J Heuristics (2020) 26:423–452
https://doi.org/10.1007/s10732-017-9363-8

A general variable neighborhood search for solving the multi-objective open vehicle routing problem

Jesús Sánchez-Oro[1] · **Ana D. López-Sánchez**[2] · **J. Manuel Colmenar**[1]

Received: 12 April 2017 / Revised: 25 September 2017 / Accepted: 15 December 2017 /
Published online: 23 December 2017

Abstract The multi-objective open vehicle routing problem (MO-OVRP) is a variant of the classic vehicle routing problem in which routes are not required to return to the depot after completing their service and where more than one objective is optimized. This work is intended to solve a more realistic and general version of the problem by considering three different objective functions. MO-OVRP seeks solutions that minimize the total number of routes, the total travel cost, and the longest route. For this purpose, we present a general variable neighborhood search algorithm to approximate the efficient set. The performance of the proposal is supported by an extensive computational experimentation which includes the comparison with the well-known multi-objective genetic algorithm NSGA-II.

Keywords General variable neighborhood search · NSGA-II · Open vehicle routing problem · Sweep algorithm · Local search · Multi-objective optimization

1 Introduction

The distribution of goods and services is a basic logistic operation for most of business activities, constituting a significant part of the overall costs of a company. There-

✉ Jesús Sánchez-Oro
jesus.sanchezoro@urjc.es

Ana D. López-Sánchez
adlopsan@upo.es

J. Manuel Colmenar
josemanuel.colmenar@urjc.es

1 Rey Juan Carlos University, C/Tulipán, s/n, 28933 Móstoles, Madrid, Spain

2 Pablo de Olavide University, Ctra. Utrera Km 1, 41013 Sevilla, Spain

fore, minimizing the cost related to transportation becomes a matter of interest for researchers of several fields.

Most of the transportation problems found in the literature have been solved by minimizing the costs associated with the transportation, that is, minimizing the total cost, following different approaches: reducing the purchase of vehicles, the wages of the drivers, the diary distance travel by all the vehicles, the economic penalty when a constraint is violated (as overtime payments), etc. Traditionally, the problem had been focused on minimizing the total travelled distance (Toth and Vigo 2002). However, nowadays there is a great interest in considering other aspects. For instance, balancing the workloads (workers' point of view) or the waiting time of the customers to receive the service (customer satisfaction).

This paper deals with real-world situations with the aim of reducing costs for transportation companies. Most of them need to purchase or hire the vehicles' fleet, to optimize the total cost (distance traveled by all vehicles), and to balance the workloads in order to assure that all routes have a similar duration. In other words, companies try to balance customer satisfaction in order to homogenize the maximum time a customer waits to receive the service. Companies are worried about the customers' satisfaction since it has an indirect benefit for the company as well as the workers' satisfaction since it is well-known that they are more productive when they work in better conditions. Therefore, it is interesting to take into account several objectives that are in conflict. That is, to solve routing problems within the multi-objective nature.

The optimization problem under consideration can be modeled as the Open Vehicle Routing Problem (OVRP) which is a variation of the well-known Vehicle Routing Problem (VRP) with the only difference that vehicles are not required to return to the depot after completing their service. Formally, the objective of the OVRP is to find a set of routes traveled by vehicles with the minimum travel cost and such that each customer is served exactly once. It is worth mentioning that each route starts at the depot and ends at one of the customers or vice versa; and the side constraints must be satisfied (for instance, vehicles' capacity or maximum duration, if any). Here, we solve a multi-objective problem instead. We introduce the multi-objective open vehicle routing problem (MO-OVRP) that seeks to minimize the number of vehicles, the total travel cost, and the maximum travel cost among all routes (makespan). The relevance of this problem is supported by many practical applications that fit into the MO-OVRP framework: companies hiring vehicles, pick up and delivery VRP, and planning of train and bus services, among others.

A general variable neighborhood search (GVNS) algorithm is proposed to tackle the MO-OVRP. The initial solution for GVNS, as well as an initial approximation of the Pareto front are obtained using a constructive method called *Sweep*. Then, the approximation of the Pareto front is improved by using a set of neighborhood structures. Contrary to traditional metaheuristics based on local search methods that follow a trajectory, GVNS is a metaheuristic framework based on systematic changes of neighborhoods.

The paper is organized as follows. Section 2 presents the considered problem. Section 3 describes the GVNS algorithm proposed in this paper, and Sect. 4 shows the adaptation of the NSGA-II to solve the MO-OVRP. Computational results are provided in Sect. 5 and finally, Sect. 6 summarizes the paper and discusses future work.

2 Problem description and related work

The Open Vehicle Routing Problem (OVRP) was first described in a paper by Scharge in 1981 Schrage (1981) and can be formally stated as follows. Let $G = (V, E)$ be a complete graph, where $V = \{0, 1, \ldots, n\}$ is the node set and $E = \{(i, j) : i, j \in V, i \neq j\}$ is the edge set. Node 0 is the depot and $N = \{1, \ldots, n\}$ is the customer set (i.e., $N = V \setminus \{0\}$). Each edge $(i, j) \in E$ has an associated cost c_{ij}. This cost could be measured in different units, for instance, the distance from i to j or the time that takes to get from i to j or even the economic cost to service customer i and then j. Furthermore, each customer $i \in V$ has a demand $q_i > 0$ (with $q_0 = 0$). Let $M = \{1, \ldots, m\}$ be the fleet of m identical vehicles located at the depot. Note that this value is traditionally known beforehand. In our case, it is important to highlight that the number of vehicles is included as an objective function as it will be explained later. Each vehicle could have some constraints, as an associated fixed cost F, a capacity Q, or a maximum cost limit C, among others. The customers must be served by at most m Hamiltonian paths, each one associated with one vehicle, starting at the depot and ending at one of the customers, or vice versa. Each vehicle's route must satisfy the imposed constrains, for instance, they cannot exceed the maximum allowed capacity. The classical OVRP is focused on firstly minimizing the number of used vehicles and then minimizing the total cost. Figure 1 shows an example of instance for this kind of problems.

There is another variant of the OVRP named Balanced Open Vehicle Routing Problem (BOVRP), originally proposed by López Sánchez et al. (2014), in which the objective is to minimize the number of used vehicles first and then the makespan, defined as the maximum cost of a route in the solution. Figures 2 and 3 show an OVRP and a BOVRP solution, respectively. Figure 2 represents three routes for a feasible OVRP solution where the total cost is minimized. In the example, the solution for OVRP has a total cost of 12 units and the longest route is the green one with a cost of 5 units. Solving a BOVRP we get a more balanced set of routes, as depicted in Fig. 3,

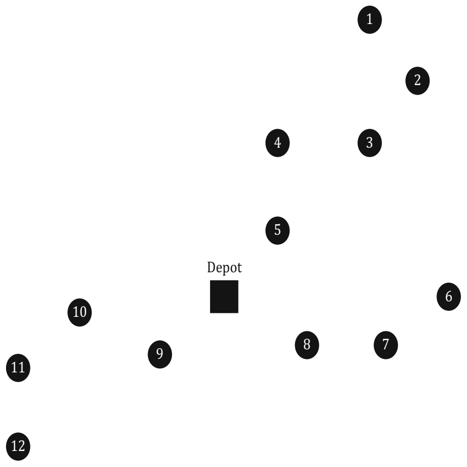


Fig. 1 Original instance

Fig. 2 Solution for f_1

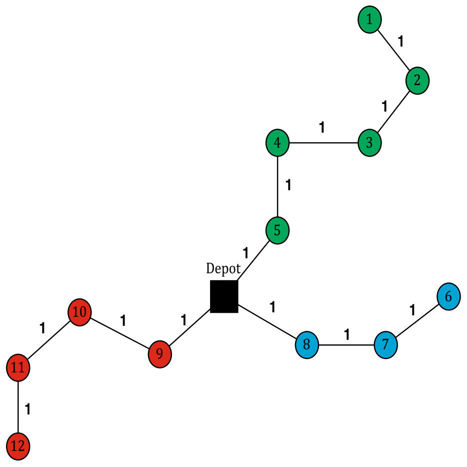
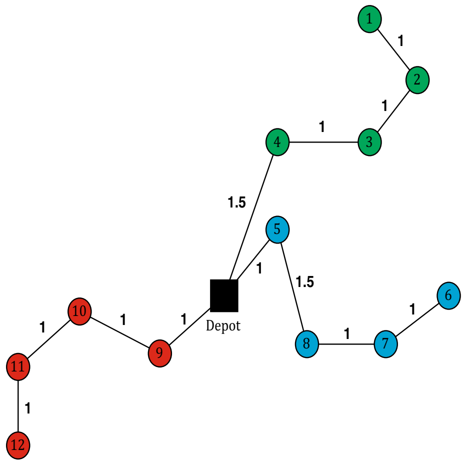


Fig. 3 Solution for f_2



where the total cost has incremented (13 units) but the cost of the longest route is now reduced (4.5 units). Furthermore, Fig. 4 shows another solution where the number of vehicles is minimized. In this example, a solution with two routes is obtained with a total cost of 13.5 units and where the cost of the longest route is incremented considerably (9.5 units). Notice that these figures show different solutions that are in conflict considering the three objective functions.

In this paper, both problems, the OVRP and the BOVRP, will be addressed at the same time. The considered problem turns into the multi-objective open vehicle routing problem (MO-OVRP), in which the objectives are three: minimize the total cost (f_1), minimize the makespan (f_2), and minimize the number of vehicles (f_3). As usual in multi-objective optimization, those objectives are in conflict since there is not a single solution that simultaneously optimizes each objective. This means

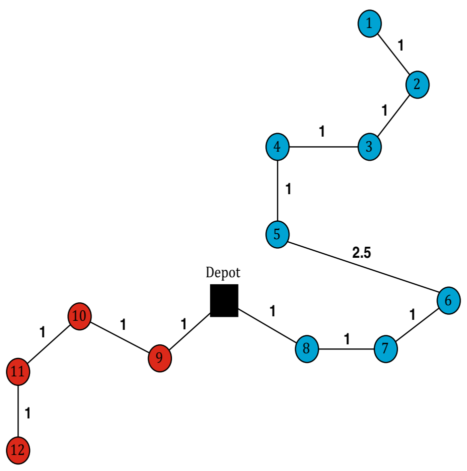


Fig. 4 Solution for f_3

that the value of one objective function cannot be improved without deteriorating the value of at least another objective function. Those solutions are known as efficient solutions, non-dominated solutions or Pareto optimal solutions and henceforth the set of those solutions will be denoted as R. If we focus again in the problem shown in Figs. 2, 3, and 4, the conflict in the solutions is clear since the first solution is $(f_1, f_2, f_3) = (12, 5, 3)$, the second solution is $(f_1, f_2, f_3) = (13, 4.5, 3)$, and the third solution is $(f_1, f_2, f_3) = (13.5, 9.5, 2)$. Each solution obtains a better value for one objective function but a worse solution regarding the other two objective functions.

As far as we know, the MO-OVRP presented in this paper has not been addressed in the literature yet. However, the OVRP has been the target of several researchers and practitioners since many real-world situations fit into the OVRP framework. For instance, companies which do not own a vehicle fleet and must contract services to external transporter (Tarantilis et al. 2005), pick up and delivery VRP (Schrage 1981), and planning of train services or bus routes (Fu et al. 2005). All those situations can be extrapolated to the BOVRP framework even if another objective is optimized. This variant has been mainly ignored in the literature, considered into the planning of bus services (López Sánchez et al. 2014). Finally, it is worth mentioning a contribution of multi-objective OVRP even if we do not consider the same objective functions. Norouzi et al. (2012) presented a MO-OVRP with competitive time windows. The model minimizes the total travel cost of routes and maximizes the obtained sales while balancing the goods distributed among vehicles, that is, the capacity of the vehicles.

3 Variable neigborhood search

Variable Neighborhood Search (VNS) is a metaheuristic framework whose main idea relies on systematic changes of neighborhood structures for finding high quality solutions, without guaranteeing their optimality. Since the original proposal (Mladenović and Hansen 1997), VNS has been in continuous evolution, resulting in different VNS strategies. Specifically, it is worth mentioning Variable Neighborhood Descent (VND), Reduced VNS (RVNS), Basic VNS (BVNS), General VNS (GVNS), Variable Neighborhood Decomposition Search (VNDS), and Variable Formulation Search (VFS), among others. We refer the reader to Hansen and Mladenović (2001) for a complete review of this methodology. VNS has lead to several successful research in recent years (Duarte et al. 2016; Hansen et al. 2017; Sánchez Oro et al. 2014, 2015).

This work is focused on the GVNS variant, which replaces the local search improvement in VNS with a complete Variable Neighborhood Descent (VND) algorithm. In VND, the neighborhood change is performed in a deterministic way, by removing the perturbation method. This methodology is particularly interesting when considering several different neighborhood structures that can be combined, as the ones defined in this paper (see Sect. 3.2). The GVNS method proposed in this work is presented in Algorithm 1.

Algorithm 1 GVNS(S, k_{max})

```
1: $k \leftarrow 1$
2: while $k \leq k_{max}$ do
3:     $S' \leftarrow Shake(S, k)$
4:     $S'' \leftarrow VND(S')$
5:     $k \leftarrow NeighborhoodChange(S, S'', k)$
6: end while
```

Given an initial solution S and the maximum neighborhood to be explored k_{max}, the algorithm starts from the first considered neighborhood (line 1), iterating until reaching the maximum neighborhood (line 2–6). For each iteration, GVNS randomly perturbs the incumbent solution to generate a new one in the neighborhood under evaluation (line 3), using the shake procedure later described in Sect. 3.3. Then, the perturbed solution S' is improved using the VND method described in Sect. 3.2 until reaching a local optimum S'' (line 4). The neighborhood change method (line 5) is responsible for deciding whether to restart the search from the first neighborhood ($k = 1$) if an improvement has been found or to continue with the next neighborhood otherwise ($k = k + 1$). The method ends when the maximum predefined neighborhood has been reached or a maximum CPU time of 1000 s is achieved.

Single-objective GVNS ends returning the best solution found during the search. However, regarding the multi-objective nature of the problem under consideration, the proposed GVNS maintains the set of all non-dominated solutions, (approximation of the Pareto front) found during the search. Specifically, each solution generated is checked to be included in the Pareto front. Then, if the solution is non-dominated, it is inserted, removing all those solutions already included in the front that are dominated by the new one. Otherwise, the solution is discarded, continuing the search. Finally, the algorithm returns the set of all non-dominated solutions.

3.1 Constructive procedure

In this section we describe the constructive procedure used to generate an initial solution for the GVNS algorithm, also creating an initial approximation of the Pareto front.

The constructive procedure, called *Sweep*, follows a strategy that leverages the structure of the problem, based on open routes. The method is a two-phase constructive algorithm that belongs to the traditional cluster-first, route-second methods used to solve routing problems, originally proposed by Gillett and Miller (1974). In the first phase, the algorithm decomposes the problem by clustering customers according to their location, that is, grouping customers that are geographically close among them in the same route. Then, in the second phase, every route (cluster) is optimized independently by using the nearest neighbor algorithm. The *Sweep* algorithm is applied iteratively as follows:

1. The depot is considered as the center (or origin) of the two-dimensional plane.
2. The polar coordinates of each customer with respect to the depot are computed.
3. All customers are sorted by increasing polar angle.
4. A seed node is chosen in order to start the assignment of customers to vehicles (clusters). The depot is joined with the arbitrarily chosen seed node. Traditional *Sweep* algorithms consider the node with the smallest polar angle as the seed. However, in order to generate diverse solutions, our constructive procedure considers a different seed node in each iteration.
5. A cluster is constructed by sweeping all nodes by increasing polar angle with respect to the selected seed node.
6. A new cluster is created when the current one is complete, considering the maximum vehicle capacity.
7. Repeat steps 5 and 6, until all customers have been included in a cluster.
8. Once all clusters have been included in a vehicle, routes are created by using the nearest neighbor algorithm.

As the number of vehicles is a discrete objective function, and in order to consider this objective specifically in the construction phase since no neighborhood will be specially designed to improve the former objective, some variability will be included in the step 6 of the *Sweep* algorithm. To vary the number of vehicles used during the execution of the algorithm, we limit the capacity of each vehicle to a percentage of its maximum. Therefore, the algorithm is executed considering that the capacity is βC where $\beta \in [0, 1]$ but satisfying that $\beta C \geq \max_{i \in N} d_i$ to avoid the construction of infeasible solutions. For each β value, randomly chosen in the interval [0, 1], a new set of non-dominated solutions will be constructed.

As we previously mentioned, during the construction phase, an approximate set of efficient solutions, $\hat{E}$, will be obtained. Every time a new feasible solution is built, we need to check whether it should be included in the approximate set of efficient solutions or not. Specifically, if the current solution S is a non-dominated one, it is included in the set, removing those that are dominated by S. Otherwise, solution S is discarded.

3.2 Neighborhood structures

The VND algorithm used as improvement method inside the proposed GVNS requires from a set of neighborhoods to be deterministically explored. In this section we describe different neighborhood structures, each one designed to find better solutions with respect to a different objective function. Specifically, we propose five different neighborhood structures for each objective function: those devoted to minimize the total cost are denoted by N_m, while the ones designed to minimize the maximum route cost are denoted by N_m^*, with $1 \leq m \leq 5$.

It is worth mentioning that the objective of minimizing the number of vehicles is not considered in any neighborhood structure, since it is used in the construction phase of the search, as described in Sect. 3.1.

We define the following five neighborhood structures:

- N_1 and N_1^*: Insertion of a node in a different position of the same route.
- N_2 and N_2^*: Swap of two consecutive nodes in the same route.
- N_3 and N_3^*: Exchange of two non-consecutive nodes in the same route.
- N_4 and N_4^*: Insertion of a node in the best position of a different route.
- N_5 and N_5^*: Exchange of two nodes of different routes, inserting them in the best position of the destination route.

Using these neighborhood definitions, we propose two local search methods for each considered neighborhood, which differ in the objective function under evaluation. On the one hand, those methods considering the minimization of the total cost (exploring neighborhoods $N_1, N_2, \ldots, N_5$) perform the corresponding move of the neighborhood for each available route, stopping when no improvement is found in any of the routes. On the other hand, those methods considering the minimization of the maximum cost (exploring neighborhoods $N_1^*, N_2^*, \ldots, N_5^*$) perform the corresponding move of the neighborhood considering only the route with the maximum cost, stopping when this maximum cost cannot be improved. It is worth mentioning that, in the case of N_4^* the node to be inserted in a new route is selected from the route with maximum cost. Similarly, the neighborhood N_5^* exchanges one node from the route with maximum cost with one node of any other route while this objective function is minimized.

Therefore, the neighborhoods explored by local search methods for the minimization of total cost are considerably larger than the ones devoted to minimize the maximum cost. Regarding the VND methodology, it is recommended to explore the neighborhoods from the smallest and fastest one to be evaluated to the largest and slowest one (Mladenović and Hansen 1997). Following this idea, the proposed VND firstly considers the local search procedures devoted to reduce the maximum cost (N_1^* to N_5^*), and then the ones intended to reduce the total cost (N_1 to N_5).

Algorithm 2 shows the pseudocode of the VND algorithm.

Algorithm 2 VND(S, k_{max})

```
1: $N \leftarrow \{N_1^*, N_2^*, \ldots, N_5^*, N_1, N_2, \ldots, N_5\}$
2: $k \leftarrow 1$
3: while $k \leq k_{max}$ do
4:     $S^\star \leftarrow \underset{S' \in N_k(S)}{\arg\min} f(S')$
5:     $k \leftarrow NeighborhoodChange(S, S^\star, k)$
6: end while
```

Starting from an initial solution S, VND iterates until reaching the maximum predefined neighborhood k_{max} (in our case, $k_{max} = 10$ neighborhoods), finding in each iteration the best solution in the neighborhood under evaluation (line 4). Then, the neighborhood change procedure decides whether to evaluate the next neighborhood ($k = k + 1$) if no improvement has been found, or to restart from the first considered neighborhood ($k = 1$) otherwise.

Notice that given the multi-objective nature of the problem considered in this work, each neighborhood is devoted to improve a different objective function. Therefore, the evaluation in line 4 of the algorithm refers to the objective function for which the corresponding local search is designed. This strategy leads the VND algorithm to focus on a different objective depending on the neighborhood under evaluation.

3.3 Shake

The shake method is used in the VNS methodology as a perturbation method with the aim of increasing the diversity of the search and escape from local optima. We define a shake method for the problem under consideration that randomly selects a source and destination route. Then, the method selects a node from each route at random and interchange their positions. Notice that source and destination route can eventually be the same route and, therefore, this method considers movements in the same route and in different ones. The maximum perturbation size is given by parameter n_{max} which indicates the percentage of nodes that will be involved in the perturbation. On the other hand, k_{step} represents the increment in the perturbation size for each iteration.

3.4 Final algorithm

This subsection is devoted to summarize the complete algorithm described in the previous subsections. A pseudocode is included containing the construction phase and the GVNS method where all the details are given such as the non-dominated set updating or the stopping conditions, among others. See Algorithm 3 for deeper details.

The algorithm receives the graph G, the parameter β to vary the number of vehicles, the number of iterations it, and the maximum number of neighborhoods to be explored k_{max} as input. The algorithm starts by creating an empty non-dominated set of solutions and then it is executed during it iterations (lines 1 and 2). In each iteration a solution is constructed (line 3) using the procedure explained in Sect. 3.1. Notice that every solution is checked for its possible inclusion in the set of non-dominated solutions, R. The function $Insert\&Update(S)$ checks whether the solution S is a non-dominated solution and, if so, the solutions of the set that are dominated by S are removed (lines 4, 9, and 11). Once a solution is built, the GVNS is applied until the maximum predefined neighborhood is reached (lines 5 to 17). For each iteration of the GVNS, the shake method is performed and the VND method is applied (see Sects. 3.2 and 3.3). Here it is extremely important to check if the size of the set of non-dominated solutions has changed and in such a case we restart the search from the first neighborhood. Otherwise, the algorithm continues the search in the next neighborhood. The algorithm finishes returning the set of non-dominated solutions (line 19).

Algorithm 3 MO-OVRP-GVNS (G, β, it, k_{max})

```
1: $R \leftarrow \emptyset$
2: for $i = 1, \ldots, it$ do
3:     $S \leftarrow construct(G, \beta)$
4:     $R \leftarrow Insert\&Update(S)$
5:     $k \leftarrow 1$
6:     while $k \leq k_{max}$ do
7:         $R' \leftarrow R$
8:         $S' \leftarrow shake(S, k)$
9:         $R \leftarrow Insert\&Update(S')$
10:        $S'' \leftarrow VND(S')$
11:        $R \leftarrow Insert\&Update(S'')$
12:        if $|R| \neq |R'|$ then
13:            $k \leftarrow 1$
14:        else
15:            $k \leftarrow k + 1$
16:        end if
17:    end while
18: end for
19: return $R$
```

4 NSGA-II adaptation

To prove the quality of the proposed algorithm, NSGA-II is adapted to our problem. NSGA-II is a multiobjective evolutionary algorithm proposed as an improvement version of the NSGA (Non-dominated Sorting Genetic Algorithm) to outperform it (Deb et al. 2002).

The NSGA-II starts by generating an initial population of solutions in the form of chromosomes, which are formed by either binary or numeric values that are called genes. Each chromosome, which is an individual from the population, represents a solution to the problem at hand. The initial population is evolved according to the features of NSGA-II, and the non-dominated solutions are kept and included in the set of non-dominated solutions, using a separated file.

Regarding the adaptation of NSGA-II to our problem, the initial population is randomly generated. Given that this algorithm is based on population, this random method provides diversity in the solutions and allows a wider cover of the objective space. However, we have implemented a method to repair the random solutions in order to fit the capacity constraints. Therefore, all the solutions of the initial population are feasible according to the capacity constraints.

We have used an integer codification for the chromosomes similar to the one described in We et al. (2013). Each chromosome is divided into two parts separated by the number zero, as shown in Fig. 5. In the first part of the chromosome, each gene represents one customer. The second part of the chromosome contains the information of the vehicles. Each gene in the second part indicates the number of customers served by each vehicle. Of course, the sum of the numbers in the second part must be equal to the number of customers. For example, Fig. 5 shows a representation of a feasible solution with 10 customers and 3 vehicles where vehicle 1 serves 3 customers which

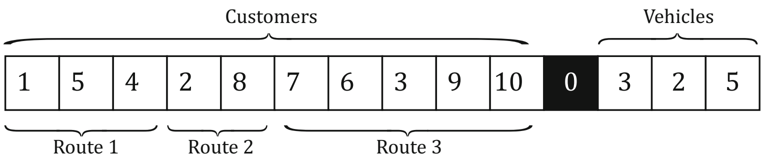


Fig. 5 Chromosome representation

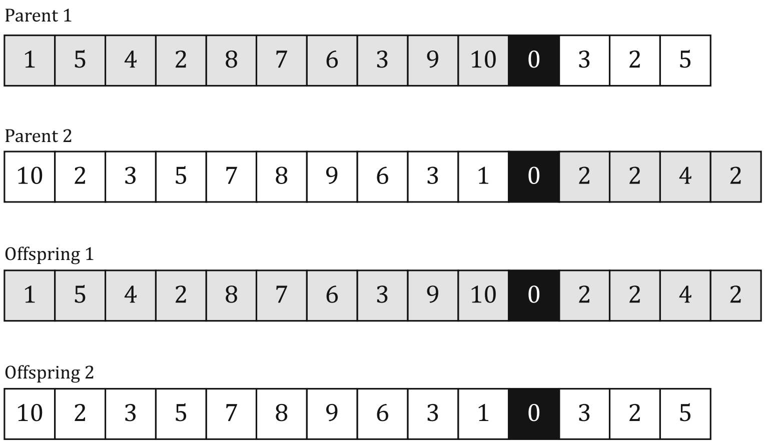


Fig. 6 Crossover operator

are 1, 5, and 4; vehicle 2 serves 2 customers which are 2, and 8; and finally, vehicle 3 serves 5 customers which are 7, 6, 3, 9, and 10 (maintaining the order).

The previously mentioned repair operator travels both parts of the chromosome: the customers and the vehicles parts. Hence, for each vehicle, it is verified that its capacity allows to serve the customers that correspond to its route. If this constraint is not satisfied, the route for this vehicle is minored until the capacity is satisfied. Those customers removed from the current vehicle are transferred to the subsequent vehicle repeating this process until the last vehicle. Notice that this strategy requires a linear cost because the next vehicle is identified by the subsequent gene in the vehicle part. Once all the vehicles are traversed, it may happen that some remaining customers do not belong to any vehicle. In that case, new vehicles are added to the chromosome until all the customers belong to one route.

The crossover operation is based on the well-known one-point crossover (Eiben and Smith 2003). However, in order to maintain the feasibility of the combined solutions, the point where the crossover is performed is always the same, which is the separator (number zero) between the two parts that we have previously defined. Figure 6 illustrates the process of one-point crossover. Offspring 1 is generated by selecting the first part of parent 1 and the second one of parent 2, while offspring 2 results from the combination of the second part of parent 1 and the first one of parent 2.

Table 1 Best genetic parameters found in a preliminary experimentation for the NSGA-II

Parameter	Value
Population size	200
Generations	5000
Crossover probability	0.6
Mutation probability	0.3

The mutation operator is different than the standard case of genetic algorithms. The aim of the operator is to disturb a solution and provide diversity changing the genes of a chromosome according to a probability. However, taking into account again the feasibility of the solutions, we decided to implement a special case of mutation. Our proposal tries to imitate the movements between neighbors of solutions, as described in Sect. 3.2. Therefore, if the mutation has to be performed on a chromosome, two equally probable operations may occur: customer part mutation or vehicle part mutation.

In the customer part mutation, two randomly selected genes are exchanged. This way, it could lead to the modification of the order in one route or the exchanging of customers among different routes. However, this mutation does not change the length of any of the routes.

In the vehicle part mutation, two genes are selected by random. One of the genes is reduced in one unit and the other one is added one unit. This process modifies the length two routes in one unit. In addition, we also allow the addition of one vehicle. To produce this behavior, we include in the random selection a final gene whose value is 0, representing the possible new vehicle. The only constraint that we have implemented is to avoid those vehicles with one customer to be minored. In order to guarantee the feasibility of the solutions, the repair operator is applied after both crossover and mutation operators.

Finally, a binary tournament operator (Eiben and Smith 2003) was applied for the selection of the offspring, taking into account the dominance.

After some preliminary experiments where several values of population size and number of generations were tested, we selected the values of the parameters that are shown in Table 1 because they obtain the best compromise between execution time and convergence of the solutions. For further details see Tables 2, 3 and 4 where a subset of instances has been selected and solved in order to set the parameters.

For each instance, the NSGA-II algorithm was executed 30 times to avoid the random bias. The Pareto fronts obtained for each instance correspond to the non-dominated solutions considering the 30 runs.

5 Computational results

This section is divided into two parts. In the first one a set of indicators or metrics are described in order to assess the quality of the algorithms. In the second part a set of instances is solved implementing the previous indicators in order to decide the best algorithm.

Table 2 Results of the coverage metric for the NSGA-II parameter tuning

	$Cr = 0.6$ $Mut = 0.1$	$Cr = 0.6$ $Mut = 0.3$	$Cr = 0.7$ $Mut = 0.1$	$Cr = 0.7$ $Mut = 0.3$	$Cr = 0.8$ $Mut = 0.1$	$Cr = 0.8$ $Mut = 0.3$
A-n33-k5	1.0000	**0.5918**	1.0000	0.7600	0.9811	0.6667
A-n37-k5	1.0000	0.6207	0.8710	0.6000	0.9394	**0.5806**
A-n45-k7	1.0000	**0.1026**	1.0000	0.8246	1.0000	0.8235
A-n55-k9	1.0000	**0.3871**	0.9565	0.6923	1.0000	0.7308
A-n61-k9	1.0000	**0.1515**	1.0000	0.8667	1.0000	0.7027
A-n65-k9	0.8699	0.9212	0.8155	0.9064	**0.8099**	0.8887
B-n34-k5	1.0000	**0.5938**	1.0000	0.6552	0.9565	0.9565
B-n38-k6	1.0000	0.8750	0.8462	**0.5294**	1.0000	0.7500
B-n45-k6	1.0000	0.6452	1.0000	**0.4783**	1.0000	0.5000
B-n56-k7	1.0000	0.7059	1.0000	0.7027	1.0000	**0.6538**
B-n64-k9	1.0000	0.6522	1.0000	**0.4375**	1.0000	0.5833
B-n67-k10	1.0000	1.0000	1.0000	**0.0000**	0.9565	0.9737
E-n30-k3	0.9630	**0.2750**	0.9583	0.6667	0.9189	0.8491
E-n76-k8	1.0000	**0.1207**	1.0000	0.9615	1.0000	0.8438
E-n101-k8	1.0000	**0.0000**	1.0000	1.0000	1.0000	0.9324
F-n45-k4	0.8500	0.9143	1.0000	0.7727	1.0000	**0.2857**
M-n121-k7	0.9000	**0.0192**	1.0000	0.9298	1.0000	0.5965
M-n151-k12	1.0000	1.0000	1.0000	0.7600	1.0000	**0.2083**
M-n200-k17	1.0000	1.0000	1.0000	0.7308	1.0000	**0.2308**
P-n20-k2	0.7647	0.9630	0.9091	0.8000	0.8824	**0.6000**
P-n22-k8	0.9474	**0.2424**	0.9615	0.7273	1.0000	0.6522
P-n45-k5	1.0000	0.8378	1.0000	**0.4324**	0.9767	0.5000
P-n50-k10	1.0000	0.8571	1.0000	**0.2000**	1.0000	0.7027
P-n55-k15	1.0000	0.8519	1.0000	**0.4545**	1.0000	0.8750
P-n60-k15	1.0000	**0.3889**	1.0000	0.7188	1.0000	0.8571
P-n76-k5	1.0000	0.8103	1.0000	0.6897	1.0000	**0.5000**
Mean	0.9729	**0.5972**	0.9738	0.6653	0.9777	0.6709
# best	0	**11**	0	7	1	7

Smaller is better

5.1 Metrics for multi-objective problems

To measure the quality of the proposed algorithms some indicators must be used. In multi-objective optimization problems, to evaluate different algorithms it is necessary to compare the whole sets of solutions obtained by each algorithm. The performance can be measured in different ways, for instance measuring the proximity of the generated solutions to the true Pareto front or measuring the diversity of the solutions.

In those problems where the true Pareto front is known it is usually selected as the reference set to compare the results of the algorithms under evaluation. However, this real Pareto front is not known for the problem under consideration so a reference set

Table 3 Results of the hypervolume metric for the NSGA-II parameter tuning

	$Cr = 0.6$ $Mut = 0.1$	$Cr = 0.6$ $Mut = 0.3$	$Cr = 0.7$ $Mut = 0.1$	$Cr = 0.7$ $Mut = 0.3$	$Cr = 0.8$ $Mut = 0.1$	$Cr = 0.8$ $Mut = 0.3$
A-n33-k5	0.7911	0.8781	0.7880	0.8631	0.7944	**0.8823**
A-n37-k5	0.7641	0.8468	0.7891	0.8473	0.8116	**0.8854**
A-n45-k7	0.7200	**0.8697**	0.7747	0.8637	0.7619	0.8433
A-n55-k9	0.7930	0.8701	0.6805	**0.8969**	0.7403	0.8476
A-n61-k9	0.7930	0.8701	0.6805	**0.8969**	0.7403	0.8476
A-n65-k9	0.9655	0.5000	**1.0000**	0.7073	**1.0000**	0.9667
B-n34-k5	0.8484	0.8956	0.8245	**0.9119**	0.8229	0.8937
B-n38-k6	0.8055	0.8971	0.8294	**0.9192**	0.7686	0.8885
B-n45-k6	0.9087	**0.9377**	0.9034	0.9201	0.8939	0.9207
B-n56-k7	0.7529	**0.9635**	0.8670	0.9359	0.7721	0.9272
B-n64-k9	0.9088	**0.9408**	0.8597	0.9204	0.8255	0.9243
B-n67-k10	0.7031	0.8403	0.7231	**0.9468**	0.7308	0.8637
E-n30-k3	0.8220	0.8697	0.8410	**0.8701**	0.8507	0.8329
E-n76-k8	0.7835	**0.9369**	0.7539	0.8922	0.7865	0.8939
E-n101-k8	0.6428	**0.8865**	0.5198	0.7800	0.5495	0.7313
F-n45-k4	0.8183	0.8774	0.7152	0.8619	0.7578	**0.9458**
M-n121-k7	0.8752	**0.9438**	0.7964	0.8901	0.7936	0.9094
M-n151-k12	0.4661	0.7794	0.6079	0.8826	0.5835	**0.9650**
M-n200-k17	0.4104	0.8539	0.5675	0.8521	0.5631	**0.9292**
P-n20-k2	0.8089	0.8023	0.7955	**0.8135**	0.7916	0.8131
P-n22-k8	0.7856	**0.8810**	0.8024	0.8754	0.7687	0.8468
P-n45-k5	0.6718	0.8081	0.6954	**0.8343**	0.7852	0.8403
P-n50-k10	0.7987	0.8948	0.8262	**0.9151**	0.8217	**0.9151**
P-n55-k15	0.8679	**0.9573**	0.8762	0.9512	0.9004	0.9346
P-n60-k15	0.8279	**0.9344**	0.8506	0.9239	0.8628	0.9311
P-n76-k5	0.7189	0.8064	0.6846	0.8339	0.6552	**0.8590**
Mean	0.7712	0.8670	0.7712	0.8772	0.7743	**0.8861**
# best	0	**10**	1	8	1	8

Larger is better

is constructed with all the non-dominated solutions of the set resulting from merging the solutions found by all the algorithms being compared. This set, named R, acts as the best known Pareto front for the problem. Then, the following metrics will be computed:

- **CPU time** This metric computes the wall-clock time spent by the computer executing one algorithm solving the given problem. Obviously, a shorter computational time is preferable.
- **Number of efficient points** This metric counts the number of efficient points found by each algorithm. The decision-maker usually prefers the maximum quantity of

Table 4 Results of the epsilon metric for the NSGA-II parameter tuning

	$Cr = 0.6$ $Mut = 0.1$	$Cr = 0.6$ $Mut = 0.3$	$Cr = 0.7$ $Mut = 0.1$	$Cr = 0.7$ $Mut = 0.3$	$Cr = 0.8$ $Mut = 0.1$	$Cr = 0.8$ $Mut = 0.3$
A-n33-k5	0.1192	0.0513	0.1409	0.0916	0.0956	**0.0344**
A-n37-k5	0.1671	0.0744	0.1737	0.0863	0.0792	**0.0541**
A-n45-k7	0.1693	0.0525	0.1242	**0.0495**	0.1377	0.0881
A-n55-k9	0.1430	0.1184	0.1996	**0.0296**	0.2964	0.1350
A-n61-k9	0.2839	**0.0358**	0.3218	0.1245	0.3357	0.1257
A-n65-k9	0.0769	**0.0110**	0.0919	0.0423	0.1574	0.0506
B-n34-k5	0.1256	0.1196	0.1396	**0.0639**	0.1315	0.0927
B-n38-k6	0.1463	**0.0645**	0.1823	0.0811	0.2370	0.1249
B-n45-k6	0.0423	0.0192	0.0532	0.0261	0.0590	**0.0065**
B-n56-k7	0.2446	**0.0396**	0.1286	0.0444	0.2795	0.0568
B-n64-k9	0.0282	0.0234	0.1310	**0.0177**	0.1372	0.0256
B-n67-k10	0.3084	0.1883	0.2107	**0.0177**	0.2926	0.0760
E-n30-k3	0.2899	**0.0000**	0.3642	0.1548	0.3723	0.1979
E-n76-k8	0.0854	0.0539	0.0661	**0.0358**	0.0611	0.1034
E-n101-k8	0.1681	**0.0000**	0.2890	0.0666	0.1765	0.0684
F-n45-k4	0.0614	0.1100	0.2168	0.1100	0.2116	**0.0471**
M-n121-k7	0.0929	**0.0146**	0.1571	0.0626	0.1904	0.0688
M-n151-k12	0.4895	0.1972	0.3930	0.1623	0.3419	**0.0000**
M-n200-k17	0.5393	0.1469	0.4442	0.2066	0.4296	**0.0000**
P-n20-k2	0.0710	**0.0440**	0.0789	0.0456	0.0805	0.0554
P-n22-k8	0.0771	0.0213	0.0485	0.0522	0.0699	**0.0212**
P-n45-k5	0.2076	0.1127	0.2013	**0.0547**	0.0960	0.0868
P-n50-k10	0.1693	0.0586	0.1421	**0.0363**	0.1280	0.0435
P-n55-k15	0.0857	0.0203	0.0789	**0.0175**	0.0600	0.0227
P-n60-k15	0.1601	**0.0122**	0.1152	0.0513	0.1113	0.0301
P-n76-k5	0.2483	0.1711	0.2238	0.1019	0.3345	**0.0581**
Mean	0.1769	0.0677	0.1814	0.0705	0.1886	**0.0644**
# best	0	8	0	**9**	0	**9**

Smaller is better

efficient points. Nevertheless, this metric ignores the quality of the points and it could happen that an approximation set dominates the other one.

- **Coverage** The coverage metric, $\mathcal{C}(A, B)$, was proposed by Zitzler in 1999 Zitzler (1999). It calculates the proportion of solutions in the estimated efficient frontier B, which are weakly dominated[1] by the efficient solutions in the estimated frontier A:

[1] $a \preceq b$ means that the solution a weakly dominates the solution b, i.e., a is not worse than b in any of the objectives.

$$\mathcal{C}(A, B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|}.$$

The metric value $\mathcal{C}(A, B) = 1$ means all the solutions in B are weakly dominated by A and $\mathcal{C}(A, B) = 0$ means that no solution of B is weakly dominated by A. Note that $\mathcal{C}(A, B)$ is not necessarily equal to $1 - \mathcal{C}(B, A)$.

- **Hypervolume Indicator** This metric, first introduced by Zitzler and Thiele in 1998 Zitzler and Thiele (1998), calculates the size of the space covered, that is, it computes the hypervolume of the portion of the objective space that is weakly dominated by the estimated efficient frontier A. In order to measure the hypervolume, the objective space must be bounded by fixing a reference set R. The larger the value, the better the quality.
- **Unary Epsilon Indicator** This metric, first introduced by Zitzler et al. in 2003 Zitzler et al. (2003), makes direct use of Pareto dominance, and hence is highly intuitive. $I_\epsilon(A)$, calculates the minimum factor ϵ by which each point of R must be multiplied or summed such that the resulting transformed approximation set is weakly dominated by the estimated efficient frontier A. The smaller the value, the better the quality. In this paper, we will use the unary multiplicative version of the epsilon indicator.

Note that each indicator is based on different preference information. Therefore, using all the indicators will provide more information than simply using just one indicator.

5.2 Computational experiments

Considering the metrics proposed above, the performance of the proposed algorithm is shown on a set of OVRP instances. The set of instances are available in http://econ.au.dk/lys, named as A, B, E, F, M, and P benchmarks. A total of 92 problems were solved on an Intel Core i7 920 (2.67 GHz) and 8 GB RAM, and the algorithms have been implemented using Java 8.

We have divided our results into two parts: preliminary and final experiments. The former are devoted to select the best parameter setting for the NSGA-II and the GVNS, while the latter show the performance of all the considered instances for the best parameterization of the NSGA-II and the GVNS.

5.2.1 Preliminary experiments

Throughout this section, where the preliminary results are shown, a representative subset of 26 instances with different characteristics is studied. Furthermore, to choose the best parameter setting three metrics will be considered; the coverage metric, the hypervolume and the epsilon indicator. In this section, the CPU time and the number of efficient points have not been taken into account.

First of all, computational results for the NSGA-II are shown in order to set the parameters (crossover and mutation probabilities). Specifically, Table 2 shows the coverage metric, Table 3 measures the hypervolume and Table 4 computes the epsilon indicator for the subset of 26 instances. In those three tables, the first column shows the name of the instance and the remaining columns (2–7) present the results for the six combinations of the three crossover probabilities (0.6, 0.7 and 0.8) and the two mutation probabilities (0.1 and 0.3). The three last rows of all the tables compute the mean and the number of times that the algorithm obtains the best indicator value. The bold values in all the tables correspond to the best value of each row.

As can be seen from Table 2, on average, the best parameter combination is a crossover probability of 0.6 and a mutation probability of 0.3. Furthermore, in 11 out of 26 instances this combination of parameters is the best by considering the coverage metric. If we check the hypervolume, see Table 3, where there are no significant differences on average considering a mutation probability of 0.3 for any crossover probability. However, when the crossover probability is 0.6, in 10 out of 26 problems the size of the space covered is better than using other probability. Similar results are obtained in Table 4, there are no significant differences on average considering a mutation probability of 0.3 for any crossover probability. Then, to solve the complete set of instances we have considered a crossover probability of 0.6 and a mutation probability of 0.3.

To design the GVNS, the first step is to compute which one is the contribution of the construction and the neighborhoods as well as to decide the order to travel through the neighborhoods. Although the neighborhoods are usually explored from the smallest and fastest to evaluate to the largest and slowest one, we have performed an additional experiment to select the best neighborhood order. In this way, we have defined six different sequences that consider the neighborhoods in different order. These sequences, VND1 to VND6, are the following:

– VND1 includes all the neighborhoods to optimize the total cost and then all the neighborhoods to optimize the makespan: N_1, N_2, N_3, N_4 N_5, N_1^*, N_2^*, N_3^*, N_4^* N_5^*
– VND2 includes all the neighborhoods to optimize the makespan and then all the neighborhoods to optimize the total cost: N_1^*, N_2^*, N_3^*, N_4^* N_5^*, N_1, N_2, N_3, N_4 N_5
– VND3 considers similar neighborhoods for each objective minimizing first the total cost and second the makespan: N_1,N_1^*, N_2, N_2^*, N_3, N_3^*, N_4, N_4^*, N_5, N_5^*
– VND4 considers similar neighborhoods for each objective minimizing first the makespan and second the total cost: N_1^*, N_1, N_2^*, N_2, N_3^*, N_3, N_4^*, N_4, N_5^*, N_5
– VND5 considers the intra-route movements and then the inter-route movements to minimize the total cost first and then the makespan: N_1, N_2, N_3, N_1^*, N_2^*, N_3^*, N_4 N_5, N_4^* N_5^*
– VND6 considers the intra-route movements and then the inter-route movements to optimize the makespan first and then the total cost: N_1^*, N_2^*, N_3^*, N_1, N_2, N_3, N_4^* N_5^*, N_4 N_5

Taking into account the different neighborhood sequences, we have run our experiments on the representative subset of 26 instances. Tables 5, 6 and 7 present the coverage metric, the hypervolume and the epsilon indicator, respectively. Column 1 in those tables contains the name of the instance. Column 2 considers the constructive procedure without neighborhoods, and the remaining columns (3–8) orders the neighborhoods in the VND algorithm according to the previously described sequences.

Table 5 Results of the coverage metric for the GVNS neighborhood selection

	Constructive	VND1	VND2	VND3	VND4	VND5	VND6
A-n33-k5	1.0000	**0.4167**	0.7143	0.7692	0.8571	0.9375	0.5385
A-n37-k5	1.0000	**0.5385**	0.6875	0.8125	0.6250	0.7333	0.6250
A-n45-k7	1.0000	0.7143	0.8667	0.7000	**0.5333**	0.5833	0.7500
A-n55-k9	1.0000	0.2500	0.3750	0.5455	**0.1250**	0.2500	0.0000
A-n61-k9	1.0000	0.8125	0.8182	0.6364	**0.6154**	0.7500	0.7692
A-n65-k9	1.0000	0.8421	**0.6818**	0.8182	0.9583	0.9167	0.7143
B-n34-k5	1.0000	**0.2500**	0.8571	1.0000	1.0000	**0.2500**	0.5714
B-n38-k6	0.7500	1.0000	**0.6667**	**0.6667**	0.8000	0.8000	0.6000
B-n45-k6	0.7500	0.8182	**0.3333**	0.7000	0.4167	0.7778	**0.3333**
B-n56-k7	1.0000	0.8750	0.7000	0.7500	0.7000	**0.5000**	0.8750
B-n64-k9	1.0000	**0.6667**	0.7500	1.0000	0.7500	1.0000	0.7500
B-n67-k10	1.0000	1.0000	0.6000	**0.5000**	0.6000	0.6000	1.0000
E-n30-k3	0.8333	0.7500	0.6250	**0.4444**	0.8889	0.7273	0.5000
E-n76-k8	1.0000	0.8696	0.7600	0.8750	**0.6522**	0.8889	0.6800
E-n101-k8	1.0000	0.7895	0.6889	0.9756	**0.6222**	0.8372	0.8491
F-n45-k4	1.0000	0.7143	0.4000	**0.5000**	0.8000	0.5714	0.4000
M-n121-k7	1.0000	1.0000	**0.2308**	0.7778	0.8750	0.8621	0.9032
M-n151-k12	1.0000	0.8621	**0.5806**	0.8214	0.8372	0.7568	0.8500
M-n200-k17	1.0000	0.9583	0.6552	0.8421	**0.5652**	0.8125	0.8649
P-n20-k2	0.7692	0.5000	0.5000	0.4375	**0.3750**	**0.3750**	0.4444
P-n22-k8	0.7500	**0.0000**	**0.0000**	0.2000	**0.0000**	0.2000	0.2000
P-n45-k5	0.9375	0.6471	0.6111	**0.5333**	0.6818	0.6875	0.6500
P-n50-k10	1.0000	0.6667	0.7692	0.5333	0.7692	0.8571	**0.4615**
P-n55-k15	1.0000	0.7500	**0.6667**	0.8182	0.7778	0.8889	0.7000
P-n60-k15	1.0000	0.6250	0.7000	0.7500	0.6000	0.5556	**0.4444**
P-n76-k5	0.9583	0.8205	**0.6944**	0.9167	0.8810	0.8108	0.7714
Mean	0.9519	0.6976	**0.6128**	0.7048	0.6656	0.6896	0.6248
# best	0	5	**8**	3	7	3	6

Smaller is better

If we focus on the coverage metric, shown in Table 5, the VND2 gives performance advantage in mean and in most of the instances (8 out of 26). The hypervolume, displayed in Table 6, gives the advantage on average to VND2 and VND3 over the other variants. Finally, the epsilon indicator, see Table 7, shows that VND2 outperforms the other strategies on average. Then, we can conclude that VND2 is the best variant on average and in most of the instances (9 out of 26). Consequently, the variant that is implemented in the GVNS will be the one that includes all the movements to optimize the makespan first and then all the movements to optimize the total distance. This result was expected since the majority of papers in the literature hold that the best order is to include neighborhoods from the smallest and fastest one to be evaluated, to the largest and slowest one. Besides, the neighborhoods designed to optimize the

Table 6 Results of the hypervolume metric for the GVNS neighborhood selection

	Constructive	VND1	VND2	VND3	VND4	VND5	VND6
A-n33-k5	0.7648	0.8536	0.8725	0.8457	0.8653	0.8598	**0.8753**
A-n37-k5	0.7425	0.8683	0.8714	0.8690	**0.8790**	0.8714	0.8742
A-n45-k7	0.8719	0.9054	0.9178	0.9084	0.9135	0.9060	**0.9198**
A-n55-k9	0.6696	0.7547	0.7547	0.7539	**0.7557**	0.7547	**0.7557**
A-n61-k9	0.6876	0.8011	0.8103	0.8088	**0.8155**	0.8145	0.8096
A-n65-k9	0.7805	0.8890	0.8927	**0.8955**	0.8853	0.8922	0.8908
B-n34-k5	0.5515	0.7713	0.6782	0.6861	0.6473	**0.7717**	0.6530
B-n38-k6	0.8083	0.8697	0.8743	0.8716	0.8746	**0.8758**	0.8738
B-n45-k6	0.5707	0.7676	**0.8010**	0.7281	0.7964	0.7775	0.7978
B-n56-k7	0.5443	0.7045	0.7060	**0.9176**	0.7044	0.7073	0.7045
B-n64-k9	0.6694	**0.6983**	0.6960	0.6885	0.6960	0.6901	0.6981
B-n67-k10	0.8948	0.9510	0.9519	0.9518	**0.9520**	0.9511	0.9510
E-n30-k3	0.6758	0.6783	0.6782	0.6783	0.6567	0.6753	**0.6786**
E-n76-k8	0.7884	0.8614	0.8733	0.8605	**0.8749**	0.8599	0.8636
E-n101-k8	0.7281	0.8177	0.8354	0.8151	**0.8384**	0.8316	0.8346
F-n45-k4	0.8108	0.8249	0.8245	0.8248	0.8248	0.8229	**0.8250**
M-n121-k7	0.8973	0.9397	**0.9544**	0.9385	0.9457	0.9432	0.9436
M-n151-k12	0.8215	0.8913	**0.8946**	0.8877	0.8901	0.8897	0.8915
M-n200-k17	0.7999	0.9264	0.9314	0.9300	**0.9329**	0.9269	0.9311
P-n20-k2	0.8102	0.8250	**0.8345**	0.8267	0.8235	0.8217	0.8226
P-n22-k8	0.6747	**0.7317**	**0.7317**	0.7132	**0.7317**	0.7132	0.7132
P-n45-k5	0.7721	0.8107	0.8101	**0.8174**	0.8096	0.8120	0.8113
P-n50-k10	0.7922	**0.8698**	0.8677	**0.8698**	0.8677	0.8690	0.8687
P-n55-k15	0.7619	**0.8622**	0.8598	0.8579	0.8616	0.8594	0.8514
P-n60-k15	0.7750	0.8319	0.8406	0.8300	0.8280	0.8314	**0.8544**
P-n76-k5	0.8133	0.8582	0.8722	0.8587	0.8711	0.8562	**0.8736**
Mean	0.7491	0.8294	**0.8321**	**0.8321**	0.8285	0.8302	0.8295
# best	0	4	5	3	**7**	2	**7**

Larger is better

makespan are contained in the neighborhoods designed to optimize the total cost and consequently, they are ordered in such a way.

Once we have decided the order of the neighborhoods, we need to set the parameters of the GVNS algorithm considering again the subset of 26 instances. We have considered a combination of parameters n_{max} that is the proportion of the number of nodes that will be perturbed in the shaking phase, and k_{step}, which is the perturbation size, considered in the shake procedure. In our computational experiments we have implemented $n_{max} = \{0.10, 0.25, 0.50\}$ and $k_{step} = 0.01$.

Tables 8, 9 and 10 show the coverage, the hypervolume and the epsilon indicator, respectively. Column 1 displays the name of the problems and columns 2, 3 and 4

Table 7 Results of the epsilon metric for the GVNS neighborhood selection

	Constructive	VND1	VND2	VND3	VND4	VND5	VND6
A-n33-k5	0.2419	0.1100	0.0387	0.0938	0.0704	0.0771	**0.0183**
A-n37-k5	0.3029	0.0579	0.0579	0.0579	0.0176	0.0611	**0.0197**
A-n45-k7	0.1275	0.0455	0.0271	0.0345	0.0123	0.0455	**0.0077**
A-n55-k9	0.2836	0.0199	0.0199	0.0211	0.0000	0.0199	**0.0000**
A-n61-k9	0.2365	0.0530	0.0619	0.0619	**0.0370**	0.0601	0.0429
A-n65-k9	0.1267	0.0706	0.0311	0.0570	0.0706	**0.0308**	0.0706
B-n34-k5	0.3995	0.0068	0.2261	0.2273	0.2261	**0.0002**	0.2736
B-n38-k6	0.0907	0.0562	**0.0108**	0.0278	0.0148	0.0474	0.0138
B-n45-k6	0.3934	0.0918	**0.0121**	0.1132	0.0467	0.0821	0.0467
B-n56-k7	0.1987	0.0452	0.0213	0.0361	**0.0121**	0.0361	0.0361
B-n64-k9	0.1018	0.0211	0.0140	0.0211	**0.0090**	0.0352	0.0124
B-n67-k10	0.1485	0.0104	**0.0002**	**0.0002**	0.0029	0.0104	0.0104
E-n30-k3	0.1222	0.0517	0.0261	0.0475	**0.0196**	0.0475	0.0284
E-n76-k8	0.0821	0.0263	0.0263	0.0263	0.0821	0.0821	**0.0251**
E-n101-k8	0.1573	0.0282	0.0246	0.0282	**0.0111**	0.0282	0.0784
F-n45-k4	0.0578	0.0007	0.0007	0.0007	0.0007	**0.0000**	0.0007
M-n121-k7	0.0987	0.0599	**0.0123**	0.0693	0.0561	0.0412	0.0333
M-n151-k12	0.1645	0.0204	**0.0110**	0.0448	0.0337	0.0535	0.0337
M-n200-k17	0.2369	0.0409	**0.0222**	0.0296	0.0224	0.0413	0.0265
P-n20-k2	0.0829	0.0541	0.0506	0.0541	**0.0361**	0.0574	0.0506
P-n22-k8	0.2687	**0.0000**	**0.0000**	0.0723	**0.0000**	0.0723	0.0723
P-n45-k5	0.0990	0.0551	0.0464	**0.0290**	0.0424	0.0584	0.0342
P-n50-k10	0.2235	0.0187	0.0267	**0.0111**	0.0267	0.0187	0.0267
P-n55-k15	0.2090	0.0257	**0.0171**	**0.0171**	0.0221	0.0249	0.0605
P-n60-k15	0.3384	0.1175	0.0517	0.1265	0.1175	0.1175	**0.0056**
P-n76-k5	0.1266	0.0492	**0.0158**	0.0536	0.0167	0.0612	0.0172
Mean	0.1892	0.0437	**0.0328**	0.0524	0.0387	0.0465	0.0402
# best	0	1	**9**	4	**9**	3	5

Smaller is better

consider $n_{max} = 0.10$, $n_{max} = 0.25$ and $n_{max} = 0.50$, respectively. In all columns, $k_{step} = 0.01$ (since this value should be small).

Table 8 shows that in most of the cases (13 out of 26 instances) the proportion of solutions coverage by the three parameters is better considering $n_{max} = 0.50$. This is supported by the mean values obtained in this experiment. The hypervolume, see Table 9, also gives the advantage when $n_{max} = 0.50$ regarding the average value and also in 13 out of 26 instances the size of the space covered is greater. However, the epsilon indicator, see Table 10, holds that $n_{max} = 0.25$ is the best approximation set on average but the results for $n_{max} = 0.50$ are quite similar. Therefore, we have selected $n_{max} = 0.50$ to perform the final computational results.

Table 8 Results of the coverage metric for the parameter tuning in GVNS

	$n_{max} = 0.10$	$n_{max} = 0.25$	$n_{max} = 0.50$
A-n33-k5	0.7097	0.7143	**0.4103**
A-n37-k5	0.7333	0.6389	**0.4000**
A-n45-k7	0.5652	0.7586	**0.3667**
A-n55-k9	0.8571	**0.3684**	0.6400
A-n61-k9	0.7692	**0.5333**	0.6842
A-n65-k9	**0.5000**	0.6818	0.6667
B-n34-k5	0.9545	0.5000	**0.1875**
B-n38-k6	0.8000	0.6429	**0.4737**
B-n45-k6	0.8000	0.5000	**0.4074**
B-n56-k7	**0.4000**	0.7647	0.5000
B-n64-k9	**0.2857**	0.8125	0.3571
B-n67-k10	0.7500	**0.3333**	0.6667
E-n30-k3	0.8889	**0.3529**	0.6667
E-n76-k8	0.6042	0.6538	**0.3846**
E-n101-k8	0.7105	**0.5385**	0.5536
F-n45-k4	0.7500	**0.2381**	0.4375
M-n121-k7	0.7255	0.6818	**0.6522**
M-n151-k12	**0.5818**	0.7000	0.7213
M-n200-k17	**0.5641**	0.7429	0.7429
P-n20-k2	0.9091	0.6522	**0.3214**
P-n22-k8	1.0000	0.3333	**0.2000**
P-n45-k5	0.6977	**0.4474**	0.5682
P-n50-k10	0.8824	0.6452	**0.5357**
P-n55-k15	1.0000	**0.1818**	0.3077
P-n60-k15	0.8333	0.8261	**0.3000**
P-n76-k5	**0.5167**	0.6232	0.5238
Mean	0.7227	0.5718	**0.4875**
# best	6	7	**13**

Lower is better

5.2.2 Comparison between NSGA-II and GVNS

According to the results obtained in the previous section, we have chosen the best parameters that will be used in the NSGA-II (crossover probability of 0.6 and mutation probability of 0.3) and the best variant that will be implemented in the considered GVNS (the VND2 that includes all the neighborhoods to optimize the makespan and then all the neighborhoods to optimize the total distance, with $k_{step} = 0.01$ and $n_{max} = 0.50$).

Finally, both algorithms are compared on the complete set of instances under consideration. Table 11 shows the quality of the efficient sets obtained for each algorithm by applying all the indicators introduced in Sect. 5.1. Specifically, column 1 contains the instance name, columns 2 and 3 show the number of efficient points obtained by the

Table 9 Results of the hypervolume metric for the parameter tuning in GVNS

	$n_{max} = 0.10$	$n_{max} = 0.25$	$n_{max} = 0.50$
A-n33-k5	0.9213	0.9218	**0.9270**
A-n37-k5	0.8732	0.8920	**0.9025**
A-n45-k7	**0.9280**	0.9242	0.9275
A-n55-k9	0.7871	**0.8630**	0.8289
A-n61-k9	0.7284	**0.8135**	0.7709
A-n65-k9	0.8623	**0.8696**	0.8566
B-n34-k5	0.6656	**0.8600**	0.8421
B-n38-k6	0.8137	**0.8241**	0.8123
B-n45-k6	0.8673	0.8852	**0.8886**
B-n56-k7	0.9718	**0.9746**	0.9734
B-n64-k9	0.9040	**0.9215**	0.9140
B-n67-k10	0.9291	0.9182	**0.9404**
E-n30-k3	0.7819	**0.8693**	0.8641
E-n76-k8	0.8772	0.8748	**0.8817**
E-n101-k8	0.8497	0.8538	**0.8602**
F-n45-k4	0.8409	0.9041	**0.9074**
M-n121-k7	0.9571	**0.9582**	0.9537
M-n151-k12	0.9146	**0.9148**	0.9100
M-n200-k17	**0.9622**	0.9601	0.9584
P-n20-k2	0.8375	0.8560	**0.8604**
P-n22-k8	0.6998	0.7667	**0.8570**
P-n45-k5	0.8188	**0.8290**	0.8260
P-n50-k10	0.8978	0.9078	**0.9117**
P-n55-k15	0.8979	0.9426	**0.9456**
P-n60-k15	0.9283	0.9358	**0.9589**
P-n76-k5	0.8768	0.8761	**0.8784**
Mean	0.8612	0.8891	**0.8907**
# best	2	11	**13**

Larger is better

GVNS and the NSGA-II, respectively. Columns 4 and 5 show the proportion of points in the estimated efficient frontier of the GVNS and the NSGA-II that are dominated by the estimated efficient frontier of the Reference set. Columns 6 and 7 show the hypervolume of the GVNS and the NSGA-II, respectively. Columns 8 and 9 calculate the epsilon indicator and finally, the last two columns indicate the computational time (in seconds) spent running each algorithm.

Table 11 shows that in most of the cases, the GVNS algorithm obtains, on average, a similar number of efficient points than the NSGA-II. If we focus on the coverage metric, in all cases $C(NSGA-II, GVNS) < C(GVNS, NSGA-II)$ and in most of the cases $C(NSGA-II, GVNS) = 0$ which means that no solution of the estimated efficient frontier of the GVNS is weakly dominated by the estimated efficient frontier of the NSGA-II set. Then, the coverage metric gives performance

Table 10 Results of the epsilon metric for the parameter tuning in GVNS

	$n_{max} = 0.10$	$n_{max} = 0.25$	$n_{max} = 0.50$
A-n33-k5	0.0372	0.0337	**0.0260**
A-n37-k5	0.0666	0.0394	**0.0241**
A-n45-k7	0.0783	0.0682	**0.0259**
A-n55-k9	**0.1200**	0.0132	0.0866
A-n61-k9	0.1537	0.0892	**0.0657**
A-n65-k9	0.0798	**0.0411**	0.0997
B-n34-k5	0.3396	**0.0176**	0.1115
B-n38-k6	0.0798	0.0463	**0.0456**
B-n45-k6	0.0674	**0.0297**	0.0394
B-n56-k7	0.0540	0.0440	**0.0278**
B-n64-k9	**0.0000**	0.0646	0.1069
B-n67-k10	**0.0192**	0.0475	0.0289
E-n30-k3	0.0362	**0.0272**	0.0344
E-n76-k8	**0.0311**	**0.0311**	0.0343
E-n101-k8	0.0242	0.0348	**0.0166**
F-n45-k4	0.0429	**0.0059**	0.0261
M-n121-k7	0.0427	**0.0181**	0.0408
M-n151-k12	**0.0162**	0.0474	0.0474
M-n200-k17	**0.0144**	0.0187	0.0183
P-n20-k2	0.0504	**0.0213**	0.0291
P-n22-k8	0.1245	0.0848	**0.0386**
P-n45-k5	0.0426	**0.0203**	0.0251
P-n50-k10	0.0899	**0.0505**	0.0712
P-n55-k15	0.1004	**0.0192**	0.0282
P-n60-k15	0.0713	0.0508	**0.0343**
P-n76-k5	**0.0147**	0.0188	0.0193
Mean	0.0691	**0.0378**	0.0443
# best	6	**12**	9

Lower is better

advantage to GVNS over NSGA-II. It is important to note that in this case, as we are only comparing two procedures, $C(NSGA-II, GVNS) = C(R, GVNS)$ and $C(GVNS, NSGA-II) = C(R, NSGA-II)$. The hypervolume also gives the advantage to GVNS over NSGA-II except in 4 out of the 92 instances since the space covered is greater in most of the instances. Finally, the epsilon indicator validates the conjecture that GVNS outperforms NSGA-II since GVNS is smaller than NSGA-II in most of the cases except in 4 out of the 92 instances. In terms of computational effort, it is worth mentioning that the CPU time of the GVNS is almost three times faster than the NSGA-II.

Table 11 Comparison between GVNS and NSGA-II

	Number efficient		Coverage		Hypervolume		Epsilon indicator		CPU time	
	GVNS	NSGA-II	C(NSGA-II,GVNS)	C(GVNS,NSGA-II)	GVNS	NSGA-II	GVNS	NSGA-II	GVNS	NSGA-II
A-n32-k5	**39**	25	**0.0000**	0.9600	**0.9193**	0.8801	**0.0000**	0.1021	**27.87**	444.07
A-n33-k5	32	**50**	**0.0000**	0.9821	**0.9023**	0.8182	**0.0084**	0.0987	**27.22**	469.38
A-n33-k6	30	**33**	**0.0000**	0.6176	**0.8869**	0.8234	**0.0299**	0.0553	**13.51**	559.76
A-n34-k5	**45**	**45**	**0.0000**	1.0000	**0.9488**	0.9000	**0.0000**	0.0865	**29.60**	484.30
A-n36-k5	**26**	18	**0.0000**	1.0000	**0.9560**	0.8182	**0.0000**	0.1432	**38.39**	484.53
A-n37-k5	**52**	33	**0.0000**	1.0000	**0.9298**	0.8456	**0.0000**	0.1221	**42.65**	450.25
A-n37-k6	**35**	31	**0.0000**	0.8125	**0.7396**	0.6244	**0.0707**	0.1476	**21.20**	561.33
A-n38-k5	**41**	40	**0.0000**	0.8810	**0.9179**	0.8119	**0.0000**	0.1599	**44.55**	533.60
A-n39-k5	36	**50**	**0.0000**	0.9216	**0.9088**	0.8069	**0.0000**	0.1405	**49.48**	502.43
A-n39-k6	34	**51**	**0.0000**	0.7636	**0.8727**	0.7735	**0.1136**	0.1571	**17.44**	581.69
A-n44-k6	**44**	**44**	**0.0000**	0.8864	**0.9244**	0.7797	**0.0000**	0.2207	**75.14**	564.03
A-n45-k6	32	**36**	**0.0000**	1.0000	**0.9680**	0.7737	**0.0000**	0.2690	**72.74**	569.52
A-n45-k7	**45**	27	**0.0000**	1.0000	**0.9074**	0.7460	**0.0000**	0.1653	**81.41**	575.29
A-n46-k7	36	**47**	**0.0000**	1.0000	**0.9387**	0.7362	**0.0000**	0.2201	**84.34**	561.64
A-n48-k7	38	**40**	**0.0000**	1.0000	**0.9682**	0.8552	**0.0000**	0.1624	**98.01**	564.41
A-n53-k7	31	**49**	**0.0000**	0.9804	**0.8863**	0.5478	**0.0000**	0.4181	**129.72**	679.61
A-n54-k7	**43**	27	**0.0000**	1.0000	**0.9590**	0.7575	**0.0000**	0.2297	**138.92**	651.69
A-n55-k9	18	**44**	**0.0000**	0.9362	**0.9320**	0.6503	**0.0000**	0.3152	**67.65**	693.79
A-n60-k9	21	**28**	**0.0000**	0.7241	**0.9123**	0.6397	**0.0000**	0.4418	**151.71**	718.90
A-n61-k9	**35**	32	**0.0000**	1.0000	**0.9310**	0.5651	**0.0000**	0.3844	**81.73**	725.95
A-n62-k8	**58**	35	**0.0000**	1.0000	**0.9612**	0.7357	**0.0000**	0.2351	**253.57**	685.27
A-n63-k10	26	**34**	**0.0000**	0.8824	**0.9720**	0.9075	**0.0000**	0.2999	**126.26**	718.06

Table 11 continued

	Number efficient		Coverage		Hypervolume		Epsilon indicator		CPU time	
	GVNS	NSGA-II	C(NSGA-II,GVNS)	C(GVNS,NSGA-II)	GVNS	NSGA-II	GVNS	NSGA-II	GVNS	NSGA-II
A-n63-k9	33	**36**	**0.0000**	0.8889	**0.9274**	0.6640	**0.0000**	0.1129	**237.17**	726.09
A-n64-k9	41	**50**	**0.0000**	0.9020	**0.9344**	0.6381	**0.0000**	0.3812	**179.03**	744.09
A-n65-k9	45	**60**	**0.0000**	0.9032	**0.9602**	0.8275	**0.0000**	0.1492	**244.85**	723.05
A-n69-k9	32	**39**	**0.0000**	0.9000	**0.9379**	0.6725	**0.0000**	0.3014	**266.36**	731.94
A-n80-k10	39	**46**	**0.0000**	0.8261	**0.9682**	0.7144	**0.0000**	0.2909	**592.05**	794.91
B-n31-k5	**27**	6	**0.1111**	0.6818	0.7068	**0.7347**	**0.0038**	**0.0038**	**23.33**	556.29
B-n34-k5	11	**25**	**0.0000**	0.9697	**0.9155**	0.8145	**0.0031**	0.1737	**14.61**	605.12
B-n35-k5	**22**	14	**0.0000**	0.7647	**0.8675**	0.8418	**0.0165**	0.0735	**30.90**	514.64
B-n38-k6	**16**	12	**0.0000**	1.0000	**0.9368**	0.7985	**0.0000**	0.2623	**41.31**	534.46
B-n39-k5	39	**45**	**0.0256**	0.9565	**0.9071**	0.8516	**0.0000**	0.0903	**54.04**	480.34
B-n41-k6	**17**	12	**0.2941**	0.5333	**0.9387**	0.9260	**0.0235**	0.0459	**52.07**	577.06
B-n43-k6	38	**55**	**0.0000**	1.0000	**0.9381**	0.8364	**0.0000**	0.1105	**65.54**	558.32
B-n44-k7	**33**	21	**0.0909**	0.9048	**0.9366**	0.7700	**0.0000**	0.2055	**31.75**	657.62
B-n45-k5	15	**32**	**0.0000**	0.8438	**0.8892**	0.6413	**0.0000**	0.3068	**83.25**	553.97
B-n45-k6	**31**	27	**0.0000**	1.0000	**0.9582**	0.8432	**0.0000**	0.1344	**77.08**	612.09
B-n50-k7	24	**29**	**0.0000**	1.0000	**0.8549**	0.4537	**0.0000**	0.4344	**60.14**	606.25
B-n50-k8	**33**	31	**0.0000**	1.0000	**0.9405**	0.7479	**0.0000**	0.2268	**51.40**	676.79
B-n51-k7	20	**23**	**0.0000**	0.9167	**0.9362**	0.8489	**0.0000**	0.1069	**89.11**	623.49
B-n52-k7	**28**	22	**0.0000**	1.0000	**0.9182**	0.7279	**0.0000**	0.2308	**118.82**	610.71
B-n56-k7	31	**42**	**0.0000**	1.0000	**0.9983**	0.9367	**0.0000**	0.1105	**159.59**	626.15
B-n57-k7	24	**33**	**0.0000**	1.0000	**0.8916**	0.4927	**0.0000**	0.3881	**90.56**	666.94
B-n57-k9	**20**	14	**0.0000**	1.0000	**0.9926**	0.8526	**0.0000**	0.2414	**166.22**	730.73

Table 11 continued

	Number efficient		Coverage		Hypervolume		Epsilon indicator		CPU time	
	GVNS	NSGA-II	C(NSGA-II,GVNS)	C(GVNS,NSGA-II)	GVNS	NSGA-II	GVNS	NSGA-II	GVNS	NSGA-II
B-n63-k10	**42**	19	**0.0000**	1.0000	**0.9740**	0.7233	**0.0000**	0.2887	**182.77**	738.79
B-n64-k9	**34**	23	**0.0000**	1.0000	**0.9804**	0.5269	**0.0000**	0.6287	**158.65**	797.13
B-n66-k9	32	**47**	**0.0000**	0.8776	**0.9423**	0.7419	**0.0000**	0.2740	**284.03**	748.42
B-n67-k10	**44**	40	**0.0000**	1.0000	**0.9508**	0.5781	**0.0000**	0.5189	**289.76**	721.92
B-n68-k9	32	**53**	**0.0000**	1.0000	**0.9799**	0.7502	**0.0000**	0.2326	**238.81**	758.21
B-n78-k10	**37**	33	**0.0000**	0.8108	**0.9645**	0.7478	**0.0000**	0.2983	**514.90**	809.85
E-n101-k14	**46**	20	**0.0870**	0.6250	**0.8770**	0.8612	**0.0000**	0.4145	1000.01	**863.64**
E-n101-k8	9	**30**	**0.2222**	0.4194	**0.7785**	0.7519	**0.0000**	0.4757	1000.01	**765.82**
E-n22-k4	25	**45**	**0.0000**	0.3559	**0.8343**	0.7528	**0.0104**	0.0351	**5.65**	423.47
E-n23-k3	25	**47**	**0.0000**	0.4694	**0.8249**	0.7414	0.3503	**0.1207**	**1.22**	465.50
E-n30-k3	**32**	16	**0.0625**	0.9375	**0.9320**	0.8565	0.1930	**0.1375**	**10.24**	444.65
E-n30-k4	**59**	38	**0.0000**	0.9231	**0.8892**	0.6877	0.2025	**0.1449**	**10.27**	461.31
E-n33-k4	**65**	40	**0.0000**	1.0000	**0.9534**	0.6473	**0.0062**	0.1211	**22.33**	464.65
E-n51-k5	**77**	38	**0.0000**	1.0000	**0.9676**	0.7226	**0.0000**	0.2619	**127.39**	538.09
E-n76-k10	51	**53**	**0.0000**	1.0000	**0.9730**	0.7092	**0.0000**	0.3493	**426.59**	722.09
E-n76-k14	27	**41**	**0.0000**	1.0000	**0.9772**	0.6665	**0.0000**	0.3132	**345.11**	778.50
E-n76-k15	**27**	23	**0.0000**	1.0000	**0.9715**	0.6226	**0.0000**	0.3318	**347.56**	735.13
E-n76-k7	65	**82**	**0.0000**	0.8434	**0.9185**	0.4370	**0.0000**	0.4131	**531.68**	622.43
E-n76-k8	45	**57**	**0.0000**	0.8167	**0.9748**	0.6516	**0.0000**	0.2756	**496.14**	680.60
F-n45-k4	27	**41**	**0.0000**	0.9615	**0.9018**	0.5535	**0.0000**	0.3790	**43.11**	515.15
F-n72-k4	30	**36**	**0.0000**	1.0000	**0.9479**	0.4699	**0.0000**	0.6001	**183.25**	615.68
M-n101-k10	**39**	37	**0.0000**	0.7895	**0.9539**	0.5109	**0.0000**	0.5830	1000.00	**813.37**

Table 11 continued

	Number efficient		Coverage		Hypervolume		Epsilon indicator		CPU time	
	GVNS	NSGA-II	C(NSGA-II,GVNS)	C(GVNS,NSGA-II)	GVNS	NSGA-II	GVNS	NSGA-II	GVNS	NSGA-II
M-n121-k7	43	**73**	**0.0000**	0.0274	**0.6740**	0.6010	**0.0000**	0.3781	1000.00	**794.41**
M-n151-k12	33	**74**	**0.0000**	0.0000	**0.5998**	0.5089	**0.0000**	0.6203	**1000.01**	1014.17
M-n200-k16	18	**35**	**0.0000**	0.0000	**0.6362**	0.6128	**0.0000**	0.6490	**1000.00**	1191.20
M-n200-k17	18	**36**	**0.0000**	0.0000	**0.6362**	0.6213	**0.0000**	0.5963	**1000.00**	1181.59
P-n101-k4	**39**	20	**0.1282**	0.6800	0.8452	**0.8543**	**0.0000**	0.5053	1000.00	**765.22**
P-n19-k2	**30**	22	**0.1000**	0.7727	**0.8384**	0.8279	**0.0083**	0.0084	**5.84**	383.63
P-n20-k2	**46**	21	**0.0435**	0.9091	**0.8140**	0.7924	**0.0006**	0.0414	**6.99**	391.90
P-n21-k2	**32**	28	**0.0938**	0.9310	**0.7846**	0.7606	**0.0159**	0.0792	**8.65**	394.35
P-n22-k2	**26**	11	**0.1923**	0.8400	0.8117	**0.8276**	**0.0001**	0.0672	**9.87**	388.58
P-n22-k8	**9**	8	**0.1111**	0.6176	0.6896	**0.8898**	**0.0131**	0.1360	**1.06**	568.84
P-n23-k8	**48**	36	**0.0000**	1.0000	**0.9038**	0.7594	**0.0782**	0.0713	**2.02**	705.04
P-n40-k5	**55**	50	**0.0000**	1.0000	**0.8912**	0.6478	**0.0000**	0.1694	**50.76**	493.51
P-n45-k5	**45**	28	**0.0000**	1.0000	**0.8577**	0.6419	**0.0000**	0.2558	**77.99**	515.08
P-n50-k10	**43**	19	**0.0000**	1.0000	**0.9163**	0.7790	**0.0000**	0.2158	**78.66**	660.34
P-n50-k7	33	**40**	**0.0000**	0.9250	**0.8890**	0.7256	**0.0000**	0.2418	**107.11**	537.67
P-n50-k8	32	**33**	**0.0000**	1.0000	**0.9258**	0.6657	**0.0000**	0.1633	**92.78**	627.02
P-n51-k10	**52**	46	**0.0000**	1.0000	**0.9273**	0.7156	**0.0000**	0.2377	**70.28**	739.92
P-n55-k10	**42**	41	**0.0000**	1.0000	**0.9473**	0.7363	**0.0000**	0.2675	**122.09**	664.84
P-n55-k15	**22**	13	**0.0000**	0.8571	**0.9246**	0.8690	**0.0000**	0.1074	**72.74**	773.50
P-n55-k7	24	**56**	**0.0000**	0.9107	**0.9129**	0.6123	**0.0000**	0.3093	**156.25**	541.66
P-n60-k10	25	**36**	**0.0000**	0.8947	**0.9135**	0.7594	**0.0000**	0.3018	**167.52**	705.50
P-n60-k15	**48**	35	**0.0000**	0.9143	**0.9264**	0.6148	**0.0000**	0.1977	**120.48**	772.24
P-n65-k10	**41**	31	**0.0000**	1.0000	**0.9632**	0.7018	**0.0000**	0.3437	**231.83**	742.84

Table 11 continued

	Number efficient		Coverage		Hypervolume		Epsilon indicator		CPU time	
	GVNS	NSGA-II	C(NSGA-II,GVNS)	C(GVNS,NSGA-II)	GVNS	NSGA-II	GVNS	NSGA-II	GVNS	NSGA-II
P-n70-k10	72	**87**	**0.0000**	1.0000	**0.9538**	0.7121	**0.0000**	0.3040	**308.46**	706.56
P-n76-k4	**78**	49	**0.0000**	0.8776	**0.9425**	0.7034	**0.0000**	0.2774	650.29	**636.70**
P-n76-k5	**100**	85	**0.0000**	0.9059	**0.9173**	0.5057	**0.0000**	0.2732	**596.47**	612.78
Mean	36.3043	**36.5761**	**0.0170**	0.8612	**0.9022**	0.7238	**0.0125**	0.2497	**214.64**	640.00
# best	**44**	**44**	**92**	0	**88**	4	**88**	4	**86**	6

6 Conclusions and future works

In this paper a general variable neighborhood search (GVNS) algorithm is proposed to deal with the Multi-Objective Open Vehicle Routing Problem (MO-OVRP), a combination of the OVRP and the BOVRP that calls for the minimization of the number of vehicles, the total cost and the makespan. The relevance of this problem relies on its practical interest having into account the company's perspective as well as the customer or the workers' satisfaction.

More precisely, we propose a constructive procedure and different neighborhood sequences, which optimize the objective functions following different orders. The considered strategies were embedded in a GVNS algorithm. An extensive experimental comparison was provided to decided the best strategy. Furthermore, the performance of the proposed GVNS algorithm is compared against the NSGA-II procedure, which is a well-known and competitive algorithm used to solve a high variety of multi-objective problems and whose efficiency has been more than proved in the literature.

Computational results show the superiority of the GVNS over the competitive NSGA-II. Our results establish the first benchmarks for this problem that can be used for future developments and improvements.

As follow up to this research, we are planning to apply the proposed procedure to a real-world problem with a larger size. Furthermore, it could be interesting to include or consider other objective functions. For instance, in a company, all salesmen should have the same quantity of product to sale (that can be seeing as vehicle capacity) since usually they receive commissions. Another objective appears in humanitarian logistic activities where an important objective is to minimize the maximum latency instead of the classical latency, that is, the waiting time of the last served customer. The classical latency is often considered in competitive environments of rival companies.

Besides, we will try to improve the performance of the algorithms by working on their implementation. We will study new codification schemes for the solutions in the NSGA-II algorithm as well as modifications like the parallelization of the GVNS algorithm.

Acknowledgements J. M. Colmenar and J. Sánchez-Oro are supported by the Spanish Ministry of "Economía y Competitividad", Grant Refs. TIN2015-65460-C2-2-P and TIN2014-54806-R. A.D. López-Sánchez acknowledge support from the Spanish Ministry of Science and Innovation through Projects ECO2013-47129-C4-1-R and ECO2016-76567-C4-1-R.

References

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evolut. Comput. **6**(2), 182–197 (2002)

Duarte, A., Pantrigo, J., Pardo, E., Sánchez Oro, J.: Parallel variable neighbourhood search strategies for the cutwidth minimization problem. IMA J. Manag. Math. **27**(1), 55 (2016)

Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing (2003)

Fu, Z., Eglese, R., Li, L.Y.O.: A new tabu search heuristic for the open vehicle routing problem. J. Oper. Res. Soc. **56**(3), 267–274 (2005)

Gillett, B., Miller, L.: A heuristic algorithm for the vehicle-dispatch problem. Oper. Res. **22**(2), 340–349 (1974)

Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001)

Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO J. Comput. Optim. **5**(3), 423–454 (2017)

López Sánchez, A.D., Hernández Díaz, A.G., Vigo, D., Caballero, R., Molina, J.: A multi-start algorithm for a balanced real-world open vehicle routing problem. Eur. J. Oper. Res. **238**(1), 104–113 (2014)

Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)

Norouzi, N., Tavakkoli Moghaddam, R., Ghazanfari, M., Alinaghian, M., Salamatbakhsh, A.: A new multi-objective competitive open vehicle routing problem solved by particle swarm optimization. Netw. Spat. Econ. **12**(4), 609–633 (2012)

Sánchez Oro, J., Pantrigo, J.J., Duarte, A.: Combining intensification and diversification strategies in VNS. An application to the Vertex Separation problem. Comput. Oper. Res. **52**(Part B), 209–219 (2014). Recent advances in Variable neighborhood search

Sánchez Oro, J., Sevaux, M., Rossi, A., Martí, R., Duarte, A.: Solving dynamic memory allocation problems in embedded systems with parallel variable neighborhood search strategies. Electron. Notes Discret. Math. **47**, 85–92 (2015)

Schrage, L.: Formulation and structure of more complex/realistic routing and scheduling problems. Networks **11**(2), 229–232 (1981)

Tarantilis, C.D., Ioannou, G., Kiranoudis, C.T., Prastacos, G.P.: Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. J. Oper. Res. Soc. **56**(5), 588–596 (2005)

Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2002)

We, Z., Tingxin, S., Fei, H., Xi, L.: Multiobjective Vehicle Routing Problem with Route Balance Based on Genetic Algorithm. Discrete Dynamics in Nature and Society **2013**(Article ID 325686), 9 (2013)

Zitzler, E.: Evolutionary algorithms for multiobjective optimization: methods and applications. PhD Thesis, Swiss Federal Institute of Technology Zurich (1999)

Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms—A comparative case study, pp. 292–301. Springer, Berlin, Heidelberg (1998)

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multi-objective optimizers: an analysis and review. Trans. Evol. Comp **7**(2), 117–132 (2003)