 4 of the 13 instances used in our computational studies, the depot capacities are also restricted. Tuzun and Burke [32] present a set of 36 instances with capacitated vehicles and without depot capacities, called Tuzun-Burke set. The instances go from 100 customers and 10 depots up to 200 customers and 20 depots. Instances with customer clusters of varying density and instances with uniformly distributed customers are contained in the set. The benchmark of Prins et al. [24], called Prodhon set, consists of 30 instances ranging from 20 customers

Table 3
Parameter tuning of GRASP/VNS. The values Δ_a are average gaps to the BKS.

l_{max}	1	2	3	4	5	6	7
Δ_a	0.75	0.80	0.68	0.74	0.77	0.72	0.68
t	46.06	46.57	45.39	46.44	46.20	46.35	48.39
n_{div}/n_{int}	1/7	2/6	3/5	4/4	5/3	6/2	7/1
Δ_a	0.92	0.84	0.67	0.78	0.77	0.71	0.77
t	43.22	42.62	43.42	45.95	47.86	50.15	51.04

and 5 depots to instances with up to 200 customers and 10 depots. All instances are restricted in terms of vehicle and depot capacity, and the set comprises instances with two or three customer clusters as well as instances with uniformly distributed customers.

3.2. Parameter setting and computational environment

Most of the algorithm parameters are iteration limits that define the termination criterion of the different algorithmic components. These parameters control the tradeoff between solution quality and runtime, and we set them to the following values that provided a desirable tradeoff during the algorithm development: $n_{grasp} = 160$, $n_{short} = 10$, $\eta_{dcr} = 15$, and $n_{final} = 600$.

For the remaining parameters, we have conducted a simple parameter tuning which aims at determining a decent parameter setting while avoiding to overfit the setting to the problem instances under consideration. To this end, we have used a restricted set of 43 instances which comprises only those instances from the three benchmark sets with up to 100 customers and 10 depots. We start from the following base setting found during the development of GRASP/VNS: size of the restricted candidate list $l_{max} = 4$, number of diversification iterations $n_{div} = 4$, and number of intensification iterations $n_{int} = 4$. In a first step, we alter the value of l_{max} in the range $1, \ldots, 7$. For each value, we carry out five runs for all instances and report the average percentage gap Δ_a to the best known solution (BKS) from Arnold and Sörensen [3] and the average runtime t in Table 3. We detect the lowest gap of $\Delta_a = 0.68$ together with the shortest average runtime at $l_{max} = 3$, which we choose as the final value for l_{max}. Next, we test the values $n_{div}/n_{int} = 1/7, 2/6, \ldots, 7/1$ and find the lowest gap of $\Delta_a = 0.67$ at $n_{div}/n_{int} = 3/5$ with an average runtime of 43.42 seconds that is very close to the overall minimum of 42.62 seconds. Thus, we choose the final parameter values $n_{div} = 3$ and $n_{int} = 5$.

The experiment clearly shows that our algorithm is rather stable with regard to modifications of the parameter values within reasonable ranges. We summarize the final setting of all parameters in Table 4.

GRASP/VNS is implemented in C++, and we ran all experiments on a single core of an Intel Xeon E5-2430v2 CPU clocked at 2.5 GHz with 64 GB RAM, running CentOS 7. The memory consumption stayed below 256 MB at all times.

Table 4
Final parameter setting of GRASP/VNS.

GRASP				Stopping criteria		
l_{max}	n_{grasp}	n_{div}	n_{int}	η_{dcr}	n_{short}	n_{final}
3	160	3	5	15	10	600

Table 5
Results of the experiment on the effectiveness of VND neighborhood operators.

Removed operator	$Diff_{\Delta_a}$	Δ_t	p_F	p_t
Split-1	0.15	37.59	0.99	1.00
Relocate-1	0.24	-1.98	1.00	0.49
Exchange-1	0.01	1.42	0.93	1.00
Split-2	-0.03	-0.97	0.03	0.66
Relocate-2	0.23	-2.59	1.00	0.99
Exchange-2	0.01	1.08	0.12	1.00
Split-b	-0.02	-0.57	0.27	0.57
Split-f	-0.06	-0.37	0.10	0.31
Relocate-b	0.05	0.84	0.94	0.98
Relocate-f	0.00	1.79	0.64	1.00
2-Opt*	0.08	0.05	1.00	0.13
Exchange-b	-0.01	0.19	0.90	0.17
Exchange-f	-0.05	0.77	0.04	0.97

3.3. Effect of selection and order of neighborhood operators

In this section, we investigate the effect of selecting the neighborhood operators and their execution order in the VND on the performance of GRASP/VNS. The first experiment investigates whether the set of neighborhood operators can be reduced without compromising the performance. We investigate all algorithmic variants that emerge by removing one of the neighborhood operators from Table 1 at a time. For each variant, we carry out five runs on all benchmark instances. In Table 5, we report the absolute change in average percentage gap to the BKS ($Diff_{\Delta_a}$) and the percentage change in average runtime (Δ_t) compared to the base setting. The statistical significance of the results is investigated using a two-sided Wilcoxon signed-rank test. Column p_F reports the p-value of the hypothesis that objective values obtained by the two algorithm variants originate from different distributions. Column p_t reports the analogous p-value for the runtimes.

The results show that there is no operator whose removal from the VND improves the solution quality with statistical significance above 90%. In particular, the largest improvement is obtained for Split-f with a significance level of only 10%. Therefore, we keep all operators from Table 1 to define the set of VND neighborhoods.

It is interesting to note that removing Split-1 increases the runtime of the algorithm by more than 35%. This can be explained by the fact that Split-1 is able to repair infeasible solutions quickly by opening additional routes and is placed at the first position of the evaluation order. Thus, it seems recommendable that the first position in the evaluation order is occupied by one of the Split operators if infeasible solutions are allowed.

Table 6
Comparison of eight different algorithm configurations on the Tuzun-Burke, Prodhon, and Barreto benchmark sets. We report aggregate results as average percentage gaps of each configuration to the BKS as reported in Arnold and Sörensen [3] (Δ_a), and the average runtime (t) in seconds.

Configuration	All instances		Tuzun-Burke		Prodhon		Barreto	
	Δ_a	t	Δ_a	t	Δ_a	t	Δ_a	t
G	3.12	114.53	3.71	180.74	3.16	75.90	1.37	20.29
G ▸ F	1.74	200.91	1.82	305.97	2.11	143.73	0.65	41.89
G ▸ RI	2.39	117.39	2.75	184.54	2.55	78.56	1.04	21.04
G ▸ RI ▸ F	1.72	204.75	1.78	312.35	2.09	147.44	0.67	39.03
G ▸ DCR	2.52	154.47	3.68	240.30	1.73	105.87	1.16	28.92
G ▸ DCR ▸ F	1.21	241.07	1.79	361.06	0.82	178.94	0.52	52.16
G ▸ RI ▸ DCR	1.91	149.99	2.70	231.41	1.40	104.46	0.92	29.56
G ▸ RI ▸ DCR ▸ F	1.20	233.62	1.72	348.82	0.85	174.36	0.55	51.35

The second experiment investigates the impact of the order in which the operators are evaluated on the performance of GRASP/VNS. We use the complete set of neighborhood operators but randomly shuffle their order in the VND in each run. We carry out five runs on all problem instances. Compared to the variant with the fixed neighborhood order of Table 1, the average gap to the BKS reduces by 0.12%, and the average runtime increases by 53.46%. For both the differences in solution quality and runtime, the two-sided Wilcoxon signed-rank test determines a significance level of at least 0.99. We consider the increase in runtime large compared to the obtained improvement in solution quality, and we therefore decide to keep the order of neighborhood operators fixed within the VND.

3.4. Effect of algorithm components

Our GRASP/VNS comprises four components: the GRASP phase (G), the route improvement phase (RI), the depot configuration refinement phase (DCR), and the finalization phase (F). In this section, we evaluate in more detail how each component contributes to the solution quality and runtime of our algorithm in order to be able to derive insights on the design of successful metaheuristics for the CLRP. To this end, we investigate eight different configurations of our algorithm, which are uniquely identified as combinations of the four components G, RI, DCR, and F. Table 6 presents aggregate results of five runs of the respective configuration over all instances and over the individual instance sets. The solution of an individual run is the best solution found by the configuration until its last component has completed. We report the average percentage gap Δ_a to the best known solution (BKS) from Arnold and Sörensen [3] and the average runtime t. The results of the complete GRASP/VNS, i.e., of configuration G ▸ RI ▸ DCR ▸ F, on a per instance basis can be found in Tables 8–10 in Appendix B.

We make the following observations:

- The final configuration containing all components, i.e., G ▸ RI ▸ DCR ▸ F, outperforms all other configurations with regard to solution quality, which shows that each component contributes to the final quality.
- The finalization phase is a vital component of GRASP/VNS because it is able to significantly improve solution quality whenever it is added to a configuration. In general, these large improvements justify the additional runtime of F, which, e.g., amounts to roughly 33% of the runtime of the complete algorithm configuration G ▸ RI ▸ DCR ▸ F. If more emphasis is put on runtime, G ▸ RI ▸ DCR provides relatively decent quality in shorter time. However, in practice, it is likely that better results can be obtained by keeping F and reducing the number of its VNS iterations.
- A comparison of configurations that do not include F appears to be misleading because the full potential of the identified depot configurations can only be harvested with a high-quality routing solution as explained in Section 2. For example, although G ▸ DCR performs clearly worse than G ▸ RI, including F reveals that configuration G ▸ DCR ▸ F provides notably better solution quality than G ▸ RI ▸ F at the cost of a moderate increase in runtime of 15%.
- The contribution of DCR is much larger on instances in which depot costs are the dominating factor than on instances in which routing costs are the main cost factor. Comparing the performance of G ▸ RI ▸ DCR ▸ F to that of G ▸ RI ▸ F, we can see that significant improvements in solution quality are only obtained on the Prodhon set, whereas only small improvements can be found on the Barreto set and the Tuzun-Burke set. In the instances of the latter two sets, depot opening costs are small compared to routing costs, and therefore, closing a depot is less likely to improve overall costs. As a consequence, to save runtime, it may be useful to disable DCR if tackling instances with such a special cost structure. However, in real-world instances, the opening costs of locations generally outweigh the routing costs, and therefore, including DCR is beneficial.

3.5. Comparison to the state-of-the-art

In this section, we compare the solution quality and runtime of GRASP/VNS to the most effective heuristic methods for the CLRP from the literature on the Tuzun-Burke, Prodhon, and Barreto benchmark. In Table 7, for each method, the average gap to the BKS (taken from Arnold and Sörensen [3]), and the average runtime is reported for each separate instance set and over all three instance sets. Results are provided for the following comparison methods (the way in which the respective authors report results and runtimes is explained in the following):

- DLPP [10] give the best solution obtained in 5 runs and the time of the best run.
- YLLT [33], ELT [11], ELBT [12], and AS-PF$_1$ [3] provide results of a single run.
- For the two variants of HCC [14] and for SL-TBSA$_{speed}$ [30], the table reports the best result of 5 runs and the average runtime per run.

Table 7
Comparison of GRASP/VNS to the state-of-the-art. For each method, we report the average gap of the best solution found by each method to the BKS as reported in Arnold and Sörensen [3] (Δ_b), and the runtime (t) in seconds on the Tuzun-Burke, Prodhon, and Barreto benchmark sets. For HCC–500K, HCC–5000K, CCG, LFS-Hybrid GA, LFS-Hybrid GA+, SL-TBSA$_{speed}$ and GRASP/VNS, we also report the average percentage gaps of the average objective value achieved in several runs (Δ_a). In addition, we report the benchmarking environment used for each method and a common time measure based on the passmark score of the processor used in the testing.

Benchmark	DLPP		YLLT		HCC–500K			HCC-5000K			ELT		CCG		
	Δ_b	t	Δ_b	t	Δ_b	Δ_a	t	Δ_b	Δ_a	t	Δ_b	t	Δ_b	Δ_a	t
Tuzun-Burke	1.40	607	1.59	826	0.53	0.99	166	0.39	0.60	1621	1.24	392	0.27	0.66	2590
Prodhon	1.17	258	0.51	422	0.50	0.78	90	0.40	0.51	844	0.62	176	0.18	0.38	1163
Barreto	0.04	188	0.25	161	0.12	0.21	35	0.02	0.05	354	0.74	105	0.15	0.66	279
Average	1.09	405	0.96	564	0.45	0.78	116	0.34	0.47	1117	0.93	263	0.22	0.55	1685
Processor type	Core 2 Quad		Core 2 Quad		Opteron 275			Opteron 275			Core 2 Duo		Xeon E5462		
Processor speed	2.83 GHz		2.66 GHz		2.20 GHz			2.20 GHz			2.00 GHz		3.00 GHz		
Passmark score	1222		1159		685			685			792		1219		
Corrected time	344 × 5		454		55 × 5			532 × 5			145		1428 × 10		

Benchmark	TC		ELBT		LFS-Hybrid GA			LFS-Hybrid GA+			SL-TBSA$_{speed}$			AS-PF$_1$		GRASP/VNS		
	Δ_b	t	Δ_b	t	Δ_b	Δ_a	t	Δ_b	Δ_a	t	Δ_b	Δ_a	t	Δ_b	t	Δ_b	Δ_a	t
Tuzun-Burke	1.33	202	0.86	201	0.96	1.55	86	0.81	1.19	364	0.31	0.57	66	0.34	160.86	0.88	1.72	351.19
Prodhon	0.45	191	0.43	91	0.43	0.71	73	0.37	0.58	199	0.10	0.20	32	0.14	104.60	0.33	0.85	174.36
Barreto	0.07	49	0.63	53	0.04	0.26	19	0.00	0.06	93	0.00	0.08	10	0.04	86.23	0.11	0.55	51.35
Average	0.80	174	0.66	135	0.61	1.02	70	0.51	0.77	257	0.18	0.35	44	0.21	127.22	0.54	1.20	233.62
Processor type	XP 2500+		Core 2 Duo		i7-4790			i7-4790			Xeon E5-2670			i7-4790		Xeon E5-2430v2		
Processor speed	1.83 GHz		2.00 GHz		3.60 GHz			3.60 GHz			2.60 GHz			3.60 GHz		2.50 GHz		
Passmark score	544		792		2226			2226			1652			2226		1439		
Corrected time	66 × 10		74		108 × 5			397 × 5			51 × 5			197		234 × 5		

- For CCG [6], TC [31], two variants of LFS [17], and GRASP/VNS, the table shows the best of 10 runs and the average runtime per run.

Furthermore, the table reports the processor used and the Passmark score for a single core of the respective processor (see www.cpubenchmark.net). Although an entirely precise comparison of runtimes is not possible due to different operating systems and programming languages, we use this information to translate all runtimes into a common time measure based on the CPU we used for testing our algorithm. In addition, the runtimes of the methods are multiplied by the number of runs on which the reported solution quality is based.

The performance of GRASP/VNS is quite reasonable in comparison to the state-of-the-art. Considering the average solution quality and runtime for each of the three benchmark sets, only the recently proposed methods TBSA$_{speed}$ and AS are able to dominate GRASP/VNS, i.e., all other comparison methods are either slower or achieve worse solution quality on at least one of the three benchmark sets. On the other hand, GRASP/VNS achieves almost the same solution quality as the recently proposed LFS-Hybrid GA+ while requiring less runtime. Regarding the average solution quality Δ_a, HCC–500K, LFS-Hybrid GA, and SL-TBSA$_{speed}$ are superior to GRASP/VNS regarding the tradeoff in solution quality and runtime. Considering the difference between Δ_b and Δ_a, GRASP/VNS seems slightly less robust than the comparison methods, however, the deviations are still within a reasonable range. Given the conceptual simplicity of GRASP/VNS, the efficiency and robustness seem quite reasonable.

In particular, we want to highlight the simplicity of our algorithm compared to the two dominating algorithms SL-TBSA$_{speed}$ and AS. The depot configuration search of GRASP/VNS is based on a straightforward sampling of available depots and the assignment of customers to their nearest depot plus the newly introduced depot configuration refinement phase. Routes are constructed by means of a randomized savings heuristic and improved using a local search within a VNS. In contrast, SL-TBSA$_{speed}$ features a sophisticated location phase operating on a depot configuration neighborhood defined by swap and invert operators. The search uses lower bounding techniques, transformation of existing routing solutions to new depot configurations, and a rating function to judge the potential of depot configurations which have not yet been evaluated by the routing component. The routing phase is a granular tabu search with continuous sparsification, a more complex penalty mechanism, and a dynamic adaptation of the sparsification strength during the search. The algorithm traverses several intensity levels, which eliminate unpromising depot configurations and increase routing precision in a stepwise fashion. Similarly, AS uses a progressive filtering framework that iteratively reduces the number of depot configurations based on an ever-increasing precision of the routing solution. The depot configuration search relies on the estimation of an upper bound for the number of open depots in promising solutions based on an empirically determined function to estimate the cost reduction when opening additional depots. If it is not possible to enumerate all depot configurations up to this lower bound, a grid-based

sampling of promising depots based on depot gross costs (estimated routing costs plus depot opening costs divided by depot capacity) is carried out. The routes are determined using a knowledge guided search that is able to solve several VRP variants in state-of-the-art fashion that is described in the separate paper of Arnold and Sörensen [2].

4. Conclusion

In this paper, we propose a hybrid of GRASP and VNS to solve the CLRP. The design of the algorithm is kept simple, and several known algorithmic components are combined in a new and successful fashion. We note that a thorough but fast routing method like our VNS contributes to the quality of the method. Still, a location phase using suboptimal routing solutions to assess the quality of depot configurations is likely to generate solutions with too many open depots. We propose a depot configuration refinement phase that aims at reducing the number of opened depots and is able to significantly increase the solution quality of the method with only a moderate increase in runtime, especially on instances in which depot costs are the dominating cost factor. This algorithmic component is likely to be beneficial if added to other CLRP algorithms that determine the final depot configuration using GRASP or similar methods. Despite its conceptual simplicity, the final configuration of GRASP/VNS provides reasonable solution quality on the standard benchmark instances from the literature.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Mathematical model

We present a simple mixed-integer programming formulation of the CLRP. A compact formulation that requires fewer variables can be found in uit het Broek et al. [5]. Let I denote the set of depot locations and J the set of customer locations, and let $V = I \cup J$. The cost for traveling from node $i \in V$ to $j \in V$ is denoted as c_{ij} and is allowed to be asymmetric in the presented model. A depot location $i \in I$ has associated opening costs O_i and a capacity of W_i demand units. An unlimited homogeneous vehicle fleet is available at each depot location. The fixed costs and the capacity of the vehicles are denoted as F and Q, respectively. The demand of customer $j \in J$ is d_j units. We introduce the following three sets of binary decision variables:

- x_{ij} indicates that a vehicle travels from node $i \in V$ to $j \in V$,
- w_{ji} indicates that customer $j \in J$ is served by depot $i \in I$,
- y_i indicates that depot $i \in I$ is opened.

Using the above definitions, the CLRP can be formulated as follows:

$$\min \sum_{i \in V} \sum_{j \in V} x_{ij} c_{ij} + \sum_{i \in I} \sum_{j \in J} x_{ij} F + \sum_{i \in I} y_i O_i \tag{1}$$

subject to

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{i \in V} x_{ji} = 1 \qquad \forall j \in J \tag{3}$$

$$\sum_{i \in I} w_{ji} = 1 \qquad \forall j \in J \tag{4}$$

$$w_{ji} \leq y_i \qquad \forall i \in I, j \in J \tag{5}$$

$$\sum_{j \in J} w_{ij} d_j \leq W_i \qquad \forall i \in I \tag{6}$$

$$x_{jk} \leq 1 - w_{ji} + w_{ki} \qquad \forall i \in I, j \in J, k \in J \tag{7}$$

$$x_{jk} \leq 1 + w_{ji} - w_{ki} \qquad \forall i \in I, j \in J, k \in J \tag{8}$$

$$x_{ij} \leq w_{ji} \qquad \forall i \in I, j \in J \tag{9}$$

$$x_{ji} \leq w_{ji} \qquad \forall i \in I, j \in J \tag{10}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - \left\lceil \frac{1}{Q} \sum_{i \in S} d_i \right\rceil \qquad \forall S \subseteq J \tag{11}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in V \tag{12}$$

$$w_{ji} \in \{0, 1\} \qquad \forall i \in I, j \in J \tag{13}$$

$$y_i \in \{0, 1\} \qquad \forall i \in I \tag{14}$$

The objective (1) is to minimize the sum of total routing, fixed vehicle cost, and depot opening cost. Constraints (2) and (3) ensure that every customer is served by exactly one vehicle and constraints (4) guarantee that every customer is assigned to exactly one depot. Customers can only be served by open depots according to constraints (5). Constraints (6) ensure that the depot capacity is respected. Constraints (7) and (8) only allow an arc between two customers to be traveled if both customers are assigned to the same depot. Constraints (9) and (10) ensure that a route starts and ends at the depot which serves the customers on the respective route. Constraints (11) are extended subtour elimination constraints, which also serve as vehicle capacity constraints. The binary decision variables are defined in (12)–(14).

Table 8
Detailed results of GRASP/VNS on the Tuzun-Burke instance set.

Instance	BKS	GRASP/VNS				
		f_b	Δ_b	f_a	Δ_a	t
111112	1467.68	1496.79	1.98	1501.23	2.29	64.67
111122	1448.37	1465.07	1.15	1478.13	2.05	80.93
111212	1394.80	1396.59	0.13	1403.67	0.64	59.18
111222	1432.29	1452.76	1.43	1468.63	2.54	77.42
112112	1167.16	1167.53	0.03	1172.31	0.44	68.54
112122	1102.24	1102.24	0.00	1122.54	1.84	99.74
112212	791.66	791.66	0.00	794.76	0.39	97.35
112222	728.30	728.30	0.00	729.58	0.18	90.53
113112	1238.24	1239.22	0.08	1252.31	1.14	74.41
113122	1245.30	1247.27	0.16	1253.28	0.64	86.34
113212	902.26	902.26	0.00	903.19	0.10	70.21
113222	1018.29	1018.29	0.00	1019.58	0.13	88.79
121112	2237.73	2266.41	1.28	2276.43	1.73	574.20
121122	2137.45	2224.69	4.08	2251.84	5.35	680.64
121212	2195.17	2204.93	0.44	2217.90	1.04	557.52
121222	2214.86	2252.00	1.68	2285.60	3.19	661.80
122112	2070.43	2077.49	0.34	2083.35	0.62	703.71
122122	1685.52	1768.04	4.90	1812.92	7.56	926.60
122212	1449.93	1458.39	0.58	1472.59	1.56	697.08
122222	1082.46	1086.15	0.34	1088.09	0.52	656.32
123112	1942.23	2000.03	2.98	2044.21	5.25	595.35
123122	1910.08	1942.73	1.71	1972.26	3.26	769.57
123212	1761.11	1766.68	0.32	1773.79	0.72	690.39
123222	1390.86	1393.85	0.21	1446.44	4.00	703.92
131112	1892.17	1932.04	2.11	1939.02	2.48	229.69
131122	1819.68	1849.43	1.64	1865.33	2.51	258.60
131212	1960.02	1966.12	0.31	1971.38	0.58	232.90
131222	1792.77	1799.17	0.36	1809.42	0.93	273.02
132112	1443.32	1446.17	0.20	1459.92	1.15	260.13
132122	1429.30	1441.75	0.87	1445.90	1.16	344.01
132212	1204.42	1205.24	0.07	1218.19	1.14	255.90
132222	924.68	932.99	0.90	934.06	1.01	317.72
133112	1694.18	1698.84	0.28	1711.99	1.05	281.40
133122	1392.01	1398.26	0.45	1417.80	1.85	309.38
133212	1197.95	1198.85	0.08	1202.37	0.37	276.36
133222	1151.37	1156.99	0.49	1157.98	0.57	343.36

Appendix B. Additional computational results

We present the detailed results of GRASP/VNS on the Tuzun-Burke (Table 8), Prodhon (Table 9) and Barreto (Table 10) instance sets. For each instance, we report the BKS as given in Arnold and Sörensen [3], the result of the best of five runs of GRASP/VNS as absolute objective function value and as percentage gap to the BKS, and the average runtime in seconds. Optimal solutions according to Contardo et al. [7] are underlined. New BKS found by our algorithm are printed in **bold**.

References

[1] M. Albareda-Sambola, Location-routing and location-arc routing, in: G. Laporte, S. Nickel, F. Saldanha da Gama, M. Albareda-Sambola (Eds.), Location Science, Springer, 2015, pp. 399–418 (chapter 15).

Table 9
Detailed results of GRASP/VNS on the Prodhon instance set.

Instance	BKS	GRASP/VNS				
		f_b	Δ_b	f_a	Δ_a	t
20-5-1a	54793	54793	0.00	54838.6	0.08	0.78
20-5-1b	39104	39104	0.00	39104.0	0.00	0.67
20-5-2a	48908	48908	0.00	48908.0	0.00	0.76
20-5-2b	37542	37542	0.00	37542.0	0.00	0.65
50-5-1a	90111	90111	0.00	90111.0	0.00	7.95
50-5-1b	63242	63242	0.00	63242.0	0.00	8.59
50-5-2a	88293	88298	0.01	88606.0	0.35	8.52
50-5-2b	67308	67308	0.00	67674.6	0.54	9.18
50-5-2bbis	51822	51822	0.00	51833.6	0.02	8.98
50-5-2bis	84055	84055	0.00	84055.0	0.00	7.90
50-5-3a	86203	86203	0.00	86370.8	0.19	7.78
50-5-3b	61830	61830	0.00	61830.0	0.00	7.59
100-5-1a	274814	275441	0.23	276026.8	0.44	70.15
100-5-1b	213568	214133	0.26	214384.2	0.38	70.81
100-5-2a	193671	193752	0.04	194116.6	0.23	82.00
100-5-2b	157095	157150	0.04	157208.4	0.07	61.93
100-5-3a	200079	200217	0.07	200556.4	0.24	64.37
100-5-3b	152441	153331	0.58	154007.0	1.03	57.29
100-10-1a	287661	288295	0.22	289423.4	0.61	78.81
100-10-1b	230989	233105	0.92	233742.0	1.19	87.95
100-10-2a	243590	243695	0.04	248613.4	2.06	75.65
100-10-2b	203988	203988	0.00	206488.6	1.23	67.50
100-10-3a	250882	250971	0.04	260501.0	3.83	71.87
100-10-3b	203114	210089	3.43	214356.2	5.53	79.76
200-10-1a	474850	478224	0.71	489848.4	3.16	752.03
200-10-1b	375177	384431	2.47	385469.8	2.74	735.75
200-10-2a	448077	448547	0.10	449783.8	0.38	642.16
200-10-2b	373696	374217	0.14	374569.0	0.23	683.19
200-10-3a	469433	470879	0.31	471686.8	0.48	661.82
200-10-3b	362320	363212	0.25	363951.2	0.45	818.25

[2] F. Arnold, K. Sörensen, Knowledge-guided local search for the vehicle routing problem, Comput. Oper. Res. 105 (2019) 32–46.
[3] F. Arnold, K. Sörensen, A progressive filtering heuristic for the location-routing problem and variants, Comput. Oper. Res. 129 (2021) 105–166.
[4] S. Barreto, Análise e Modelização de Problemas de localização-distribuição (Analysis and modelling of location-routing problems), PhD thesis, University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal, 2004.
[5] M.A.J. uit het Broek, A.H. Schrotenboer, B. Jargalsaikhan, K.J. Roodbergen, L.C. Coelho, Asymmetric multidepot vehicle routing problems: valid inequalities and a branch-and-cut algorithm, Oper. Res. 69 (2) (2021) 380–409.
[6] C. Contardo, J.-F. Cordeau, B. Gendron, A GRASP + ILP-based metaheuristic for the capacitated location-routing problem, J. Heuristics 20 (2014) 1–38.
[7] C. Contardo, J.-F. Cordeau, B. Gendron, An exact algorithm based on cut-and-column generation for the capacitated location-routing problem, INFORMS J. Comput. 26 (1) (2014) 88–102.
[8] R. Cuda, G. Guastaroba, M.G. Speranza, A survey on two-echelon routing problems, Comput. Oper. Res. 55 (2015) 185–199.
[9] M. Drexl, M. Schneider, A survey of variants and extensions of the location-routing problem, Eur. J. Oper. Res. 241 (2) (2014) 283–308.
[10] C. Duhamel, P. Lacomme, C. Prins, C. Prodhon, A GRASP×ELS approach for the capacitated location routing problem, Comput. Oper. Res. 37 (11) (2010) 1912–1923.
[11] J. Escobar, R. Linfati, P. Toth, A two-phase hybrid heuristic algorithm for the capacitated location-routing problem, Comput. Oper. Res. 40 (1) (2013) 70–79.
[12] J.W. Escobar, R. Linfati, M.G. Baldoquin, P. Toth, A granular variable tabu neighborhood search for the capacitated location-routing problem, Transp. Res., Part B, Methodol. 67 (2014) 344–356.

Table 10
Detailed results of GRASP/VNS on the Barreto instance set. BKS marked with † are taken from Contardo et al. [7]. BKS marked with ‡ are taken from http://sweet.ua.pt/sbarreto/_private/SergioBarretoHomePage.htm.

Instance	BKS	GRASP/VNS				
		f_b	Δ_b	f_a	Δ_a	t
Instances included in comparison						
Christofides69-50x5	565.6	565.6	0.00	580.0	2.55	7.44
Christofides69-75x10	848.8	848.9	0.01	852.2	0.40	27.06
Christofides69-100x10	833.4	837.5	0.50	844.3	1.30	63.86
Daskin95-88x8	355.8	355.8	-0.00	359.0	0.91	46.31
Daskin95-150x10	43919.9	44194.0	0.62	44281.9	0.82	344.88
Gaskell67-21x5	424.9	424.9	0.00	424.9	0.00	0.85
Gaskell67-22x5	585.1	585.1	0.00	585.1	0.00	0.75
Gaskell67-29x5	512.1	512.1	0.00	512.1	0.00	1.85
Gaskell67-32x5-1	562.2	562.2	0.00	562.2	0.00	2.32
Gaskell67-32x5-2	504.3	504.3	0.01	504.3	0.01	2.18
Gaskell67-36x5	460.4	460.4	-0.01	460.4	-0.01	2.50
Min92-27x5	3062.0	3062.0	0.00	3062.0	0.00	1.55
Min92-134x8	5709.0	5724.4	0.27	5772.5	1.11	166.02
Instances excluded from comparison						
Or76-117x14	12474.2‡	**12296.9**	-1.42	12367.7	-0.85	102.78
Perl83-12x2	204.0†	204.0	0.00	204.0	0.00	0.15
Perl83-55x15	1112.1†	1116.2	0.38	1121.8	0.87	10.16
Perl83-85x7	1622.5†	1626.1	0.22	1628.3	0.36	31.24
Perl83-318x4-8000	747619.0‡	**661564.8**	-11.51	666500.0	-10.85	2356.78
Perl83-318x4-25000	580680.2‡	**565516.2**	-2.61	567996.7	-2.18	2021.03

[13] N. Ghaffari-Nasab, M.S. Jabalameli, M.B. Aryanezhad, A. Makui, Modeling and solving the bi-objective capacitated location-routing problem with probabilistic travel times, Int. J. Adv. Manuf. Technol. 67 (9) (2012) 2007–2019.
[14] V. Hemmelmayr, J.-F. Cordeau, T.G. Crainic, An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics, Comput. Oper. Res. 39 (12) (2012) 3215–3228.
[15] T. Ibaraki, S. Imahori, M. Kubo, T. Matsuda, T. Uno, M. Yagiura, Effective local search algorithms for routing and scheduling problems with general time-window constraints, Transp. Sci. 39 (2) (2005) 206–232.
[16] R.B. Lopes, C. Ferreira, B.S. Santos, S. Barreto, A taxonomical analysis, current methods and objectives on location-routing problems, Int. Trans. Oper. Res. 20 (6) (2013) 795–822.
[17] R.B. Lopes, C. Ferreira, B.S. Santos, A simple and effective evolutionary algorithm for the capacitated location-routing problem, Comput. Oper. Res. 70 (2016) 155–162.
[18] S.T.W. Mara, R. Kuo, A.M.S. Asih, Location-routing problem: a classification of recent research, Int. Trans. Oper. Res. 28 (6) (2021) 2941–2983.
[19] H. Min, V. Jayaraman, R. Srivastava, Combined location-routing problems: a synthesis and future research directions, Eur. J. Oper. Res. 108 (1) (1998) 1–15.
[20] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (11) (1997) 1097–1100.
[21] G. Nagy, S. Salhi, Location-routing: issues, models and methods, Eur. J. Oper. Res. 177 (2) (2007) 649–672.
[22] V.-P. Nguyen, C. Prins, C. Prodhon, Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking, Eur. J. Oper. Res. 216 (1) (2012) 113–126.
[23] S. Pirkwieser, G. Raidl, Variable neighborhood search coupled with ILP-based very large neighborhood searches for the (periodic) location-routing problem, in: M. Blesa, C. Blum, G. Raidl, A. Roli, M. Sampels (Eds.), Hybrid Metaheuristics, in: Lecture Notes in Computer Science, vol. 6373, Springer, 2010, pp. 174–189.
[24] C. Prins, C. Prodhon, R. Wolfler Calvo, Nouveaux algorithmes pour le problème de localisation et routage sous contraintes de capacité, in: A. Dolgui, S. Dauzère-Pérèz (Eds.), MOSIM' 04, vol. 2, 2004, pp. 1115–1122.

[25] C. Prins, C. Prodhon, R. Wolfler Calvo, Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking, 4OR 4 (3) (2006) 221–238.
[26] C. Prins, C. Prodhon, A. Ruiz, P. Soriano, R. Wolfler Calvo, Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic, Transp. Sci. 41 (4) (2007) 470–483.
[27] C. Prodhon, C. Prins, A survey of recent research on location-routing problems, Eur. J. Oper. Res. 238 (1) (2014) 1–17.
[28] S. Salhi, G. Rand, The effect of ignoring routes when locating depots, Eur. J. Oper. Res. 39 (2) (1989) 150–156.
[29] M. Schneider, M. Drexl, A survey of the standard location-routing problem, Ann. Oper. Res. 259 (1) (2017) 389–414.
[30] M. Schneider, M. Löffler, Large composite neighborhoods for the capacitated location-routing problem, Transp. Sci. 53 (1) (2019) 301–318.
[31] C.-J. Ting, C.-H. Chen, A multiple ant colony optimization algorithm for the capacitated location routing problem, Int. J. Prod. Econ. 141 (1) (2013) 34–44.
[32] D. Tuzun, L.I. Burke, A two-phase tabu search approach to the location routing problem, Eur. J. Oper. Res. 116 (1) (1999) 87–99.
[33] V. Yu, S.-W. Lin, W. Lee, C.-J. Ting, A simulated annealing heuristic for the capacitated location routing problem, Comput. Ind. Eng. 58 (2) (2010) 288–299.