Very large-scale neighborhood search for solving multiobjective combinatorial optimization problems

Thibaut Lust, Jacques Teghem, Daniel Tuyttens

Faculté Polytechnique de Mons, Laboratory of Mathematics & Operational Research
20, place du parc 7000 Mons (Belgium)
lust.thibaut@gmail.com

Abstract. Very large-scale neighborhood search (VLSNS) is a technique intensively used in single-objective optimization. However, there is almost no study of VLSNS for multiobjective optimization. We show in this paper that this technique is very efficient for the resolution of multiobjective combinatorial optimization problems. Two problems are considered: the multiobjective multidimensional knapsack problem and the multiobjective set covering problem. VLSNS are proposed for these two problems and are integrated into the two-phase Pareto local search. The results obtained on biobjective instances outperform the state-of-the-art results for various indicators.

1 Introduction

During the last 20 years, many heuristic methods for solving multiobjective combinatorial optimization (MOCO) problems have been proposed. From the first survey [33] in 1994 till [13] in 2002, a lot of papers have been published and this flow is still increasing. The main reason of this phenomenon is the success story of metaheuristics [15].

Effectively, it is quite difficult to determine exactly the whole set of Pareto efficient solutions X_E and the set of Pareto non-dominated points Y_N for MOCO problems. This is a $\mathcal{NP}$-Hard problem even for MOCO problems for which polynomial algorithms exist for the single-objective counterpart, such as the linear assignment problem. Therefore, there exist only few exact methods able to determine the sets X_E and Y_N and we can only expect to apply these methods for small instances and for few number of objectives. For this reason, many methods are heuristic methods which produce approximations $\widetilde{X}_E$ and $\widetilde{Y}_N$ to the sets X_E and Y_N. Due to the success of metaheuristics for single-objective CO, multiobjective metaheuristics became quickly a classic tool to tackle MOCO problems and it is presently a real challenge for the researchers to improve the results previously obtained.

If evolutionary methods [9, 12] and simple local search [13] (LS) have intensively been applied to solve multiobjective problems (MOPs), few results are

known about the use of elaborate LS techniques, as very large-scale neighborhood search (VLSNS) [1] or variable neighborhood search (VNS) [16] for solving MOPs. It is mainly because it is more natural to use a method based on a population, as we are looking for an approximation to a set. However, by embedding these evolved LS techniques into the two-phase Pareto local search (2PPLS) [28], which uses as population the set of potentially efficient solutions, we show in this paper that we can produce very effective heuristics in order to solve MOCO problems.

The paper is organized as follows. In the next section, we present VLSNS for the multiobjective multidimensional knapsack problem (MOMKP) and for the multiobjective set covering problem (MOSCP). We show after how VLSNS can be embedded into 2PPLS. We finally present the results obtained for the two MOCO problems considered, with a comparison to state-of-the-art results.

2 Very large-scale neighborhood search

With LS, the larger the neighborhood, the better the quality of the local optimum obtained is. However, by increasing the size of the neighborhood, the time to explore the neighborhood becomes higher. Therefore, using a larger neighborhood does not necessary give rise to a more effective method. If we want to keep reasonable running times while using a large neighborhood, an efficient strategy has to be implemented in order to explore the neighborhood; this is what is done in VLSNS.

VLSNS is very popular in single-objective optimization [1]. For example, the Lin-Kernighan heuristic [25], one of the best heuristics for solving the single-objective traveling salesman problem (TSP), is based on VLSNS. On the other hand, there is almost no study of VLSNS for solving MOCO problems. The only known result is the LS of Angel *et al.* [6], which integrates a dynasearch neighborhood (the neighborhood is solved with dynamic programming) to solve the biobjective TSP.

We present VLSNS for two problems: the MOMKP and the MOSCP. Starting from a current solution, called x^c, the aim of VLSNS is to produce a set of neighbors of high quality, in a reasonable time. The technique that we use for solving the MOMKP or the MOSCP is the following:

1. Identification of a set of variables candidates to be removed from x^c (set $\mathcal{O}$)
2. Identification of a set of variables, not in x^c, candidates to be added (set $\mathcal{I}$)
3. Creation of a residual multiobjective problem formed by the variables belonging to $\{\mathcal{O} \cup \mathcal{I}\}$.
4. Resolution of the residual problem: a set of potentially efficient solutions of this problem is produced. The potentially efficient solutions are then merged with the unmodified variables of x^c to produce the neighbors.

We present now how we have adapted this technique to the two MOCO problems considered in this paper.

2.1 The multiobjective multidimensionnal knapsack problem

The MOMKP consists in determining a subset of items, among n items ($i = 1, \ldots, n$) having m characteristics w_j^i ($j = 1, \ldots, m$) and p profits c_l^i ($l = 1, \ldots, p$), such that the p total profits are maximized and the m knapsack capacities W_j regarding the different characteristics are respected.

The formulation is the following:

$$\begin{aligned} \text{“max”}\ f_l(x) &= \sum_{i=1}^{n} c_l^i x_i \quad && l = 1, \ldots, p \\ \text{subject to} \sum_{i=1}^{n} w_j^i x_i &\leq W_j \quad && j = 1, \ldots, m \\ x_i &\in \{0, 1\} && i = 1, \ldots, n \end{aligned}$$

where $x_i = 1$ means that the item i is selected to be in the knapsack. It is assumed that all coefficients c_l^i, w_j^i and W_j are nonnegative.

According to the general technique previously presented to define VLSNS, the different steps to produce neighbors from a current solution x^c of the MOMKP are as follows:

1. The set $\mathcal{O}$ of items candidates to be removed will be composed of the k worst items (present in x) for the ratio R_1, defined by

$$R_1 = \frac{\sum_{l=1}^{p} \lambda_l c_l^s}{\sum_{j=1}^{m} w_j^s} \tag{1}$$

   for an item s. This ratio is simply equal to the weighted linear aggregation of the profits on the sum of the weights of the item s.
2. The set $\mathcal{I}$ of items candidates to be added will be composed of the k best items (not in x) for the ratio R_2 [30], defined by

$$R_2 = \frac{\sum_{l=1}^{p} \lambda_l c_l^s}{\sum_{j=1}^{m} \left(\frac{w_j^s}{W_j - \sum_{\substack{i=1 \\ i \notin \mathcal{O}}}^{n} w_j^i x_i + 1} \right)} \tag{2}$$

   for an item s.
   This ratio is also used in MOGLS [21], and consists in selecting items of high profit and low weight, by giving higher influence to the constraints whose the values (measured by $\sum_{\substack{i=1 \\ i \notin \mathcal{O}}}^{n} w_j^i x_i$) is close to the maximum capacity W_j.

3. A residual problem is defined, of size $(k * 2)$, and composed of the items belonging to the set $\{\mathcal{O} \cup \mathcal{I}\}$. The capacities W_j^r of the residual problem are equal to $W_j - \sum_{\substack{i=1 \\ i \notin \mathcal{O}}}^{n} w_j^i x_i$ with $j = 1, \ldots, m$.
4. To solve the residual problem, we have employed the multiobjective metaheuristic MEMOTS [26], a memetic algorithm integrating tabu search, that has already been applied to the MOMKP. MEMOTS presents some parameters that we will not tune here; we will rather take into account the conclusions obtained in [26]. We will only study the number of iterations N performed in MEMOTS to solve the residual problems.
   Once the residual problem has been solved, we merge the potentially efficient solutions of the residual problem with the unmodified variables of x^c, to obtain the neighbors.

2.2 The multiobjective set covering problem

In the MOSCP, we have a set of m items, and each item can be covered by a subset of n sets. Each set j has p costs c_l^j $(l = 1, \ldots, p)$. The MOSCP consists in determining a subset of sets, among the n sets $(j = 1, \ldots, n)$ such that all the items are covered by at least one set and that the p total costs are minimized.

The formulation is the following:

$$\begin{aligned} &\text{“min”} && f_l(x) = \sum_{j=1}^{n} c_l^j x_j && l = 1, \ldots, p \\ &\text{subject to} && \sum_{j=1}^{n} t_{ij} x_j \geq 1 && i = 1, \ldots, m \\ & && x_j \in \{0, 1\} && j = 1, \ldots, n \end{aligned}$$

with x the decision vector, formed of the binary variables x_j ($x_j = 1$ means that the set j is considered), and the binary data t_{ij} equal to 1 if the set j covers the item i and 0 otherwise.

It is assumed that all coefficients c_l^j are nonnegative.

The VLSNS that we have defined for the MOSCP is similar to the one defined for the MOMKP. However, the definition is a bit more complicated. With the MOMKP, the two sets $\mathcal{O}$ and $\mathcal{I}$ were independently defined. With the MOSCP, that cannot be anymore the case. Indeed, when we remove some sets of the current solution x^c, some items are not anymore covered. The set $\mathcal{I}$ must thus be composed of sets than can cover these items. Also, we have noticed that removing a small number k of sets (less than 4, as we will see later in Figure 3) from x^c was enough. Indeed, when we work with a larger set $\mathcal{O}$, it becomes more difficult to define the set $\mathcal{I}$ since many items become uncovered.

Therefore, as we will keep k small, for each value of k, we will create more than one residual problem. If $k = 1$, the number of residual problems will be

equal to the number of sets present in the current solution x. For each residual problem, the set $\mathcal{O}$ will be thus composed of one of the sets present in x^c.

If $k > 1$, we create a set $\mathcal{L}$ of size L ($L \geq k$) that includes the interesting sets to remove from x^c. Then, all the combinations of k sets from $\mathcal{L}$ will be considered to form the set $\mathcal{O}$. The number of residual problems will be thus equal to the number of combinations of k elements into a set of size L, that is C_k^L. The size of the set $\mathcal{I}$ will also be limited to L.

More precisely, the VLSNS works as follows:

1. If $k > 1$, the set $\mathcal{O}$ will be composed of k sets chosen among the list $\mathcal{L}$ containing the L worst sets (present in x^c) for the ratio R_3, defined by

$$R_3 = \frac{\sum_{l=1}^{p} \lambda_l c_l^s}{\sum_{i=1}^{m} t_{is}} \tag{3}$$

   for a set s. This ratio is equal to the weighted aggregation of the costs on the number of items that the set s covers.
2. The set $\mathcal{I}$ of sets candidates to be added will be composed of the L best sets (not in x) for the ratio R_4, defined by

$$R_4 = \frac{\sum_{l=1}^{p} \lambda_l c_l^s}{n_s} \tag{4}$$

   for a set s, where n_s represent the number of items that the set s covers among the items that are not anymore covered following the removals of the k preceding sets. This ratio is only computed for the sets that cover at least one of the items not anymore covered ($n_s > 0$).
3. A residual problem is defined, of size $k + L$, and composed of the sets belonging to the set $\{\mathcal{O} \cup \mathcal{I}\}$.
4. To solve the residual problem, we simply generate all the efficient solutions of the problem, with an enumeration algorithm. We then merge the efficient solutions of the residual problem with the unmodified variables of x^c, to obtain the neighbors.

3 Two-Phase Pareto Local Search

In order to be able to use VLSNS in MO, we have integrated this technique into the two-phase Pareto local search (2PPLS) [28]. As indicated by its name, the method is composed of two phases. In the first phase, an initial population of potentially efficient solutions is produced. In the second phase, the Pareto local search [6, 31] (PLS) is run from this population. PLS is a straightforward adaptation of LS to MO and only needs a neighborhood function $\mathcal{N}(x)$, which

is applied to every new potentially non-dominated solution generated. At the end, a local optimum, defined in a MO context, is obtained [31] (called a Pareto local optimum set). Therefore, to adapt 2PPLS to a MOCO problem, we have to define an initial population and a neighborhood function. As neighborhood, we will use a VLSNS.

However, comparing to the initial version of 2PPLS [28], we have adapted the method in order to be able to use different sizes of neighborhood, through a variable neighborhood search (VNS) technique [16]. This is particularly useful in the case of the MOSCP. Indeed, for the MOSCP, the VLSNS of size $(k+1)$ will not necessary produce the neighbors obtained with the VLSNS of size k (as the set $\mathcal{I}$ depends on the set $\mathcal{O}$ of size k). On the other hand, for the MOMKP, the VLSNS of size $(k+1)$ will also produce the neighbors generated by the VLSNS of size k. Therefore, for the MOMKP, the higher k, the better the results are, and a VNS technique is not worthwhile.

The pseudo-code of 2PPLS with VNS is given by the Algorithm 1.

The method needs four parameters: an initial population P_0, the size of the smallest ($k_{\min}$) (respectively largest ($k_{\max}$)) neighborhood structure, and the different neighborhood functions $\mathcal{N}_k(x)$ for each $k \in \mathbb{Z} : k_{\min} \leq k \leq k_{\max}$.

The method starts with the population P composed of potentially efficient solutions given by the initial population P_0. The neighborhood structure initially used is the smallest ($k = k_{\min}$). Then, considering $\mathcal{N}_k(x)$, all the neighbors p' of each solution p of P are generated. If a neighbor p' is not weakly Pareto dominated ($\not\preceq_P$) by the current solution p, we try to add the solution p' to the approximation $\widetilde{X}_E$ of the efficient set, which is updated with the procedure `AddSolution`. This procedure is not described in this paper but simply consists of updating an approximation $\widetilde{X}_E$ of the efficient set when a new solution p' is added to $\widetilde{X}_E$. This procedure has four parameters: the set $\widetilde{X}_E$ to actualize, the new solution p', its evaluation $f(p')$ and a boolean variable called *Added* that returns $True$ if the new solution has been added and $False$ otherwise. If the solution p' has been added to $\widetilde{X}_E$, *Added* is true and the solution p' is added to an auxiliary population P_a, which is also updated with the procedure `AddSolution`. Therefore, P_a is only composed of new potentially efficient solutions. Once all the neighbors of each solution of P have been generated, the algorithm starts again, with P equal to P_a, until $P = P_a = \varnothing$. The auxiliary population P_a is used such that the neighborhood of each solution of P is explored, even if some solutions of P become dominated following the addition of a new solution to P_a. Thus, sometimes, neighbors are generated from a dominated solution.

In the case of $P = P_a = \varnothing$, the set $\widetilde{X}_E$ obtained is a Pareto local optimum set according to $\mathcal{N}_k$, and cannot be thus improved with $\mathcal{N}_k$. We then increase the size of the neighborhood ($k \leftarrow k+1$), and apply again PLS with this larger neighborhood.

Please note that, in general, a solution Pareto local optimum for the neighborhood k is not necessary Pareto local optimum for the neighborhood $(k-1)$. That is why, after considering a larger neighborhood, we always restart the search with the smallest neighborhood structure.

After the generation of a Pareto local optimum set according to $\mathcal{N}_k$, the population P used with the neighborhood $\mathcal{N}_{k+1}$ is $\widetilde{X}_E$, without considering the solutions of $\widetilde{X}_E$ that could already be Pareto local optimal for $\mathcal{N}_{k+1}$.

The method stops when a Pareto local optimum set has been found, according to all the neighborhood structures considered.

Algorithm 1 `2PPLS` with `VNS`

Parameters $\downarrow$: an initial population P_0, neighborhood functions $\mathcal{N}_k(x)$, $k_{\min}$, $k_{\max}$.
Parameters $\uparrow$: an approximation $\widetilde{X}_E$ of the efficient set X_E.

--| Initialization of $\widetilde{X}_E$ and a population P with the initial population P_0
$\widetilde{X}_E \leftarrow P_0$
$P \leftarrow P_0$
--| Initialization of an auxiliary population P_a
$P_a \leftarrow \varnothing$
--| Initialization of the neighborhood structure
$k \leftarrow k_{\min}$
repeat
  while $P \neq \varnothing$ **do**
    --| Generation of all neighbors p' of each solution $p \in P$
    for all $p \in P$ **do**
      for all $p' \in \mathcal{N}_k(p)$ **do**
        if $f(p) \not\preceq_P f(p')$ **then**
          `AddSolution`($\widetilde{X}_E \updownarrow$, $p' \downarrow$, $f(p') \downarrow$, *Added* $\uparrow$)
          if *Added* $=$ *true* **then**
            `AddSolution`($P_a \updownarrow$, $p' \downarrow$, $f(p') \downarrow$)
  if $P_a \neq \varnothing$ **then**
    --| P is composed of the new potentially efficient solutions
    $P \leftarrow P_a$
    --| Reinitialization of P_a
    $P_a \leftarrow \varnothing$
    --| We start again with the smallest neighborhood structure
    $k \leftarrow k_{\min}$
  else
    --| We use a larger neighborhood structure
    $k \leftarrow k + 1$
    --| We use as population the solutions of X_E that are not Pareto local optimum for $\mathcal{N}_k(x)$
    $P \leftarrow \widetilde{X}_E \backslash \{x \in \widetilde{X}_E \mid x \text{ Pareto local optimum for } \mathcal{N}_k(x)\}$
until $k > k_{\max}$

4 Results

We will mainly focus the presentation of the results on showing that the use of VLSNS gives a competitive method. We will not have enough space to show all

the results of the parameters' analysis of our methods. We will however point out the most important conclusions obtained.

To compare the quality of the approximations generated, we use four indicators: the hypervolume $\mathcal{H}$ (to be maximized) [35], the average distance D_1 and maximal distance D_2 (to be minimized) [10] between the points of Y_N and the points of $\widetilde{Y}_N$, by using the Euclidean distance, and the proportion P_{Y_N} (to be maximized) of non-dominated points generated.

The last three indicators can be used since we have mainly considered instances of the MOMKP and the MOSCP for which we were able to generate exactly Y_N with the ϵ-constraint method [24].

We will only present some results for biobjective instances. The computer used for the experiments is a Pentium IV with 3 GHz CPUs and 512 MB of RAM. Twenty runs of the algorithms are performed for each instance. To compute the ratios R_1, R_2, R_3 and R_4 used in the VLSNS, random generations of weight sets are considered.

4.1 The multiobjective multidimensional knapsack problem

The MOMKP is one of the most studied MOCO problem. This problem has been used by many different groups of authors in order to test the performances of new multiobjective metaheuristics. A survey of this problem can be found in [27].

As many authors previously did, we use the instances of Zitzler and Thiele [37] with 250, 500 or 750 items, two objectives and two constraints (the instances are called 250-2, 500-2 and 750-2).

Adaptation of 2PPLS To adapt 2PPLS to the resolution of the MOMKP, we first need to define how the initial population is generated.

As we did not find any efficient published implementations of heuristics solving the single-objective MKP, we have implemented a simple greedy heuristic. To create a new solution, the items are added to the knapsack one by one. At each iteration, the item s that maximizes the ratio (R_2) (see section 2.1) is selected.

The greedy procedure is run with 100 weight sets uniformly generated in $[0, 1]$. We have studied the influence of the number of weight sets, but this number has a low influence on the quality of the results, given the quality of the neighborhood used in PLS.

Influence of the size of the neighborhood We show now some results about the influence of the size of the neighborhood k. For the MOMKP, no VNS is used.

In Figure 1, we show the evolution of P_{Y_N} and the running time according to k for the 500-2 instance. We vary the values of k from 4 to 20. We use three different numbers of iterations for MEMOTS (the method used to solve the residual problems defined in the VLSNS): $N = 100$, $N = 200$ and $N = 400$. We see that for small values of k, P_{Y_N} is more or less equal no matter the number of iterations. From k equal to 10, it is clear that we obtain better results if

N is higher. On the other hand, the running time is bigger when N is higher, but still evolves more or less linearly according to k. An interesting behavior is pointed out by the figure showing the evolution of P_{Y_N} according to k. From k equal to about 16 and for a number of iterations N equal to 100 or 200, there is a deterioration of P_{Y_N} while the running time is increasing. It means that the number of iterations performed in MEMOTS is not high enough to solve the residual problems, and that therefore the quality of the approximations obtained for the residual problems is not good enough to improve P_{Y_N}. Fixing good values for k and N should be thus carefully done since these two values have to be increased at the same time if we want to improve the quality of the results.

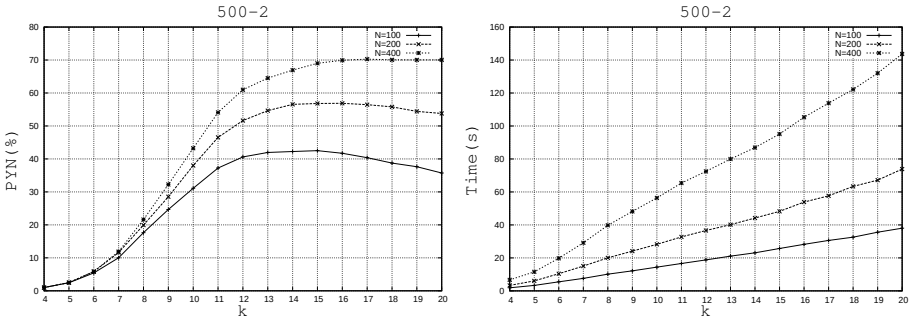

Fig. 1. Influence of k for the MOMKP.

Comparison to state-of-the-art results We have obtained the following results for the 250-2, 500-2, 750-2 instances: SPEA [37] (30 runs); SPEA2 [36] (30 runs, but only for the 750-2 instance); MOGLS00 [19] (20 runs); MOGLS04 [19] (20 runs, different than MOGLS00 since obtained with the MOMHLib++ library [18]); PMA [22] (20 runs); IMMOGLS [17] (20 runs); MOGTS [7] (1 run); GRASP [34] (1 run); MOTGA [4] (20 runs); PATH-RELINKING [8] (30 runs); GPLS [2] (30 runs); mGPLS [3] (30 runs); iGPLS [3] (30 runs). These results have been obtained from web sites or directly from the different authors. We see that we have obtained quite a lot of results. It is only a pity that Gomes da Silva *et al.* [11] did not send us their results.

Thanks to these results, we have generated a reference set, called ALL, formed by merging the potentially non-dominated points obtained by all the runs of all algorithms, which gives a very high quality set.

However, we show that is possible to obtain better results than this set, for the indicators considered in this paper, in reasonable times, with the VLSNS integrated into 2PPLS. We have carefully selected the parameters such that we obtain better or equal results than the reference set ALL for all indicators. The parameters are the following:

- 250-2: $k = 9$ and $N = 200$.

– 500-2: $k = 15$ and $N = 100$.
– 750-2: $k = 9$ and $N = 100$.

The results for 2PPLS are given in Table 1. $|PE|$ gives the number of potentially efficient solutions generated. We see that is possible to obtain better or equal values for all indicators, in very reasonable running times: 7s for 250-2, 23s for 500-2 and 18s for 750-2.

Table 1. Comparison between 2PPLS and ALL based on the indicators.

Instance	Algorithm	$\mathcal{H}(10^7)$	D_1	D_2	$\|PE\|$	$P_{Y_N}(\%)$	Time(s)
250-2	2PPLS	**9.8690**	**0.029**	**2.680**	482.10	**68.05**	7.27
	ALL	**9.8690**	0.069	2.838	376.00	31.87	/
500-2	2PPLS	**40.7873**	**0.025**	**1.976**	1131.00	**42.85**	23.43
	ALL	40.7850	0.081	2.045	688.00	5.51	/
750-2	2PPLS	**89.3485**	**0.076**	**1.494**	1558.90	**4.15**	17.52
	ALL	89.3449	0.092	**1.494**	996.00	0.99	/

We compare now MEMOTS and 2PPLS, for different running times. In [26], we have showed that MEMOTS was a performing method since this method gives better values than MOGLS [19] and PMA [22] for different indicators.

The results are presented in Figure 2, for the 250-2 and 750-2 instances, where the evolutions of D_1 and P_{Y_N} according to the running time are showed. The running time of 2PPLS is controlled by k and N, while the running of MEMOTS is controlled by the number of iterations performed.

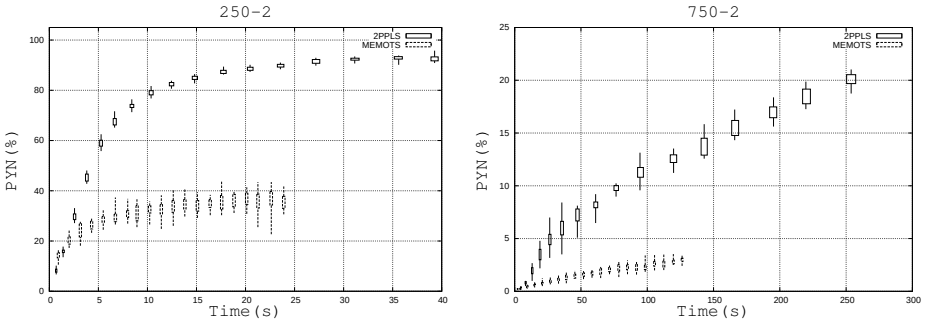

Fig. 2. Comparison of MEMOTS and 2PPLS: evolution of D_1 and P_{Y_N} according to the running time.

We see that except with small running times, the results obtained with 2PPLS are better than with MEMOTS. With 2PPLS, we can generate, for the 250-2 instance, about 90% of the non-dominated points, for the 500-2 instance,

about 70% and for the 750-2 instance, about 20%, in reasonable running times. The running times are remarkable since, for example, Mavrotas *et al.* [29] can generate for the 250-2 instance, 81% of Y_N, but in about 30 minutes, while we can attain this result in about 15s. Also, for this same instance, they need 21 hours to generate 93% of Y_N, while we only need 42 seconds. We are thus 1800 times faster!

4.2 The multiobjective set covering problem

The MOSCP has not received as much attention as the MOMKP. To our knowledge, only two groups of authors have tackled this problem. Jaszkiewicz was the first one, in 2001 [20], with the adaptation of the Pareto memetic algorithm (PMA). In 2006, Prins *et al.* [32] have also tackled this problem, by using a two-phase heuristic method (called TPM) using primal-dual Lagrangian relaxations to solve different single-objective SCPs.

We use the same instances than those authors considered, from the size 600x60 (600 sets, 60 items) to the size 1000x200. Seven instances have been considered, and for each size instance, four different kinds of objectives (a, b, c and d) are defined [14]. More informations about these instances can be found in [20, 32].

Adaptation of 2PPLS As initial population, we use a good approximation of the supported efficient solutions. These solutions can be generated by resolution of weighted sum single-objective problems obtained by applying a linear aggregation of the objectives. To generate the different weight sets, we have used the dichotomic method of Aneja and Nair [5]. Depending on the difficulty of the instances, we have used an exact solver (the free MILP solver lp_solve) or a heuristic (the meta-raps heuristic developed by Lan *et al.* [23] whose the code has been published) to solve the single-objective SCPs.

Influence of the size of the neighborhood We show now some results about the influence of the size of the neighborhood k and the length L of the lists $\mathcal{L}$ and $\mathcal{I}$. For the MOSCP, VNS is used.

In Figure 3, we show the evolution of P_{Y_N} and the running time according to L for the 61a instance (600x60), for different values of k. We vary the values of L from 1 to 9 and k from 1 to 4. We see that, of course, increasing the values of L or k allows to obtain better quality results. The best improvement is when k is moved from 1 to 2 for values of L superior to 5. On the other hand, using $k = 3$ or $k = 4$ in place of $k = 2$ does not give impressive improvements, whatever the value of L.

Concerning the running time, as we use an exact enumeration algorithm to solve the residual problems, we see that its evolution is exponential. Using $k = 3$ or $k = 4$ with $L \geq 8$ becomes very time-consuming.

Therefore, for the comparison to state-of-the-art results, we will use $k = 2$ and $L = 9$.

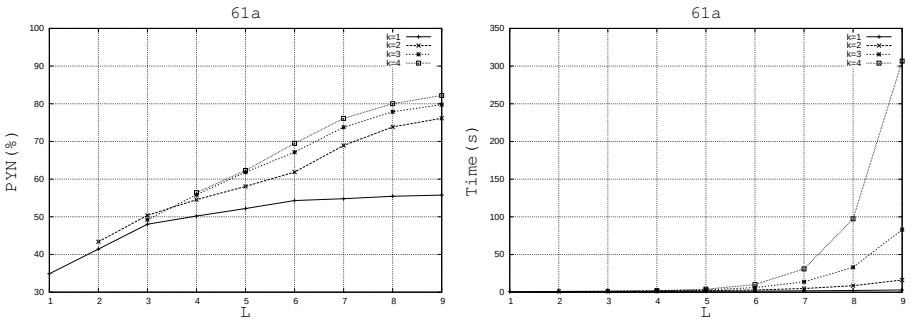

Fig. 3. Influence of k and L for the MOSCP.

Comparison to state-of-the-art results In Table 2, we compare the results obtained with 2PPLS to the results of TPM and PMA. We only show the results for the instance 81 of size 800x80, for the four different kinds of objectives (a, b, c and d) . We see that we obtain better results for the indicators considered. For these instances, we always obtain a value of P_{Y_N} superior to 50%, in less than 41s.

Table 2. Comparison between 2PPLS, TPM and PMA based on the indicators.

Instance	Algorithm	$\mathcal{H}(10^5)$	D_1	D_2	$P_{Y_N}(\%)$	$\|PE\|$	Time(s)
	2PPLS	**1804.7490**	**0.094**	**1.164**	**58.87**	351.65	31.79
81a	TPM	1787.6098	1.957	9.729	10.61	64.00	/
	PMA	1777.7066	1.253	3.792	0.61	173.80	/
	2PPLS	**2295.7315**	**0.141**	**1.176**	**56.20**	272.25	30.97
81b	TPM	2277.4616	1.690	6.398	11.30	55.00	/
	PMA	2267.9781	1.197	6.756	3.64	138.10	/
	2PPLS	**114.3475**	**0.711**	**3.820**	**71.43**	11.55	19.5
81c	TPM	114.0509	3.596	15.592	35.71	9.00	/
	PMA	110.0607	21.113	61.595	0.00	1.50	/
	2PPLS	**169.5468**	**0.073**	**0.875**	**91.67**	11.00	40.58
81d	TPM	169.3999	2.827	23.631	75.00	10.00	/
	PMA	160.0092	10.640	42.617	0.00	6.60	/

We have obtained similar results for all the other instances, except for the instances 62c and 62d for which TPM obtains slightly better values for some indicators. For example, for the bigger instances 201a and 201b (1000x200), for which we were not able to generate Y_N, we have obtained that, in average, more than 80% of the points generated by 2PPLS dominate the points generated by PMA.

The comparison of the average running times for solving the different instances is given in Table 3. We see that the running times of 2PPLS are less or

equal than the running times of TPM and PMA (coming however from slower computers).

Table 3. Comparison of the average running times of 2PPLS, TPM and PMA (in seconds).

Instance	Dimension	2PPLS	TPM	PMA
61	60 x 600	20.41	20.27	132.4
62	60 x 600	9.54	20.35	98.8
81	80 x 800	30.71	30.22	165.9
82	80 x 800	16.41	30.27	148.4
101	100 x 1000	26.48	50.10	311.7
102	100 x 1000	23.75	50.41	282.1
201	200 x 1000	60.36	70.81	686.8

5 Conclusion

We have shown through this paper the effectiveness of approaches based on VLSNS to solve MOCO problems. The integration of VLSNS into 2PPLS allowed to obtain new state-of-the-art results for two MOCO problems: the MOMKP and MOSCP. However, these results are only for biobjective instances. We have performed some tests for three-objective instances of the MOMKP. The results obtained were of high quality too, but with a very high running time (more than 7h). Indeed, the number of potentially non-dominated points generated was very high (more than 60000 points for an instance with only 250 items). Therefore, in order to make this new approach more practical, the integration of the preferences of the decision maker into the Pareto dominance will be necessary to tackle instances with more than two objectives.

References

1. R.K. Ahuja, Ö. Ergun, J.B. Orlin, and A.P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Appl. Math.*, 123(1-3):75–102, 2002.
2. A. Alsheddy and E.P.K. Tsang. Guided Pareto local search and its application to the 0/1 multi-objective knapsack problems. In *Proceedings of the eighth metaheuristic international conference (MIC'09)*, Hamburg, 2009.
3. A. Alsheddy and E.P.K. Tsang. Guided Pareto local search based frameworks for Pareto optimization. In *Proceedings of the WCCCI IEEE World Congress on Computational Intelligence*, Barcelona, 2010.
4. M.J. Alves and M. Almeida. MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem. *Computers & Operations Research*, 34:3458–3470, 2007.
5. Y.P. Aneja and K.P.K. Nair. Bicriteria transportation problem. *Management Science*, 25:73–78, 1979.

6. E. Angel, E. Bampis, and L. GourvÃ¨s. A dynasearch neighborhood for the bicriteria traveling salesman problem. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, pages 153–176. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, Berlin, 2004.
7. V. Barichard and J.K. Hao. An empirical study of tabu search for the MOKP. In *Proceedings of the First International Workshop on Heuristics*, volume 4, pages 47–56, China, 2002. Series of Information & Management Sciences.
8. R.P. Beausoleil, G. Baldoquin, and R.A. Montejo. Multi-start and path relinking methods to deal with multiobjective knapsack problems. *Annals of Operations Research*, 157:105–133, 2008.
9. C.A. Coello Coello, D.A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.
10. P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.
11. C. Gomes da Silva, J. Clímaco, and J. R. Figueira. Scatter search method for the bi-criteria multi-dimensional {0,1}-knapsack problem using surrogate relaxation. *Journal of Mathematical Modelling and Algorithms*, 3(3):183–208, 2004.
12. K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, New-York, 2001.
13. M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization – State of the Art Annotated Bibliographic Surveys*, volume 52, pages 369–444. Kluwer Academic Publishers, Boston, MA, 2002.
14. X. Gandibleux, D. Vancoppenolle, and D. Tuyttens. A first making use of GRASP for solving MOCO problems. In *14th International Conference in Multiple Criteria Decision-Making*, Charlottesville, 1998.
15. F. Glover and G. Kochenberger. *Handbook of Metaheuristics*. Kluwer, Boston, 2003.
16. P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449 – 467, 2001.
17. H. Ishibuchi and T. Murada. A multi-objective genetic local search algorithm and its application to flow shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 28(3):392–403, 1998.
18. A. Jaszkiewicz. Experiments done with the MOMHLIB: http://www-idss.cs.put.poznan.pl/ jaszkiewicz/momhlib/. Technical report, Institute of Computing Science, Poznań University of Technology, 2000.
19. A. Jaszkiewicz. On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem—A Comparative Experiment. Technical Report RA-002/2000, Institute of Computing Science, Poznań University of Technology, Poznań, Poland, July 2000.
20. A. Jaszkiewicz. A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the Pareto memetic algorithm. Technical Report RA-003/01, Institute of Computing Science, Poznań University of Technology, Poznań, Poland, 2001.
21. A. Jaszkiewicz. On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem—A Comparative Experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412, August 2002.

22. A. Jaszkiewicz. On the Computational Efficiency of Multiple Objective Metaheuristics. The Knapsack Problem Case Study. *European Journal of Operational Research*, 158(2):418–433, October 2004.
23. G. Lan, G.W. DePuy, and G.E. Whitehouse. An effective and simple heuristic for the set covering problem. *European Journal of Operational Research*, 176(3):1387 – 1403, 2007.
24. M. Laumanns, L. Thiele, and E. Zitzler. An adaptative scheme to generate the Pareto front based on the epsilon-constraint method. Technical Report 199, Technischer Bericht, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2004.
25. S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.
26. T. Lust and J. Teghem. Memots: a memetic algorithm integrating tabu search for combinatorial multiobjective optimization. *RAIRO: Operations Research*, 42(1):3–33, 2008.
27. T. Lust and J. Teghem. The multiobjective multidimensionnal knapsack problem: a survey and a new approach. Technical Report arXiv:1007.4063v1, arXiv, 2010.
28. T. Lust and J. Teghem. Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16(3):475–510, 2010.
29. G. Mavrotas, J.R. Figueira, and K. Florios. Solving the bi-objective multi-dimensional knapsack problem exploiting the concept of core. *Applied Mathematics and Computation*, 215(7):2502–2514, 2009.
30. Z. Michalewicz and J. Arabas. Genetic algorithms for the 0/1 knapsack problem. In Z.W. Ras and M. Zemankova, editors, *Methodologies for Intelligent Systems Conference (ISMIS)*, pages 134–143, Berlin, 1994.
31. L. Paquete, M. Chiarandini, and T. Stützle. Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, pages 177–199, Berlin, 2004. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535.
32. C. Prins, C. Prodhon, and R. Wolfler Calvo. Two-phase method and lagrangian relaxation to solve the bi-objective set covering problem. *Annals OR*, 147(1):23–41, 2006.
33. E.L. Ulungu and J. Teghem. Multiobjective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, 3:83–104, 1994.
34. D.S. Vianna and J.E.C. Arroyo. A GRASP algorithm for the multi-objective knapsack problem. In *QEST '04: Proceedings of the The Quantitative Evaluation of Systems, First International Conference*, pages 69–75, Washington, DC, USA, 2004. IEEE Computer Society.
35. E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.
36. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.
37. E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.