J Glob Optim (2015) 63:515–536
DOI 10.1007/s10898-014-0213-z

Multi-objective variable neighborhood search: an application to combinatorial optimization problems

Abraham Duarte · Juan J. Pantrigo · Eduardo G. Pardo · Nenad Mladenovic

Received: 15 February 2013 / Accepted: 12 June 2014 / Published online: 25 June 2014

Abstract Solutions to real-life optimization problems usually have to be evaluated considering multiple conflicting objectives. These kind of problems, known as multi-objective optimization problems, have been mainly solved in the past by using evolutionary algorithms. In this paper, we explore the adaptation of the Variable Neighborhood Search (VNS) metaheuristic to solve multi-objective combinatorial optimization problems. In particular, we describe how to design the shake procedure, the improvement method and the acceptance criterion within different VNS schemas (Reduced VNS, Variable Neighborhood Descent and General VNS), when two or more objectives are considered. We validate these proposals over two multi-objective combinatorial optimization problems.

Keywords Multi-objective VNS · Multi-objective optimization · Antibandwidth · Cutwidth · Knapsack

1 Introduction

Solutions to real-life optimization problems usually have to be evaluated considering different points of view corresponding to multiple objectives, often in conflict. In this kind of problems, the goal is to optimize (minimize or maximize) r objective functions $(z_1, z_2, \ldots, z_r)$. Specifically, we want to determine the set of efficient solutions, also known as non-dominated points or Pareto front. We refer the reader to [4,20,37], and [13] for some tutorials on multi-objective optimization.

A. Duarte (✉) · J. J. Pantrigo · E. G. Pardo
Dept. Ciencias de la Computación, Universidad Rey Juan Carlos,
c/ Tulipán s/n, 28933 Móstoles, Madrid, Spain
e-mail: abraham.duarte@urjc.es

N. Mladenovic
Laboratory of Industrial and Human Automation control, Mechanical Engineering and Computer Science (LAMIH), Université de Valenciennes, Mont Houy, 59313 Valenciennes, France

Variable Neighborhood Search (VNS) is a metaheuristic proposed by Mladenovic and Hansen [27] as a general framework to solve hard optimization problems. It is based on the idea of systematic changes of neighborhood in both, the descent phase (to find a local optimum), and the perturbation phase (to get out of the corresponding basin of attraction). The original metaheuristic has been widely evolved with many extensions. Only to mention a few: Variable Neighborhood Descent (VND), Reduced VNS (RVNS), Basic VNS (BVNS), Skewed VNS (SVNS), General VNS (GVNS), Variable Neighborhood Decomposition Search (VNDS) or Reactive VNS, among others. See [18,19] for a recent thorough review. VNS has been successfully applied to a wide variety of optimization problems such as Max-Cut [7], Antibandwidth [24], or Cutwidth [30]. However, as far as we know, the adaptation of VNS to multi-objective optimization problems has been mainly ignored by the research community. The first multi-objective VNS algorithm was proposed by Geiger [14]. The main goal of that paper was to solve the Permutation Flow Shop Scheduling problem minimizing different combinations of criteria. The author claimed that, to solve these kind of problems, it is important to define an appropriate neighborhood structure. The paper presented a study of neighborhood search operators for the considered problem. Experimental results showed that no single neighborhood operator is able to identify all Pareto optimal alternatives. However, when these approaches are hybridized into a multi-objective VNS, significant improvements were observed. A similar approach has also been used to solve multi-objective Parallel Machine Scheduling [23] and multi-objective Redundancy Allocation Optimization [22] problems. More recently, Claudio et al. [3] addressed the Single Machine Scheduling problem with Sequence Dependent Setup Times and Distinct due Windows, using three multi-objective algorithms based on VNS. The authors introduced two intensification procedures to improve the multi-objective VNS algorithm proposed by Geiger [14]. The performance of the algorithms was tested on a set of medium and large-sized instances, showing that the proposed algorithms outperform, in terms of quality, the original multi-objective VNS algorithm. Schilde et al. [33] proposed a VNS algorithm hybridized with a Path Relinking procedure to tackle the Bi-orienteering problem. However, this bi-objective problem was transformed into a single-objective optimization problem by generating random weights and aggregating all the objectives (multiplied by its corresponding weight) in a single function. The algorithm also includes a set of efficient solutions, but they were found by optimizing the aggregated function.

The main objective of this paper is to propose the adaptation of the VNS framework to deal with combinatorial multi-objective optimization problems. In order to success in this objective, we first redefine the concept of solution in the multi-objective optimization context. In particular, the solution is now defined as the approximate set of efficient points found during the search process. Indeed, this redefinition is valid for any other trajectory-based metaheuristic such us Tabu Search, Simulated Annealing, or Iterated Local Search, among others. The new definition of solution allows us to also redefine the meaning of improving, when referring to moves performed in multi-objective optimization. Specifically, we consider that a move improves the current solution when a new generated point is included in the approximate set of efficient points (i.e., in the solution). Taking these two definitions into account the adaptation of different VNS variants becomes natural for researchers familiar with this methodology. We illustrate this fact by adapting RVNS, VND, and GVNS, to solve two challenging multi-objective combinatorial optimization problems. In particular, we describe how to design the shake, the improvement and the acceptance criterion procedures, when two or more objectives are considered. Finally, we provide experimental evidences of the effectiveness of our proposals when

comparing them with two classical multi-objective methods: the Non-dominated Sorting Genetic Algorithm II, usually denoted as NSGA-II [6], and the Strength Pareto Evolutionary Algorithm 2, known as SPEA2 [36]. The comparison was performed using specific multi-objective metrics (hyper-volume, utility functions, epsilon indicator, and coverage).

The rest of the paper is organized as follows: in Sect. 2 we reinforce the new definition of solution and the new Neighborhood Change phase of VNS for the multi-objective context. Section 3 is devoted to detail the Shake procedure. Next, in Sect. 4, we propose the adaptation of the RVNS, VND and GVNS algorithms to tackle multi-objective combinatorial optimization problems. The case studies, and the particularization of the methodology for the considered problems are presented in Sects. 5 and 6, respectively. Finally, the computational experiments and concluding remarks are conducted in Sects. 7 and 8.

2 Multi-objective neighborhood change

One of the main reasons why population-based metaheuristics (Genetic Algorithms, Particle Swarm Optimization, Differential Evolution, etc.) are more used than trajectory-based metaheuristics (VNS, Tabu Search, Simulated Annealing, etc.) in the multi-objective context is because the former uses a set of solutions (which can be easily identified with the Pareto front) while the later traditionally uses only one solution. We propose to overcome this situation by considering the approximate Pareto front found during the search process as the incumbent solution to a multi-objective problem for trajectory-based metaheuristics. As far as we know, this is the first time that the approximate Pareto front is treated as a solution to a multi-objective problem in this context. In fact, this new approach might be helpful for any trajectory-based metaheuristic to deal with multi-objective optimization problems. For the rest of the paper, we denote the elements in the Pareto front as efficient points and we reserve the term solution only to denote the approximate set of efficient points (i.e., the approximate Pareto front).

In some optimization problems this approach has been proved to be very computationally demanding since the size of the efficient set is too large. If that fact happens, it might be convenient to use a reduced set of efficient points (with the desired size) as the incumbent solution. The selection criterion of the points depends on the problem. See for instance [4] and [13]. The remaining efficient points would be stored in an external archive (as it is customary in population-based metaheuristics). The final solution is conformed with all the non-dominated points found during the whole exploration (selected from the archive and from the solution).

We now define an improving move as the one that includes a new efficient point x in the solution E. This definition is used in the multi-objective improvement procedure to compare two solutions. Algorithm 1 illustrates the pseudocode of the procedure (MO-Improvement). Specifically, this method tests whether points in E' are dominated or not by any point in E (procedure Dominated in step 3). If there is at least one non-dominated point in E', then the method returns `True` (step 4); otherwise (i.e., all points in E' are dominated), it returns `False` (step 7).

Algorithm 1 Pseudocode of multi-objective Improvement

```
1: procedure MO-Improvement(E, E')
2: for all x ∈ E' do
3:   if (x ∉ E ∧ ¬ Dominated(x, E)) then
4:     return True
5:   end if
6: end for
7: return False
8: end MO-Improvement
```

Once the definition of improvement in the multi-objective context (see Algorithm 1) is at hand, we can also redefine the neighborhood change procedure of the VNS framework for multi-objective optimization (see Algorithm 2). In this algorithm, if an improvement in the approximate set of efficient points (i.e., in the solution) is performed (step 2), the procedure indicates that the exploration must go back to the initial neighborhood (step 3). Additionally, an improvement implies that the solution must be updated by inserting those non-dominated points in the approximate set of efficient points and by removing the subset of points that become dominated (step 4). Otherwise, the algorithm will continue with the next available neighborhood (step 6). Notice that the Update procedure is not explicitly described since its implementation is trivial. We would like to draw attention about the similarity between MO-NeighborhoodChange and the classical single-objective NeighborhoodChange. This similarity is due to the new definition of solution and improvement move.

Algorithm 2 Pseudocode of multi-objective Neighborhood Change

```
1: procedure MO-NeighborhoodChange(E, E', k)
2: if MO-Improvement(E, E') then
3:   k ← 1
4:   E ← Update(E, E')
5: else
6:   k ← k + 1
7: end if
8: end MO-NeighborhoodChange
```

3 Multi-objective shake

The shake stage is usually introduced in VNS as an efficient strategy to get out from a basin of attraction. In single-objective optimization, given a solution s and the k-th neighborhood $N_k(s)$, the shake procedure usually generates, at random, a solution $s' \in N_k(s)$. We use s to denote a solution to a single-objective optimization problem, E to represent a solution (approximate set of efficient points) to a multi-objective problem, and $x \in E$ to denote an efficient point.

We propose in this paper an extension of the design of the shake procedure adapted to the multi-objective optimization context. In this new design, the size of the perturbation of any point is k. Notice that this adaptation is not unique and there could be other approaches.

The pseudocode of MO-Shake (shown in Algorithm 3) has two input arguments: an initial solution E (i.e., the approximate set of efficient points) and the size of the perturbation

(k) that will be applied to each point $x \in E$. The algorithm starts by initializing the set of perturbed points (step 2). Then, for each point $x \in E$ the procedure randomly perturbs it with a typical single-objective shake procedure. As a result, it produces a new point x' (step 4) which is included in the set of perturbed points (step 5). This process is repeated until each point in the solution E is perturbed. The method finishes by returning the new solution E' (step 7).

Algorithm 3 Pseudocode of multi-objective Shake

```
1: procedure MO-Shake(E, k)
2: E' ← ∅
3: for all x ∈ E do
4:    x' ← Shake(x, k)
5:    E' ← E' ∪ {x'}
6: end for
7: return E'
8: end MO-Shake
```

There are some optimization problems where it is required a more elaborated shake procedure since a random perturbation of the incumbent solution could not drive to promising regions of the search space. Therefore, it would be convenient that the shake procedure considers information about the objective function. In this situation, the adaptation of the shake procedure to the multi-objective context can be harder. We propose in this paper a general strategy to implement a multi-objective intensified shake procedure. This strategy perturbs all the points in the solution, where the size of the perturbation is also k, but instead of perturbing each point in a random way, it perturbs them attending to one of the objective functions considered. Taking the pseudocode presented in Algorithm 3 into account, the only difference is that the procedure Shake uses the value of the objective function. In Sect. 6 we describe several variants of this procedure for two different multi-objective optimization problems. Again, notice that this adaptation is not unique and there could be other approaches.

4 Multi-objective VNS

This section is devoted to present the adaptation of three VNS variants to solve multi-objective optimization problems. Specifically, we propose a multi-objective Reduced VNS (Sect. 4.1), a multi-objective VND (Sect. 4.2), and a multi-objective General VNS (Sect. 4.3).

4.1 Multi-objective Reduced VNS

Given the definition of MO-NeighborhoodChange and MO-Shake, presented in Sects. 2 and 3, respectively, we can easily adapt the Reduced VNS variant of the VNS framework for multi-objective optimization problems. This new version that we propose here is named multi-objective Reduced VNS (MO-RVNS) and it considers the MO-NeighborhoodChange and the MO-Shake procedures, presented in Algorithms 2 and 3 respectively, instead of the classical ones. The algorithm returns a solution which contains an approximate set of efficient points. The pseudocode of this variant is presented in Algorithm 4.

Algorithm 4 Pseudocode of multi-objective RVNS

```
1: procedure MO-RVNS(E, k_max, t_max)
2: E' ← ∅
3: repeat
4:   k ← 1
5:   repeat
6:     E' ← MO-Shake(E, k)
7:     E ← MO-NeighborhoodChange(E, E', k)
8:   until k = k_max
9:   t ← CpuTime()
10: until t > t_max
11: return E'
12: end MO-RVNS
```

We would like to draw attention again about the similarity between MO-RVNS and the classical single-objective RVNS. This similarity is due to the definition of solution and improvement move introduced in this paper.

4.2 Multi-objective Variable Neighborhood Descent

We propose an adaptation of the improvement strategy to the context of multi-objective optimization by considering two different perspectives. On the one hand, how we define the set of pre-established neighborhoods and, on the other hand, how they are explored. For the sake of simplicity, the neighborhoods are defined with respect to a point in the solution. In addition, the way in which each neighborhood is explored depends on the objective function. Specifically, we propose to use a different Variable Neighborhood Descent (VND) to improve each objective separately. For example, in an optimization problem with two objectives, we would design two VND procedures: VND-1 and VND-2. VND-1 (symmetrically VND-2) tries to improve the value of z_1 (symmetrically z_2) regardless the value of z_2 (symmetrically z_1), obtaining a local optimum with respect to z_1 (symmetrically z_2) in the set of pre-established neighborhoods.

It is worth mentioning that an important difference between single-objective optimization and multi-objective optimization is that in the former, we only need to check whether the final point obtained by the corresponding algorithm (the local optimum) improves upon the incumbent point or not. On the contrary, in multi-objective optimization, every point visited in the search must be checked for its possible inclusion in the solution. Therefore, each VND-i returns the solution found in the search process (i.e., the approximate set of efficient points found). The pseudocode of VND-i is shown in Algorithm 5. Given a point x, this procedure performs a deterministic exploration of several neighborhoods $N'_k(x)$, where $1 \leq k \leq k'_{max}$ (see step 5). Notice that these neighborhoods are different than the ones considered in MO-RVNS. In particular, N'_k neighborhoods are related with local descent procedures (i.e., each N'_k typically identifies a different local search method), while N_k neighborhoods (used in MO-RVNS) are related with a random perturbation of all the points in the current solution, where the size of the perturbation of each one is k.

We would like to highlight that the only differences of VND-i with respect to the single-objective variant are the following two: (i) step 6, which tests whether x' qualifies to enter in E or not; and (ii) step 14, where the approximate set of efficient points (i.e., the corresponding solution) is returned.

Algorithm 5 Pseudocode of VND-i

1: **procedure** VND-$i(x, k'_{max})$
2: $k \leftarrow 1$
3: $E \leftarrow \{x\}$
4: **repeat**
5: $x' \leftarrow \arg \min_{y \in N'_k(x)} z_i(y)$ // *Find the best neighbor in* $N'_k(x)$
6: Update$(E, N'_k(x))$
7: **if** $z_i(x') < z_i(x)$ **then**
8: $x \leftarrow x'$
9: $k \leftarrow 1$
10: **else**
11: $k \leftarrow k + 1$
12: **end if**
13: **until** $k = k'_{max}$
14: **return** E
15: **end** VND-i

We propose in this paper a general template for multi-objective VND, which we name MO-VND. It alternates improvements in each objective until no further improvements are found. At the end of the procedure, this method ensures that each point in the solution has been improved with respect to each objective function and each neighborhood. In order to do that, we introduce sets of exploited points, where each one S_i (for all $1 \leq i \leq r$, where r is the number of objective functions) records the points already intensified with VND-i. This strategy has two goals. On the one hand, to avoid the intensification of a point already exploited and, on the other hand, to prevent the ending of MO-VND without having exploited all the points in the solution.

Algorithm 6 Pseudocode of multi-objective VND

1: **procedure** MO-VND(E, k'_{max}, r)
2: $S_1 \leftarrow \emptyset, S_2 \leftarrow \emptyset, \ldots, S_r \leftarrow \emptyset$
3: $i \leftarrow 1$
4: **repeat**
5: **repeat**
6: $x' \leftarrow$ Select$(E \setminus S_i)$ // *Random selection among the non-exploited points*
7: $E_i \leftarrow$ VND-$i(x', k'_{max})$
8: $S_i \leftarrow S_i \cup E_i$
9: **until** $E \setminus S_i = \emptyset$
10: MO-ObjectiveChange(E, S_i, i)
11: **until** $i > r$
12: **return** E
13: **end** MO-VND

In Algorithm 6 we show the pseudocode of the MO-VND procedure. It has three input parameters, the starting solution (E), the number of objective functions (r), and the maximum number of considered neighborhoods (k'_{max}) in each VND-i. The algorithm initializes the sets of exploited points from the solution for each VND-i (step 2) and the initial value of i (step 3). Then, MO-VND performs iterations (steps 4–11) until exploring the last neighborhood for each objective (i.e., i larger than r). In step 6, the algorithm randomly selects a point x' from the solution, but excluding those points already exploited with VND-i (which are stored in S_i). Then, starting from x', the algorithm improves z_i,

returning a new approximate set of efficient points E_i that will be merged with other points obtained with the same VND-i, but starting from a different point of E. Considering that each VND-i is a deterministic procedure, all points in E_i converge to the same basin of attraction. Then, all of them are marked as exploited (step 8) with respect to objective z_i. As it is usual in single-objective VND, the multi-objective version checks whether the method has improved the current solution or not. As it was previously mentioned, the meaning of improving in this context implies that, at least, one new point has been added to the solution.

If there is an improvement, the algorithm updates E and resorts to the first objective ($i = 1$) and, additionally, the solution E is updated with the points in S_i that qualify (i.e., the non-dominated points) to enter in E (this is performed in the procedure MO-ObjectiveChange in step 10). Notice that the pseudocode of this procedure is not provided since its implementation is equal to MO-NeighborhoodChange (see Algorithm 2). The MO-VND only explores the next objective if and only if, all the points in the solution E have been explored with the corresponding VND-i. If there is an improvement in any objective (larger than 1), the MO-VND resorts again to the first objective.

4.3 Multi-objective General VNS

In single-objective optimization problems, the Basic VNS (BVNS) and the General VNS (GVNS) are conceived as procedures which balance intensification and diversification stages. Specifically, the intensification is performed by the improvement procedure (a local search in BVNS and a VND in GVNS), while the diversification is achieved by the shake procedure. The adaptation of GVNS to the multi-objective context is straightforward when considering MO-NeighborhoodChange (Algorithm 2), MO-Shake (Algorithm 3), and MO-VND (Algorithm 6). In Algorithm 7 we present the pseudocode of multi-objective GVNS (denoted as MO-GVNS). As in the case of MO-RVNS and MO-VND, we would like to highlight the similarities between the proposed MO-GVNS and the classical single-objective variant. Again, this similar design is possible due to the definition of solution and improving move proposed in this paper.

Algorithm 7 receives as input parameters a solution, E; the maximum size of the perturbation performed by the shake procedure, k_{max}; the number of objective functions, r; and the number of different neighborhoods explored by each VND-i procedure, k'_{max}. In order to provide a clear stopping criterion, the algorithm also receives the maximum computing time allowed, t_{max}. The algorithm starts by initializing the current neighborhood to the first one (step 3). Points in E are then perturbed using the function MO-Shake (step 5), obtaining a new solution E'. Then, E' is improved with the MO-VND, which returns the solution E'' (step 6). In step 7, it is decided whether the MO-GVNS needs to explore a larger neighborhood (E'' does not improve E) or not (which implies to set $k = 1$). Notice that the comparison between E and E'' is performed with the MO-Improvement procedure described in Sect. 2. Steps 4–8 are repeated until k_{max} is reached. This process is then repeated while there is time available (step 10) starting in each iteration from the incumbent solution.

Algorithm 7 Pseudocode of multi-objective GVNS

```
1: procedure MO-GVNS (E, k_max, r, k'_max, t_max)
2: repeat
3:   k ← 1
4:   repeat
5:     E' ← MO-Shake(E, k)
6:     E'' ← MO-VND(E', k'_max, r)
7:     E ← MO-NeighborhoodChange(E, E'', k)
8:   until k = k_max
9:   t ← CPUTime()
10: until t > t_max
11: return E
12: end MO-GVNS
```

5 Case studies

The definition of solution in the context of multi-objective optimization introduced in this paper establishes that a solution contains an approximate set of efficient points. Without loss of generality, each point x is formed by a set of variables $x_i, \ldots, x_n$, being n the size of the problem. Depending on the values of each x_i we can distinguish among different types of problems: continuous ($x_i \in \mathbb{R}$), binary ($x_i \in \{0, 1\}$), integer ($x_i \in \mathbb{N}$), or permutations ($1 \leq x_i \leq n$ and $x_i \neq x_j$ if $i \neq j$). We could also identify a fifth class of problems which encompasses mixed variables.

The discipline in charge of studying continuous problems is usually referred to as Continuous Optimization, while the discipline which tackles the other kind of problems is called Combinatorial Optimization. In this paper, we focus our attention on the adaptation of the proposed multi-objective VNS to deal with multi-objective Combinatorial Optimization problems. Therefore, we postpone the study of the adaptation of multi-objective VNS for multi-objective Continuous Optimization problems for future works. In order to accomplish the aforementioned study, we selected two challenging multi-objective combinatorial optimization problems to test the effectiveness of our multi-objective VNS variants. In particular:

- Multi-objective Knapsack, MOKP (binary)
- Multi-objetive Antibandwidth–Cutwidth, MO-ABCW (permutation)

We center our attention on these two problems because they are well known by the scientific community, they are different in nature (binary vs. permutation) and they have high-quality solutions available for several instances of each problem. These two case studies allow us to illustrate the performance of the proposed methods. On the one hand, it is relatively simple to find efficient points for the MOKP. Then, we will be able to test the performance of our multi-objective VNS variants when dealing with large efficient sets. On the other hand, it is very hard to obtain non-dominated points in the MO-ABCW. Therefore, we will test whether our search strategies are able to find efficient points or not. The rest of the section is devoted to provide a brief description of each problem as well as a survey of the most relevant heuristic methods presented for them.

5.1 Multi-objective knapsack problem

The single objective Knapsack Problem (KP) is one of the most studied combinatorial optimization problems in the operations research literature. The Multi-Objective Multidimensional Knapsack Problem (MOMKP) can be considered a generalization of the classical KP

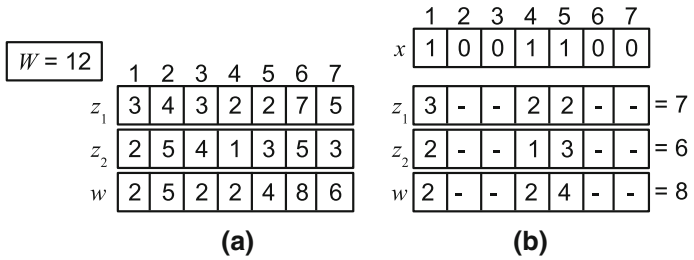


Fig. 1 **a** Instance representation with 7 items available, its associated weight and the value of each item. **b** A candidate point for the MOKP problem

where we have a set of n items, with q different characteristics (weight, volume, etc.), and r profits. The problem then consists of selecting a subset of items which maximize the r total profits without exceeding the q knapsack capacities W_l (with $1 \leq l \leq q$). In mathematical terms:

$$\text{(MOMKP)} \quad \max z_i(x) = \sum_{j=1}^{n} c_i^j x_j \qquad 1 \leq i \leq r$$

$$s.t.$$

$$\sum_{j=1}^{n} w_l^j x_l \leq W_l \qquad 1 \leq l \leq q$$

$$x_j \in \{0, 1\} \qquad 1 \leq j \leq n$$

where $x_j = 1$ indicates that the item j is included in the knapsack. The bi-objective version (i.e., $r = 2$) with a single constraint (i.e., $q = 1$) is usually known as bi-objective Knapsack Problem (MOKP). In this case, each point is represented as a binary vector. Figure 1a shows an example of an instance of this problem, where there are 7 different items, two objective functions (z_1 and z_2) and the maximum weight, W, is 12. Figure 1b depicts a point (candidate to be part of the efficient set) represented as $x = \{1, 0, 0, 1, 1, 0, 0\}$. This representation indicates that there are 3 items (out of 7) in the knapsack. In particular, the items 1, 4, and 5. This solution is feasible since the total weight is 8 (lower than 12) and the values of the objective functions are $z_1 = 7$ and $z_2 = 6$.

Most of the papers in the related literature address this variant. In particular, Ulungu [34] proposed a Simulated Annealing (SA) method for the MOKP. The method is mainly a single-objective SA applied to an aggregated objective function, where the weights are generated with a well-diversified strategy. Abdelaziz and Krichen [1] proposed the adaptation of several principles of the Tabu Search (TS) methodology to the multi-objective case. They improved their method in [2] by hybridizing the TS with a Genetic Algorithm. Czyżżak and Jaszkiewicz [5] also proposed an improved version of the multi-objective SA presented in [1]. The initial set of non-dominated points is constructed in a similar way, but the weights are then changed according to a repulsion mechanism to force the dispersion of the solutions over the approximate Pareto front. Gandibleux and Freville [11] presented a multi-objective TS that used a weighted Tchebycheff scalar function to identify the best neighbor. The weight set is dynamically modified with the goal of redirecting the search to those regions with few points in the approximate Pareto front. Gandibleux et al. [12] introduced a two-phase genetic algorithm. The first phase is based on an exact method able to produce the

set of supported solutions. Then, in the second phase, a straightforward adaptation of a GA is applied. Gomes Da Silva et al. [15] described an adaptation of the Scatter Search (SS) metaheuristic to the multi-objective scenario. The authors described how the five strategies of SS can be adapted to the MOKP. The same authors in [16] improved the SS algorithm by including a new combination method. Lust et al. [26] proposed a multi-objective Very Large-Scale Neighborhood Search (VLSNS) based on generating high quality neighbor solutions in reasonable computing time. This method is further improved in [25], becoming it in the state-of-the-art method for the MOKP.

5.2 Multi-objective Antibandwidth–Cutwidth problem

In recent years there has been a growing interest in studying graph layout problems where the main objective is to find a labeling of a graph in such a way that a specific objective function is maximized or minimized. The Linear Arrangement [32], Bandwidth [28], or Vertex Separation [9] fall into this class of optimization problems. In this paper, we tackle a particular challenging bi-objective layout optimization problem. Specifically, it consists of finding a layout such that the minimum difference between labels of adjacent vertices is maximized and, simultaneously, the maximum linear cut between consecutive vertices in the layout is minimized. This problem can be modeled in terms of graphs as follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected and unweighted graph, where $\mathcal{V}$ is the vertex set, with $|\mathcal{V}| = n$, and $\mathcal{E}$ is the edge set. A point in this bi-objective optimization problem is usually represented as an ordering (a.k.a., labeling, permutation, layout, etc.) of the vertices. This ordering is a one-to-one function $\pi : \mathcal{V} \leftarrow \{1, 2, \ldots, n\}$.

The multi-objective Antibandwidth–Cutwidth (MO-ABCW) optimization problem then consists of finding the labeling that maximizes the antibandwidth of a graph and, simultaneously, minimizes the cutwidth of the same graph. The antibandwidth of a graph $\mathcal{G}$ with respect to the labeling π is denoted as $z_1(\pi, \mathcal{G})$ and it is mathematically defined as:

$$z_1(\pi, \mathcal{G}) = \min_{v \in \mathcal{V}} \{AB(\pi, v)\}$$

where,

$$AB(\pi, v) = \min_{(u,v) \in \mathcal{E}} \{\mathtt{abs}(\pi(u) - \pi(v))\}$$

where abs indicates the absolute value of $\pi(u)$ minus $\pi(v)$. We do not use the usual symbol $|.|$ since we reserve it for expressing the cardinality of a set.

The cutwitdh of a graph $\mathcal{G}$ with respect to the labeling π is denoted as $z_2(\pi, \mathcal{G})$ and mathematically defined as:

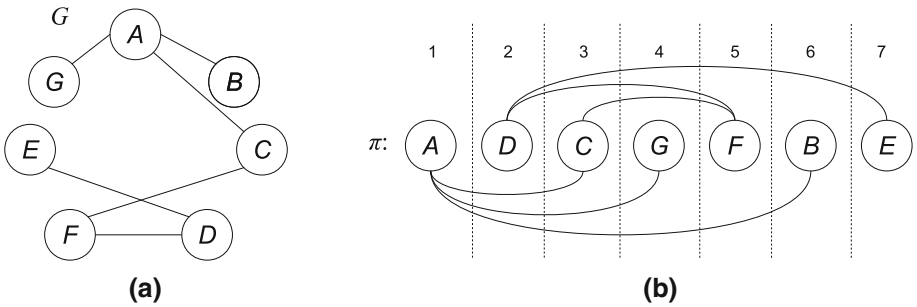
$$z_2(\pi, \mathcal{G}) = \max_{v \in \mathcal{V}} \{CW(\pi, v)\}$$

where,

$$CW(\pi, v) = |\{(u, w) \in \mathcal{E} : \pi(u) \leq \pi(v) < \pi(w)\}|$$

Then, the bi-objective MO-ABCW problem consists of finding the set of efficient points over all possible orderings Π. In mathematical terms:

$$\text{(MO-ABCW)} \begin{array}{l} \max\limits_{\pi \in \Pi} z_1(\pi, \mathcal{G}) \\ \min\limits_{\pi \in \Pi} z_2(\pi, \mathcal{G}) \end{array}$$



v	AB(π,v)
A	2
D	3
C	2
G	3
F	2
B	5
E	5

(c)

v	CW(π,v)
A	3
D	5
C	5
G	4
F	2
B	1
E	0

(d)

Fig. 2 **a** Graph example. **b** A point π in the MO-ABCW problem. **c** $AB(\pi, v)$ values for the vertices in π. **d** $CW(\pi, v)$ values for the vertices in π

Figure 2a shows an example of a graph with 7 vertices and 6 edges. Figure 2b shows a possible ordering π of the graph. In particular, the label of vertex A is $\pi(\mathtt{A}) = 1$, the label of vertex B is $\pi(\mathtt{B}) = 6$ and so on. Figure 2c reports the antibandwidth of each vertex calculated as the minimum difference between the label of the corresponding vertex and all the labels of its neighbors. For example, $AB(\pi, \mathtt{A}) = 2$ since $\pi(\mathtt{A}) = 1$ and the labels of their adjacent vertices are $\pi(\mathtt{C}) = 3$, $\pi(\mathtt{G}) = 4$, and $\pi(\mathtt{B}) = 6$. Then, $\min\{\mathtt{abs}(1-3), \mathtt{abs}(1-4), \mathtt{abs}(1-6)\} = 2$. Finally, Fig. 2d reports the cutwidth of each vertex. For example, the cutwidth of vertex C, $CW(\pi, \mathtt{C}) = 5$, is obtained by counting the edges that have an endpoint on C or before C in π, and the other endpoint after C in π. Specifically, these edges are (A, G), (A, B), (D, F), (D, E), and (C, F). Then, the value of the first objective is $z_1 = 2$ since it is computed as the minimum across the AB-values, while the value of the second objective is $z_2 = 5$ since it is the maximum of the cutwidth of all vertices.

We have selected this problem since each objective function separately is extremely hard to solve for most of the search procedures since they present a flat landscape (see for instance [8, 31], or [30]). It means that there may be many different points with the same objective function value, which makes that most of the moves performed by a search procedure result in a null value. Therefore, the combination of both objectives becomes in an even harder optimization problem. Consequently, obtaining a large amount of non-dominated points is a real challenge.

6 Adaptation of MO-VNS to the case studies

This section describes an adaptation of the multi-objective VNS strategies (MO-RVNS, MO-VND, and MO-GVNS) presented in this paper, to the two aforementioned optimization

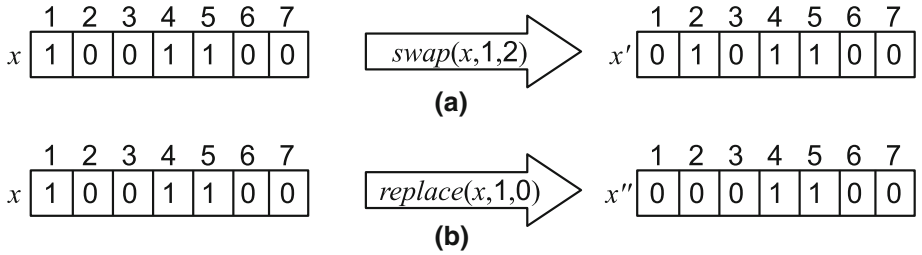


Fig. 3 **a** Swap move for the MOKP problem. **b** Replace move for the MOKP problem

problems. Attending to the procedures showed in Algorithms 4, 5, and 7, we can see that they mainly depend on the particularization of MO-Shake, and VND-i to the specific problem, since MO-NeighborhoodChange and MO-ObjectiveChange are problem independent (see Algorithm 2). MO-Shake depends on the problem through the procedure Shake. Therefore, if we want to apply any variant of the multi-objective VNS for a specific multi-objective problem, we are only required to design the VND-i and Shake procedures.

6.1 Adaptation of VND-i

Attending to Algorithm 5, the only problem-dependent instruction is the one that finds the best point in a neighborhood (step 5). In general, neighborhoods are built from an efficient point representation and a move mechanism. In our context, we represent a point x as a vector of n variables, i.e., $x = (x_1, x_2, \ldots, x_n)$. With this representation, we define the following moves:

- *Exchange of values*: Given a point x, the exchange of the values of the variables x_i and x_j is denoted by $swap(x, i, j)$ and generates a new point x' such that $x'_i = x_j \wedge x'_j = x_i$.
- *Replacement of a single value*: Given a point x, the replacement move denoted by $replace(x, j, v)$ creates a point x' for which $x'_j = v$. The move is such that $x_j \neq v$.

We denote *swap neighborhood* the one generated by the swap moves and *replace neighborhood* the one generated by the replacement moves. To be coherent with the notation used in Algorithm 5, we identify *swap neighborhood* with N'_1, *replace neighborhood* with N'_2 and therefore, k'_{max} is equal to 2.

The definition of neighborhoods strongly depends on the particular representation of the corresponding point. As it was described in Sect. 5.1, each point in the MOKP is represented as a binary vector. In this case, the $swap$ move removes one element in the knapsack and includes another element not present in the knapsack. For example, considering the example shown in Fig. 1, the move $swap(x, 1, 2)$ generates a new point $x' = \{0, 1, 0, 1, 1, 0, 0\}$, depicted in Fig. 3a, means that the item 1 is removed from the current knapsack x and item 2 is included on it, obtaining x'. Similarly, $replace$ moves implies to add an element (if that item is not in the knapsack) or to remove it (if the item is in the knapsack). For example, $replace(x, 1, 0) = \{0, 0, 0, 1, 1, 0, 0\}$, depicted in Fig. 3b means that item 1 is removed from the current knapsack. It is important to remark that not all moves produce feasible points. For instance, the interchange of two elements or the addition of an element could violate the capacity constraint. For the sake of simplicity, we only consider feasible moves. Notice that removing one element is always feasible for this problem.

Again, as it was pointed out above, the definition of neighborhoods is problem-dependent. Section 5.2 describes that each point in the MO-ABCW is represented as a permutation of n

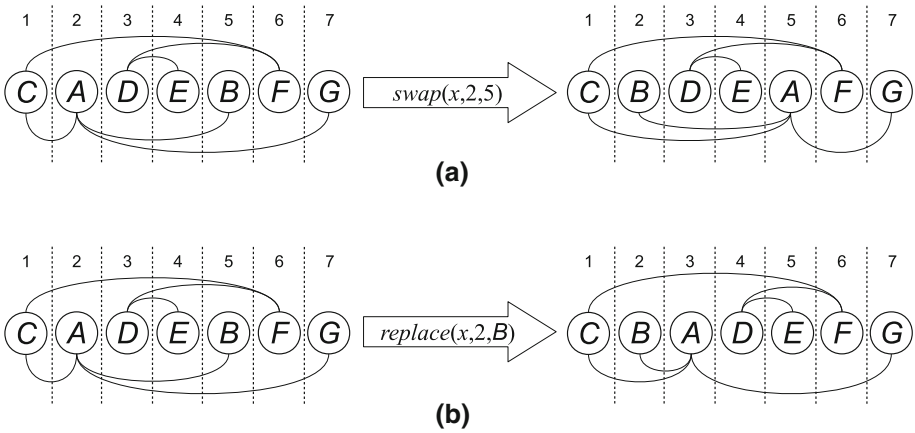


Fig. 4 **a** Swap move for the MO-ABCW problem. **b** Replace move for the MO-ABCW problem

variables (i.e., a point x is an ordering of the vertices of the graph). Considering the graph introduced in Fig. 2, we show an example of $swap$ move in Fig. 4a. In particular, if we consider the point $x = \{\texttt{C}, \texttt{A}, \texttt{D}, \texttt{E}, \texttt{B}, \texttt{F}, \texttt{G}\}$ and we apply the move $swap(x, 2, 5)$, it produces a new ordering where vertex B is inserted in position 2 and, simultaneously, vertex A is inserted in position 5. The new resulting point is $x' = \{\texttt{C}, \texttt{B}, \texttt{D}, \texttt{E}, \texttt{A}, \texttt{F}, \texttt{G}\}$. Figure 4b illustrates the other type of move. Specifically, considering the point $x = \{\texttt{C}, \texttt{A}, \texttt{D}, \texttt{E}, \texttt{B}, \texttt{F}, \texttt{G}\}$, if we apply the move $replace(x, 2, \texttt{B})$, then the vertex B is removed from its current position (i.e., position 5) and it is inserted in position 2, shifting the rest of the elements placed in a position larger than 2. Therefore, we would obtain a new point $x' = \{\texttt{C}, \texttt{B}, \texttt{A}, \texttt{D}, \texttt{E}, \texttt{F}, \texttt{G}\}$.

6.2 Adaptation of Shake procedure

As it was indicated in Sect. 5, a solution E to a multi-objective problem contains an approximate set of efficient points, where each point $x \in E$ is defined by a set of variables $x_1, \ldots, x_n$. Given a point x and the k-th neighborhood, N_k, we propose four different ways to implement the procedure Shake(x, k). All of them are based on performing k consecutive moves starting from the point x. Particularly, we propose to use the move $swap(x, i, j)$ where i and j identify variables x_i and x_j, respectively (see Sect. 6.1 for a detailed description).

In order to design more elaborated Shake procedures, we divide the $swap$ into two different stages. The first stage consists of identifying the variable x_i, while the second stage consist of identifying the variable x_j. Both stages can be performed in a random and/or a greedy way. The only restriction to apply the $swap$ move is that it results in a feasible point for the considered optimization problem.

- ***Shake*$_1$** performs the k consecutive $swap$ moves by the selection the x_i and the x_j variables at random. This method clearly favor the diversification of the search. Notice that this variant is the traditional shake implementation in the VNS community.
- ***Shake*$_2$** focuses on the intensification of the search in an objective. Specifically, it performs the k consecutive $swap$ moves by the selection the x_i and the x_j in a greedy way with respect to the intensified objective.
- ***Shake*$_3$** tries to find a balance between intensification and diversification. In particular, the selection of the x_i variables is performed at random, meanwhile the selection of the x_j variables follows a greedy criterion with respect to the intensified objective.

- ***Shake*$_4$**, in a similar way to $Shake_3$, finds a balance between intensification and diversification but, this time, the random and greedy stages are inverted. Specifically, the selection of the x_i variables is now performed in a greedy way with respect to the intensified objective and the selection of the x_j is performed at random.

Notice that other implementations of the shake procedures are possible just by changing the move used, e.g., using the $replace$ move instead of $swap$ move (see Sect. 6.1).

7 Computational experience

This section describes the computational experiments that we performed to test the effectiveness of our multi-objective VNS variants as well as to compare them with methods identified in the state of the art of multi-objective optimization. We have implemented all the procedures in Java SE 6. The experiments were conducted on an Intel Core 2 Quad CPU and 6 GB RAM. We derived the following set of instances:

- *MOKP set:* 20 instances from MOCOLib. These instances have been previously used in [11] and they correspond to 5 bi-objective 01 uni-dimensional knapsack problems. All instances have a tightness ratio $r = W/\sum_{i=1}^{n} w_i = 0.5$. For each problem, 4 variants (class A, B, C and D) are given: 1B/A, the weights and the costs are uniformly generated (in [1:100]); 1B/B, created from 1B/A by replacing the second vector of costs by the first one in reverse order; 1B/C, vector of costs generated with plateaus of values of length $\leq 0.1n$, being n the size of the problem; 1B/D, created from 1B/C by replacing the second vector of costs by the first one in reverse order.
- *MO-ABCW set:* 20 instances from the Harwell–Boeing Sparse Matrix Collection. This collection has been previously used to test algorithms in the single-objective case of both, Antibandwidth and Cutwidth. See for instance [8,29]. This benchmark consists of a set of standard test matrices $M = (M_{ij})$ arising from problems in linear systems, least squares, and eigenvalue calculations from a wide variety of scientific and engineering disciplines. Graphs are derived from these matrices by considering an edge (i, j) for every element $M_{ij} \neq 0$. From the original set we have selected the 20 graphs with $n \leq 150$. Their number of vertices ranges from 30 to 132 and the number of edges from 46 to 1758.

We have divided our experimentation into two different parts: preliminary experiments and final experiments. The following sections describe the most relevant findings of these experiments.

7.1 Preliminary experimentation

The preliminary experiments were performed to set the values of the key search parameters of each multi-objective VNS variant as well as to show the merit of the proposed search strategies. We consider a representative subset of instances with different sizes and densities. We qualitatively analyze the performance of each algorithm by considering the graphical representation of the Pareto fronts. In our first experiment we compare the four different shake procedures embedded in the MO-Shake algorithm described in Sect. 3. In particular, we compare the performance of 4 MO-RVNS variants (MO-RVNS-1, MO-RVNS-2, MO-RVNS-3, and MO-RVNS-4), where the only difference among them is the shake procedure used in each one. Each MO-RVNS-i considers the $Shake_i$ function, where $i \in \{1, 2, 3, 4\}$. As it is pointed out in [19], the best outcomes in RVNS are obtained when the value of k_{max} is set to 2 or 3. Then, we selected $k_{max} = 3$ since we obtained slightly better results. We

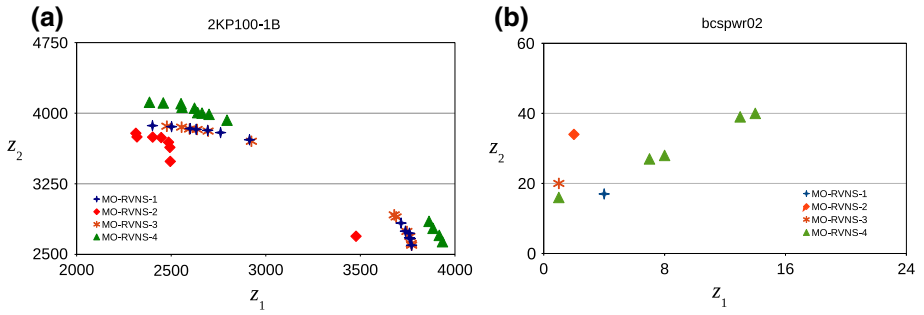


Fig. 5 Approximation of the Pareto front of the MO-RVNS variants on the **a** MOKP problem, **b** MO-ABCW

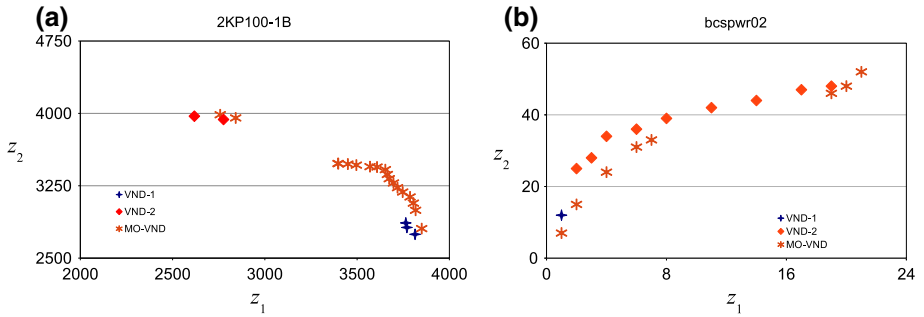


Fig. 6 Approximation of the Pareto front of the MO-VND, VND-1 and VND-2 on the **a** MOKP problem, **b** MO-ABCW problem

set t_{max} to 60 s since none of the MO-RVNS were able to significantly improve the results in longer computing times. On the other hand, we have experimentally tested that reducing this time, the quality of these methods decreases. Notice that MO-RVNS is a extremely fast method, therefore it performs millions of iterations in the computing time allowed.

Figure 5a, b shows the approximation of the Pareto front obtained by each MO-RVNS method on the MOKP and MO-ABCW, respectively. For the sake of brevity, we only show the results on a particular instance, but the analysis can be extended to the remaining ones. At a first glance, this experiment shows that none of the MO-RVNS variants is able to find a large amount of efficient points. In the case of the MO-ABCW it is even worse since this problem presents a flat landscape. In both problems, MO-RVNS-4 obtains the best outcomes, partially dominating the rest of the MO-RVNS variants. This method removes the worst elements (in terms of the two objective functions) and, afterwards, it completes them at random. It is important to remark that MO-RVNS-1 (traditional implementation of the shake procedure) obtains relatively good results, resulting in the second best method in both problems. We will use MO-RVNS-4 in the final comparison, even though it is expected that the other methods (i.e. MO-VND and MO-GVNS) will outperform this variant.

In the next experiment, we compare the performance of the MO-VND with the VND-i procedures in isolation (i.e., VND-1 and VND-2). Notice that both methods (VND-1 and VND-2) test after a move whether a point is admitted in the approximation of the Pareto front or not. Thus, both methods return an efficient set (i.e., a solution). The three methods were executed only for one iteration. Approximation to the Pareto fronts are depicted in Fig. 6a, b,

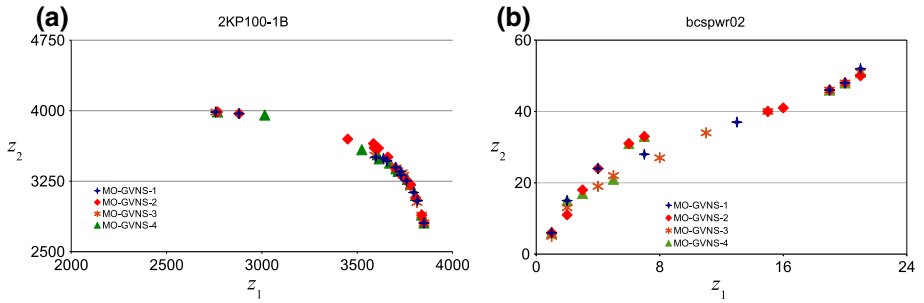


Fig. 7 Approximation of the Pareto front of the MO-GVNS variants on the **a** MOKP problem, **b** MO-ABCW problem

which confirm that the MO-VND procedure compares favorably with VND-1 and VND-2 in the two considered problems. For the MOKP, we can observe that VND-1 only obtains points with relatively good value of z_1. Symmetrically, VND-2 presents a similar behavior but in z_2. The analysis of MO-ABCW is considerably more complex since it is harder to find efficient points. In particular, VND-1 only finds 1 point while VND-2 is able to find 6. This experiment then reveals that it is more difficult to increase z_1 than to reduce z_2. In fact, all points found by VND-1 have $z_1 = 1$. On the other hand, VND-2 obtains, as expected, better results with respect to its own objective, but the associated values to z_1 present a medium quality, i.e., worse than MO-VND (as it was also expected).

The third experiment consists of selecting the best MO-GVNS variant by running a single full-factorial experiment by considering several values of t_{max} and k_{max}. Iterations in MO-GVNS are considerably larger than those in MO-RVNS due to the improvement strategy. Then, if we stop the method by setting t_{max} to a maximum computing time it is really likely to interrupt it in the middle of an iteration. Therefore, we set $t_{max} = \{15, 10, 5\}$ being these values the maximum number of iterations. Notice that we fixed the maximum value of t_{max} to 15 since the improvement when considering larger numbers is not significant for the considered case studies. Additionally we also tried shorter number of iterations (5 and 10) in order to evaluate shorter horizons of time.

For the sake of simplicity, we do not report the Pareto front found for each variant, but only the best of the four MO-GVNS strategies. These variants are: MO-GVNS-1 ($t_{max} = 5, k_{max} = 5$), MO-GVNS-2 ($t_{max} = 10, k_{max} = 10$), MO-GVNS-3 ($t_{max} = 15, k_{max} = 15$), and MO-GVNS-4 ($t_{max} = 5, k_{max} = 15$). Figure 7a, b shows the approximation of the Pareto front that find each MO-GVNS variant. As expected, the higher the value of t_{max} and k_{max}, the better the results. Unfortunately, the computing time also increases with these parameters, so we set $t_{max} = 5$ and $k_{max} = 5$ in order to reduce the computing time needed.

Figure 8a, b shows the approximation of the Pareto front obtained by the best variants. In particular, we compare MO-RVNS-4, MO-VND and MO-GVNS-1. As expected, MO-GVNS-1 emerges in both problems as the best method since it partially dominates the other two algorithms. MO-VND presents a remarkable behavior in MO-ABCW, obtaining non-dominated points in the objective z_2, although the general performance of MO-GVNS is better. Finally, MO-RVNS-4 obtains poor results in the MOKP. However, in MO-ABCW its performance is similar to the one of MO-VND, even obtaining non-dominated points.

Notice that the three multi-objective methods represented in these figures are not executed for the same CPU time. As we mentioned above, each method is run with its best configuration.

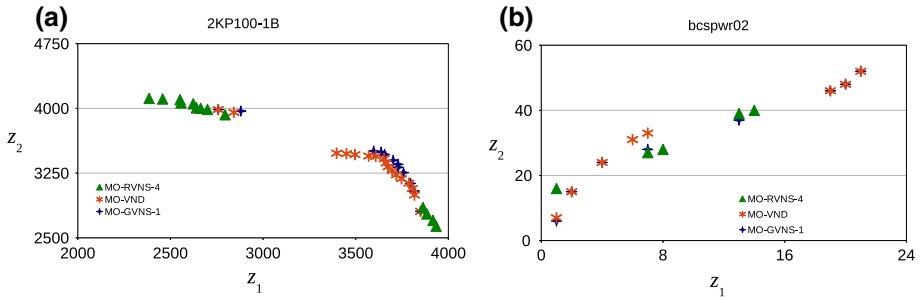


Fig. 8 Comparison of the best MO-VNS variants on the **a** MOKP problem, **b** MO-ABCW problem

With these figures, we only wanted to show the best performance for each method. In order to have a fair comparison among the methods (i.e., executed for the same computing time in the same computer) we refer the reader to Sect. 7.2.

7.2 Final comparison

The comparison of the performance of heuristic methods is always a difficult task since it involves solution quality, computational effort, robustness, and other factors. In the case of multi-objective optimization, performance assessment is even more complicated because we compare sets of points (approximations to the efficient front). We apply the indicators and guidelines proposed in [21]. In particular, they propose the following four evaluators as the most discriminant indicators to compare multi-objective methods:

- *Hyper-volume*: This metric suggested by Zitzler and Thiele [35] measures the size of the space covered. In other words, it computes the hypervolume of the portion of the objective space that is weakly dominated by an approximation set. This measure is usually computed as the hypervolume difference to a reference set where the smaller values correspond to higher quality (in contrast with the original hyper-volume).
- *Unary epsilon*: The epsilon indicator family, introduced in [37], has additive, multiplicative, unary and binary versions. In our comparison we consider the unary multiplicative version given by default in PISA: Given a front A and a reference front R, $\epsilon(A, R)$ gives the minimum factor ϵ by which each point of R can by multiplied such that the resulting transformed approximation front is weakly dominated by A. Therefore, as with the hypervolume difference, the smaller the measure, the higher the quality of the method.
- r: [17] proposed three different indicators $r1$, $r2$, and $r3$, based on using a set of utility functions, which can be defined as a mapping from the objective vectors to the set of real numbers. These utility functions represent the preferences of the decision maker. Different functions have been proposed, from the simple weighted combination of objectives to the complex Tchebychev function. Similarly to the ϵ indicator, the R indicator can be defined as unary or binary. In this section we consider the R2 unary indicator since it is a standard in the PISA project.
- C: This is known as the coverage measure of two sets [35]. $C(A, B)$ represents the proportion of points in the estimated efficient front B that are dominated by the efficient points in the estimated front A. The value $C(A, B) = 1$ means that all decision vectors in B are weakly dominated by A. The opposite, $C(A, B) = 0$, represents the situation where none of the points in B are weakly dominated by A. Note that always both directions have to be considered, since $C(A, B)$ is not necessarily equal to $1 - C(B, A)$.

Table 1 Final comparison over the whole set of instances for each bi-objective problem

	MOKP (20)				MO-ABCW (20)			
	$Avg.(\|E\|)$	$Hyp.$	$eps.$	$r2$	$Avg.(\|E\|)$	$Hyp.$	$eps.$	$r2$
MO-RVNS	25.90	0.48	0.53	0.14	2.81	0.28	0.73	0.22
MO-VND	18.30	0.16	0.06	0.02	6.15	0.10	0.16	0.02
MO-GVNS	16.01	0.14	0.05	0.02	6.05	0.06	0.12	0.01
NSGA-II	6.02	0.18	0.21	0.05	1.45	0.35	0.89	0.26
SPEA2	5.35	0.21	0.16	0.07	1.70	0.36	0.89	0.27

As mentioned in [21], if two reliable indicators, as the four above, are in contradiction then it implies that the two sets are incomparable.

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) [6], and the Strength Pareto Evolutionary Algorithm 2 (SPEA-2) [36], have become standard approaches in solving multi-objective optimization problems. In fact, in order to test the performance of any new multi-objective method, a comparison with these two methods is always desirable. We have used the implementation of these procedures provided by Durillo and Nebro [10] in the jMetal framework.

We compare in the next experiment our best variants of MO-RVNS, MO-VND and MO-GVNS with NSGA-II and SPEA-2. In order to have a fair comparison, all methods were executed for a similar CPU time in the same computer (about 150 s on average). Table 1 show the average number of efficient points found ($Avg.(|E|)$), hyper-volume ($Hyp.$), the unary epsilon ($eps.$), and the $r2$ indicator, for the five multi-objective algorithms executed over all instances in the two considered bi-objective problems. The reference set R in the first two indicators is the set of non-dominated points obtained when we merge the approximations of the efficient front obtained with the five methods. The analysis was performed with the multi-objective software library PISA (http://www.tik.ee.ethz.ch/pisa).

Considering the $Hyp.$, $eps.$ and $r2$ results presented in Table 1, they clearly indicate that the MO-GVNS is the best method overall in the two considered bi-objective problems, closely followed by the MO-VND. Specifically, our best method (MO-GVNS) obtains the lowest values in the three measures reported in these tables. MO-VND also outperforms previous methods in the two considered problems. In particular, it obtains slightly better results in the MOKP and considerably better results in the case of MO-ABCW. Finally, we highlight that NSGA-II and SPEA2 clearly outperform MO-RVNS in the MOKP. However, MO-RVNS compares favorably when considering only the MO-ABCW problem. With respect to the average number of points in the efficient set, $Avg.(|E|)$, it is interesting to remark that all the methods are able to find, on average, a larger number of points in the MOKP set of instances than in the MO-ABCW one. This fact is partially explained by the flat landscape (many solutions with the same value of the objective function) present in the solution space of the Antibandwidth and Cutwidth optimization problems.

Tables 2 and 3 show the coverage among all pairs of these five methods when considering the MOKP and MO-ABCW. The results reported in the MO-GVNS row shows that the coverage of this method is larger than the rest of the compared ones in both bi-objective optimization problems. This fact is also corroborated when considering its corresponding column since the coverage of the other methods over MO-GVNS is the smallest in the table. It is interesting to remark that NSGA-II and SPEA2 perform much better in the MOKP instances than in the MO-ABCW ones. In particular, C(MO-GVNS, NSGA-II) = 0.17 in the

Table 2 Coverage analysis of the five multi-objective algorithms for the MOKP set of instances

	MO-RVNS	MO-VND	MO-GVNS	NSGA-II	SPEA2
MO-RVNS	0.00	0.00	0.00	0.00	0.00
MO-VND	0.91	0.00	0.02	0.13	0.18
MO-GVNS	0.91	0.67	0.00	0.17	0.25
NSGA-II	0.65	0.07	0.02	0.00	0.63
SPEA2	0.52	0.06	0.02	0.12	0.00

Table 3 Coverage analysis of the five multi-objective algorithms for the MO-ABCW set of instances

	MO-RVNS	MO-VND	MO-GVNS	NSGA-II	SPEA2
MO-RVNS	0.00	0.15	0.08	0.08	0.15
MO-VND	0.34	0.00	0.14	0.50	0.54
MO-GVNS	0.43	0.42	0.00	0.53	0.52
NSGA-II	0.29	0.10	0.07	0.00	0.51
SPEA2	0.29	0.07	0.03	0.15	0.00

MOKP, while C(MO-GVNS, NSGA-II) = 0.53. Therefore NSGA-II obtains much more non-dominated points in the MOKP than in the MO-ABCW. This analysis can be also extended to SPEA2.

Conducting a similar study over the other two variants proposed in this paper, we can observe that C(MO-VND,*) > C(*, MO-VND), where * stands for MO-RVNS, NSGA-II, and SPEA2. Then, this method also outperforms the procedures in the literature, but MO-RVNS performs worse than NSGA-II and SPEA2 in terms of coverage.

8 Conclusions

In this paper we have proposed new adaptations of the single-objective VNS framework to deal with multi-objective combinatorial optimization problems. In particular, we have redefined the concept of solution in this context and we have described how to design the shake, the improvement and the acceptance criterion procedures, when two or more objectives are considered. Using some of the previous ideas we have proposed an adaptation of Reduced VNS, Variable Neighborhood Descent, and General VNS for solving multi-objective optimization problems. We would like to draw attention about the similarities between the multi-objective VNS variants and the classical single-objective ones. This similarities are due to the definition of solution and improvement move introduced in this paper. In fact, taking these two definitions into account the adaptation of different VNS variants becomes natural for researchers familiar with VNS.

To validate our proposals we have tackled two different well-known bi-objective combinatorial optimization problems, whose solutions present different structures. On the one hand, we have considered a permutation-based problem which consists of two single-objective optimization problems (the Antibandwidth Maximization Problem and the Cutwidth Minimization Problem). On the other hand, we have considered a binary problem, the Multi-Objective Knapsack Problem. For those problems, we have compared our best multi-objective VNS variants with the Non-dominated Sorting Genetic Algorithm II and the Strength Pareto Evolutionary Algorithm 2, considered as an standard in the multi-objective community. We reported

empirical tests over 20 instances for each problem, showing that the proposed heuristics outperform the reference methods.

Acknowledgments This research has been partially supported by the Spanish Ministry of "Economía y Competitividad", Grants Ref. TIN2011-28151, and TIN2012-35632-C02, and the Government of the Community of Madrid, Grant Ref. S2009/TIC-1542.

References

1. Abdelaziz, F.B., Krichen, S.: A tabu search heuristic for multiobjective knapsack problems. In: Computational Engineering in Systems applications (CESA'98). IEEE Systems Man and Cybernetics, vol. 2, pp. 212–216 (1998)
2. Abdelaziz, F.B., Krichen, S., Chaouachi, J.: A hybrid heuristic for multiobjective knapsack problems. In: Voß, S., Martello, S., Osman, I.H., Roucairol, C. (eds.) Meta-heuristics. Springer, Berlin, pp. 205–212. ISBN:978-1-4613-7646-0 (1999)
3. Claudio, J.E., dos Santos, R., de Paiva, A.: Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. Electron. Notes Theor. Comput. Sci. **281**, 5–19 (2011)
4. Coello, C., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-objective Problems. Kluwer, Dordrecht (2002)
5. Czyżżak, P., Jaszkiewicz, A.: Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. J. Multi-Criteria Decis. Anal. **7**(1), 34–47. ISSN:1099-1360 (1998)
6. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. Springer, Berlin (2000)
7. Duarte, A., Sánchez, A., Fernández, F., Cabido, R.: A low-level hybridization between memetic algorithm and vns for the max-cut problem. In: ACM Genetic and Evolutionary Computation Conference (GECCO 05), pp. 999–1006, Washington. ACM (2005)
8. Duarte, A., Martí, R., Resende, M.G.C., Silva, R.M.A.: Grasp with path relinking heuristics for the antibandwidth problem. Networks **58**, 171–189 (2011)
9. Duarte, A., Escudero, L.F., Martí, R., Mladenovic, N., Pantrigo, J.J., Sánchez-Oro, J.: Variable neighborhood search for the vertex separation problem. Comput. Oper. Res. **39**, 3247–3255 (2012)
10. Durillo, J.J., Nebro, A.J.: jMetal: a java framework for multi-objective optimization. Adv. Eng. Softw. **42**, 760–771. ISSN:0965-9978 (2011)
11. Gandibleux, X., Freville, A.: Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: the two objectives case. J. Heuristics **6**(3), 361–383. ISSN:1381-1231 (2000)
12. Gandibleux, X., Morita, H., Katoh, N.: The supported solutions used as a genetic information in a population heuristic. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) Evolutionary Multi-criterion Optimization, vol. 1993 of Lecture Notes in Computer Science, pp. 429–442. Springer, Berlin. ISBN:978-3-540-41745-3 (2001)
13. Gandibleux, X., Sevaux, M., Sorensen, K., T'kindt, V.: Metaheuristics for Multi-objective Optimization. Springer, Berlin (2004)
14. Geiger, M.J.: Randomised variable neighbourhood search for multi objective optimisation. In: Proceedings of EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-heuristics, pp. 34–42 (2004)
15. Gomes Da Silva, C., Clímaco, J., Figueira, J.: A scatter search method for bi-criteria 0-1-knapsack problems. Eur. J. Oper. Res. **169**(2), 373–391. ISSN:0377-2217 (2006)
16. Gomes Da Silva, C., Figueira, J., Clímaco, J.: Integrating partial optimization with scatter search for solving bi-criteria 0,1-knapsack problems. Eur. J. Oper. Res. **177**(3), 1656–1677. ISSN:0377-2217 (2007)
17. Hansen, M.P., Jaszkiewicz, A.: Evaluating the quality of approximations to the non-dominated set. Technical report, Technical University of Denmark. IMM-REP-1998-7 (1998)
18. Hansen, P., Mladenović, N., Moreno Pérez, J.A.: Variable neighbourhood search: methods and applications. 4OR Q. J. Oper. Res. **6**, 319–360. ISSN:1619-4500 (2008)
19. Hansen, P., Mladenović, N., Moreno Pérez, J.A.: Variable neighbourhood search: algorithms and applications. Ann. Oper. Res. **175**, 367–407 (2010)
20. Knowles, J.: Local-search and hybrid evolutionary algorithms for Pareto optimization. Ph.D. thesis (2002). http://dbkgroup.org/knowles/thesis.html
21. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers, 214. Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland (revised version) (2006)

22. Liang, Y.-C., Lo, M.-H.: Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm. J. Heuristics **16**, 511–535 (2010)
23. Liang, Y.-C., Chen, H.-L.A., Tien, C.-Y.: Variable neighborhood search for multi-objective parallel machine scheduling problems. In: Proceedings of the 8th International Conference on Information and Management Sciences, pp. 519–522 (2009)
24. Lozano, M., Duarte, A., Gortázar, F., Martí, R.: Variable neighborhood search with ejection chains for the antibandwidth problem. J. Heuristics **18**, 919–938 (2012)
25. Lust, T., Teghem, J.: The multiobjective multidimensional knapsack problem: a survey and a new approach. Int. Trans. Oper. Res. **19**(4), 495–520. ISSN:1475-3995 (2012)
26. Lust, T., Teghem, J., Tuyttens, D.: Very large-scale neighborhood search for solving multiobjective combinatorial optimization problems. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) Evolutionary Multi-criterion Optimization, vol. 6576 of Lecture Notes in Computer Science, pp. 254–268. Springer, Berlin. ISBN:978-3-642-19892-2 (2011)
27. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
28. Mladenovic, N., Urosevic, D., Pérez-Brito, D., García-González, C.G.: Variable neighbourhood search for bandwidth reduction. Eur. J. Oper. Res. **200**(1), 14–27. ISSN:0377-2217 (2010)
29. Pantrigo, J.J., Martí, R., Duarte, A., Pardo, E.G.: Scatter search for the cutwidth minimization problem. Ann. Oper. Res. **199**. ISSN:0254-5330 (2012)
30. Pardo, E.G., Mladenovic, N., Duarte, A., Pantrigo, J.J.: Variable formulation search for the cutwidth minimization problem. Appl. Soft Comput. **13**, 2242–2252 (2013)
31. Resende, M.G.C., Martí, R., Gallego, M., Duarte, A.: Grasp and path relinking for the max–min diversity problem. Comput. Oper. Res. **37**, 498–508 (2010)
32. Rodriguez-Tello, E., Hao, J.K., Torres-Jimenez, J.: An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem. Comput. Oper. Res. **35**(10), 3331–3346. ISSN:0305-0548 (2008)
33. Schilde, M., Doerner, K.F., Hartl, R.F., Kiechle, G.: Metaheuristics for the bi-objective orienteering problem. Swarm Intell. **3**, 179–201 (2009)
34. Ulungu, E.L.: Optimisation Combinatoire Multicritère : Détermination de l'Ensemble des solutions Efficaces et Méthodes Interactives. Ph.D. thesis, Université de Mons-Hainaut (Mons, Belgium) (1993)
35. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evolut. Comput. **3**(4), 257–271 (1999)
36. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C. (ed.) Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE), Barcelona (2002)
37. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evolut. Comput. **7**(2), 117–132 (2003)