Two-phase Pareto local search to solve the biobjective set covering problem

Thibaut Lust
LIP6-CNRS
Université Pierre et Marie Curie
Paris, France
thibaut.lust@lip6.fr

Daniel Tuyttens
Laboratory of Mathematics and Operational Research
University of Mons - Faculté Polytechnique de Mons
Mons, Belgium
daniel.tuyttens@umons.ac.be

***Abstract*—In this paper, we study the biobjective version of the set covering problem. To our knowledge, this problem has only been addressed in two papers before, and with heuristic methods. We propose a new heuristic, based on the two-phase Pareto local search, with the aim of generating a good approximation of the Pareto efficient solutions. In the first phase of this method, the supported efficient solutions or a good approximation of these solutions is generated. Then, a neighborhood embedded in the Pareto local search is applied to generate non-supported efficient solutions. In order to get high quality results, two elaborate local search techniques are considered: a very large-scale neighborhood search and a variable neighborhood search. We intensively study the parameters of these two techniques. We compare our results with state-of-the-art results and we show that with our method, better results are obtained for different indicators.**

***Keywords*-Multiobjective optimization ; combinatorial optimization ; metaheuristics ; set covering problem ; very large-scale neighborhood search ; variable neighborhood search;**

I. Introduction

We consider in this paper the biobjective set covering problem (BOSCP). In the BOSCP, we have a set of m rows (or items), and each row can be covered by a subset of n columns (or sets). Each column j has two costs c_l^j $(l = 1, 2)$. The BOSCP consists in determining a subset of columns, among the n columns $(j = 1, \ldots, n)$ such that all the rows are covered by at least one column and that the two total costs are minimized.

More precisely, the biobjective set covering problem is defined as follows:

$$\text{(BOSCP)} \begin{cases} \text{“min”} & z_l(x) = \sum_{j=1}^{n} c_l^j x_j & l = 1, 2 \\ \text{s.t.} & \sum_{j=1}^{n} t_{ij} x_j \geq 1 & i = 1, \ldots, m \\ & x_j \in \{0, 1\} & j = 1, \ldots, n \end{cases}$$

with n the number of columns, m the number of rows, x the decision vector, formed of the binary variables x_j ($x_j = 1$ means that the column j is in the solution), c_l^j the cost l of the column j and the binary data t_{ij} equal to 1 if the column j covers the row i and equal to 0 otherwise. It is assumed that all coefficients c_l^j are nonnegative integer.

The data associated to the BOSCP are thus a cost matrix of size (n x 2) and a covering matrix of size (m x n).

We denote by $\mathcal{X}$ a feasible set in decision space, defined by $\mathcal{X} = \left\{x \in \{0,1\}^n | \{\sum_{j=1}^{n} t_{ij} x_j \geq 1, \forall\, i = 1, \ldots, m\}\right\}$. The corresponding feasible set in the objective space is called $\mathcal{Z}$ and is defined by $\mathcal{Z} = z(\mathcal{X}) = \left\{(z_1(x), z_2(x)),\, \forall\, x \in \mathcal{X}\right\} \subset \mathbb{N}^2$.

Due to the typically conflictive objectives, the notion of optimal solution does not exist anymore for multiobjective problems. However, based on the *dominance relation of Pareto* (see Definition 1), the notion of optimal solution can be replaced by the notion of *efficient* (or Pareto optimal) *solution* (see Definition 2).

Definition 1: A vector $z \in \mathcal{Z}$ dominates a vector $z' \in \mathcal{Z}$ if, and only if, $z_l \leq z'_l$, $\forall\, l \in \{1, \ldots, p\}$, with at least one index l for which the inequality is strict. We denote this dominance relation by $z \prec z'$.

Definition 2: A feasible solution $x \in \mathcal{X}$ is efficient if there does not exist any other solution $x' \in \mathcal{X}$ such that $z(x') \prec z(x)$. The image of an efficient solution in objective space is called a non-dominated point.

In the following, we will denote by $\mathcal{X}_E$ the set of all efficient solutions (the efficient set) and by $\mathcal{Z}_N$ the image of $\mathcal{X}_E$ in objective space (the Pareto front).

In the case of multiobjective combinatorial optimization (MOCO) problems, we can distinguish two types of efficient solutions: the supported efficient solutions and the non-supported efficient solutions. The supported efficient solutions are optimal solutions of weighted single-objective problems $(\min \sum_{l=1}^{p} \lambda_l z_l(x)$ s.t. $x \in \mathcal{X})$ where $\lambda \in \mathbb{R}_+^p$ is a weight vector with all positive components λ_l, $\forall\, l = 1, \ldots, p$. The non-supported efficient solutions are efficient solutions that are not supported.

As the single-objective version of the SCP is $\mathcal{NP}$-Hard, the BOSCP is $\mathcal{NP}$-Hard too. Therefore, our aim is to generate a good approximation of the efficient set.

Although evolutionary methods have intensively been applied to solve MOCO problems, few results are known about the use of elaborate local search (LS) techniques, such as very large-scale neighborhood search (VLSNS) [1] and variable neighborhood search (VNS) [2]. 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, we will show in this paper that by embedding these evolved LS techniques into the two-phase Pareto local search (2PPLS) [3], which uses as population the set of potentially efficient solutions, we can produce a very effective heuristic. 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 [4] (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 efficient solution generated. At the end, a local optimum, defined in a MO context, is obtained [4] (called a Pareto local optimum set). Therefore, to adapt 2PPLS to a MOCO problem, we only have to define an initial population and a neighborhood function.

The MOSCP did not receive as much attention as other classic MOCO problems, like the MO multidimensional knapsack problem [5] (MOMKP) or the MO traveling salesman problem [6] (MOTSP). To our knowledge, only two groups of authors have tackled this problem. Jaszkiewicz was the first one, in 2003 [7], with the adaptation of the Pareto memetic algorithm (PMA). In 2006, Prins *et al.* [8] 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 present in this work a new strategy to escape from a Pareto local optimal set, based on the variable neighborhood search (VNS) technique [2]: once a Pareto local optimum set has been found according to a neighborhood, we increase the size of the neighborhood in order to generate new potentially efficient solutions and to escape from the Pareto local optimum set.

The contributions of this paper are two folds. First, we present a metaheuristic method in order to solve MOCO problems, that includes at the same time VLSNS and VNS techniques (section 3 and 4). Secondly, we demonstrate the efficiency of the method through the biobjective set covering problem (BOSCP) (section 2), for which we present a new heuristic (section 5) and new state-of-the-art results (section 6).

II. Two-phase Pareto local search

The new heuristic, in order to solve the BOSCP, is based on the two-phase Pareto local search (2PPLS), developed by Lust and Teghem [3], mainly composed of two phases: generation of an initial population and application of PLS [4] from this population. We have however integrated the VNS technique in order to escape from a Pareto local optimum set.

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

The method needs four entries: an initial population P_0, the sizes of the different neighborhood structures to order

Algorithm 1 `2PPLS` with `VNS`

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

--| Initialization of $\widetilde{\mathcal{X}}_E$ and a population P with the initial population P_0
$\widetilde{\mathcal{X}}_E \leftarrow P_0$
$P \leftarrow P_0$
--| Initialization of an auxiliary population P_a
$P_a \leftarrow \varnothing$
--| Initialization of the neighborhood structure
$k \leftarrow k_{\min}$
--| Initialization of $k(x)$: the maximal size of the neighborhood that has been explored from a solution x
for all $x \in \widetilde{\mathcal{X}}_E$ **do**
  $k(x) = k_{\min}$
--| PLS with VNS
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 $z(p) \not\preceq z(p')$ **then**
          `AddSolution`($\widetilde{\mathcal{X}}_E \updownarrow$, $p' \downarrow$, $z(p') \downarrow$, $Added \uparrow$, $k \uparrow$)
          if $Added = true$ **then**
            `AddSolution`($P_a \updownarrow$, $p' \downarrow$, $z(p') \downarrow$)
  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 $\widetilde{\mathcal{X}}_E$ that are not already Pareto local optimum for $\mathcal{N}_k(x)$
    $P \leftarrow \{x \in \widetilde{\mathcal{X}}_E \,|\, k(x) < k\}$
until $k > k_{\max}$

them from the smallest ($k_{\min}$) to the largest ($k_{\max}$) one, 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 ($k = k_{\min}$). To each solution $x \in \widetilde{\mathcal{X}}_E$, we also associate the value $k(x)$. This function gives for a solution x the maximal size of the neighborhood that has been explored from the solution (for example, if $k(x) = 3$, that means that the neighborhoods of size $k = 1, 2$ and 3 have been explored from the solution). This function will help us to avoid to explore the neighborhood of a solution if it has already been explored before. Then, PLS is started and considering $\mathcal{N}_k(x)$, all the neighbors p' of each solution p of P are generated. If a neighbor p' is not weakly dominated by the current solution p (that is $z(p) \not< z(p')$ and $z(p) \neq z(p')$), we try to add the solution p' to the approximation $\widetilde{\mathcal{X}}_E$ of the efficient set, which is updated with the procedure `AddSolution`. This procedure is not described in this paper but simply consists in updating an approximation $\widetilde{\mathcal{X}}_E$ of the efficient set when a new solution p' is added to $\widetilde{\mathcal{X}}_E$. This procedure has five parameters: the set $\widetilde{\mathcal{X}}_E$ to actualize, the new solution p', its evaluation $z(p')$, an optional boolean variable called $Added$ that returns $True$ if the new solution has been added and $False$ otherwise, and another optional variable k. This last variable allows to update the value of the function $k(p')$ associated to the solution p'. Then, if the solution p' has been added to $\widetilde{\mathcal{X}}_E$, $Added$ 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 from $\mathcal{N}_k$ 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 $\widetilde{\mathcal{X}}_E$ obtained is a Pareto local optimum set according to $\mathcal{N}_k$, and cannot be improved with $\mathcal{N}_k$. We thus 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 $\widetilde{\mathcal{X}}_E$, without considering the solutions of $\widetilde{\mathcal{X}}_E$ that could already be Pareto local optimal for $\mathcal{N}_{k+1}$, that is the solutions of $\widetilde{\mathcal{X}}_E$ such that $k(x) < (k+1)$.

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

III. Adaptation of 2PPLS to the BOSCP

In this section, we present how 2PPLS has been adapted to the BOSCP. We first present how the initial population has been generated. We expose then the VLSNS considered in PLS, method used as second phase in 2PPLS.

A. Initial population

The initial population is composed of a good approximation of the supported efficient solutions. These solutions can be generated by solving single-objective problems obtained by applying a weighted sum of the objectives. To generate the different weight sets, we have used the dichotomic method of Aneja and Nair [9]. This method consists in generating all the weight sets necessary to get the supported efficient solutions that are located on the vertex set of the convex hull of $\mathcal{Z}$. The weight sets are based on the computation of normal vectors to two consecutive supported non-dominated points.

For solving the single-objective SCP obtained, we have considered two different methods:

- The free mixed integer linear programming (MILP) solver `lp_solve`.
- An efficient heuristic developed by Lan *et al.* in 2007 [10], based on the `Meta-Raps` metaheuristic, whose the code has been published.

With the exact `lp_solve` solver, an exact minimal set of the extreme supported efficient solutions can be generated. However, for the biggest instances of the BOSCP, we will see that using this method can be time-consuming. That is why the heuristic based on the `Meta-Raps` metaheuristic has also been considered.

B. Very Large-Scale Neighborhood Search

With LS (and therefore PLS), 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. It is important to point out that VLSNS is not only a heuristic that uses a large neighborhood (furthermore the definition of large is imprecise), but essentially a heuristic that uses an efficient strategy to explore a large neighborhood.

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 general technique that we use for solving the BOSCP 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 $\{\mathcal{O} \cup \mathcal{I}\}$, and the constraints not anymore fulfilled (that is the items not anymore covered)
4) Resolution of the residual problem: a set of potentially efficient solutions of this problem is produced. The potentially efficient solutions of the residual problem are then merged with the unmodified variables of x^c to produce the neighbors.

We present now how we have adapted the VLSNS to the BOSCP. Before giving more details about the VLSNS, we point out two particularities related to the BOSCP:

1) When we remove columns from a solution, some rows are not anymore covered. The set $\mathcal{I}$ must thus be composed of columns that can cover these rows.
2) We have noted that removing a small number k of columns (less than 4) 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 rows become uncovered. Moreover, the VLSNS always starts from a solution of good quality, and only some small modifications of these solutions are needed to produce new potentially efficient solutions. 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 columns present in the current solution x. For each residual problem, the set $\mathcal{O}$ will be thus composed of one of the columns present in x^c. If $k > 1$, we create a set $\mathcal{L}$ of size L ($L \geq k$), that includes the columns that are candidates to be removed from x^c (these columns are identified by the ratio R_1^s defined in the following). Then, all the combinations of k columns 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 $\mathcal{O}$ is composed of one of the columns present in x^c. If $k > 1$, the set $\mathcal{O}$ will be composed of k columns chosen among the list $\mathcal{L}$ (all combinations are tested) containing the L worst columns (present in x^c) for the ratio R_1^s, defined by

$$R_1^s = \frac{\sum_{l=1}^{2} \lambda_l c_l^s}{\sum_{i=1}^{m} t_{is}} \quad (1)$$

for a column s. This ratio is equal to the weighted aggregation of the costs on the number of rows that the column s covers (the smaller the better). The weight set λ is randomly generated.

2) The set $\mathcal{I}$ of columns candidates to be added will be composed of the L best columns (not in x^c) for the ratio R_2^s, defined by

$$R_2^s = \frac{\sum_{l=1}^{2} \lambda_l c_l^s}{n_s} \quad (2)$$

for a column s, where n_s represent the number of rows that the column s covers among the rows that are not anymore covered when the k columns have been removed. This ratio is only computed for the columns that cover at least one of the rows not anymore covered ($n_s > 0$). The weight set λ is randomly generated.

3) A residual problem is defined, of size $k+L$, composed of the columns belonging to the set $\{\mathcal{O} \cup \mathcal{I}\}$ and of the items not anymore covered.
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.

IV. Results

A. Data and reference sets

We use the same instances as Jaszkiewicz [7] and Prins *et al.* [8] considered, from the size 100x10 (100 columns, 10 rows) to the size 1000x200 (1000 columns, 200 rows). For each size instance, four different kinds of objectives A, B, C and D are defined. In the case of instances of type A, the costs of each objective are randomly generated. For the type B, the costs of the first objective are randomly generated and the ones of the second objective are made dependent in the following way:

$$c_2^j = c_1^{n-j+1} \ \forall j = 1, \ldots, n$$

For the type C, a subdivision of the columns into several subsets is done for each objective (for each objective, the subdivision is different). Then, all the columns from one subset have the same cost for the objective corresponding to the subdivision. Finally, the instances of type D combine characteristics of the sets B and C.

B. Measuring the quality of the approximations

We have used the following unary indicators:

- The hypervolume $\mathcal{H}$ (to be maximized) [11]: the volume of the dominated space defined by $\widetilde{\mathcal{Z}}_N$, limited by a reference point.
- The average distance D_1 and maximal distance D_2 (to be minimized) [12] between the points of a reference

set $\mathcal{Z}_R$ and the points of $\widetilde{\mathcal{Z}}_N$, by using the Euclidean distance.

- The proportion $P_{\mathcal{Z}_N}$ (to be maximized) of non-dominated points generated.

The reference set used is the Pareto front, that we have generated with an exact method based on the ϵ-constraint method [13]. The reference point used in the hypervolume is the Nadir point, multiplied by 1.1 for both objectives. However, we were not able to generate the Pareto front for some of the biggest and most difficult instances tackled (the 82c, 82d, 101c, 101d, 102c, 102d, 201a and 201b instances). Therefore, for these instances, we have considered outperformance relations [14].

The computer used for the experiments is a Pentium IV with 3 GHz CPUs and 512 MB of RAM. For each instance, the algorithm is run twenty times.

C. Values of the parameters

Several experiments, not described here, have been conducted to determine the values of the parameters of the method. The conclusions are the following:

- We have adopted the following rule to define if `lp_solve` or `Meta-Raps` has to be used in the first phase:
  - If the instance has less than 600 columns and 60 rows, or if the instance has not more than 800 columns and 80 rows, and it is of type A or B, `lp_solve` is used.
  - Otherwise, `Meta-Raps` is used.
- 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 $k = 4$ instead of $k = 2$ does not give impressive improvements, whatever the value of L. Therefore, for the comparison to state-of-the-art results, we will use $k = 2$ and $L = 9$.

D. Comparison to state-of-the-art results

In Table I, we compare the results obtained with 2PPLS to the results of TPM and PMA. We use the quality indicators described in section IV-B, that is the hypervolume $\mathcal{H}$, the average distance D_1, the maximal distance D_2 and the proportion $P_{\mathcal{Z}_N}$ of non-dominated points generated. We also indicate the number of potentially efficient solutions generated ($|PE|$) and the running time in seconds. We do not have the running times of TPM and PMA for each instance, but we will compare later in this section, the average running times obtained by each algorithm, for each group of instances.

We see that we obtain better results for the indicators considered, except for the instance 62c for which TPM obtains slightly better values for some indicators, and for the instances 81c and 81d for which TPM obtains better values for most of the indicators.

The proportion $P_{\mathcal{Z}_N}(\%)$ of non-dominated points generated is included between 21% and 76% and the running time is always less than 45 seconds.

Table I
COMPARISON BETWEEN 2PPLS, TPM AND PMA BASED ON THE INDICATORS (1).

Instance	Algorithm	$\mathcal{H}(10^5)$	D_1	D_2	$P_{\mathcal{Z}_N}$	$\|PE\|$	Time(s)
61a	2PPLS	**650.4639**	**0.090**	**2.287**	**76.15**	221.10	24.94
	TPM	646.9003	1.349	5.949	15.56	55.00	/
	PMA	639.0582	1.233	7.015	2.61	110.10	/
61b	2PPLS	**806.8344**	**0.148**	**2.510**	**52.35**	267.40	30.91
	TPM	801.0863	1.224	6.449	13.31	64.00	/
	PMA	784.6685	1.798	4.815	0.09	136.20	/
61c	2PPLS	**76.9568**	**1.895**	**7.465**	**30.36**	15.60	6.42
	TPM	72.8138	7.073	17.823	21.43	7.00	/
	PMA	66.1876	5.598	14.767	0.00	9.60	/
61d	2PPLS	**585.2976**	**1.061**	**6.246**	**34.63**	50.75	19.36
	TPM	581.5946	2.403	10.061	28.36	24.00	/
	PMA	549.4480	4.819	15.118	0.00	46.20	/
62a	2PPLS	**139.4577**	**0.308**	**2.746**	**57.35**	75.80	27.82
	TPM	136.7701	2.448	9.112	16.33	20.00	/
	PMA	138.5127	0.583	3.652	34.80	70.10	/
62b	2PPLS	**196.6011**	**0.187**	**2.085**	**64.49**	79.35	31.01
	TPM	195.4355	1.171	5.363	24.24	31.00	/
	PMA	195.5629	0.671	4.317	27.58	63.80	/
62c	2PPLS	1.7594	**2.859**	6.905	**44.17**	3.40	1.22
	TPM	**1.7849**	2.866	**6.886**	33.33	3.00	/
	PMA	.1029	27.260	40.446	0.00	1.30	/
62d	2PPLS	**7.6825**	**0.652**	**3.292**	**46.71**	25.30	7.08
	TPM	7.6778	1.056	3.717	34.21	14.00	/
	PMA	7.2203	1.248	3.713	16.58	26.00	/
81a	2PPLS	**1804.7490**	**0.094**	**1.164**	**58.87**	351.65	31.79
	TPM	1787.6098	1.957	9.729	10.61	64.00	/
	PMA	1777.7066	1.253	3.792	0.61	173.80	/
81b	2PPLS	**2295.7315**	**0.141**	**1.176**	**56.20**	272.25	30.97
	TPM	2277.4616	1.690	6.398	11.30	55.00	/
	PMA	2267.9781	1.197	6.756	3.64	138.10	/
81c	2PPLS	113.9085	2.998	12.301	21.79	12.00	8.63
	TPM	**114.0509**	**3.596**	**15.592**	**35.71**	9.00	/
	PMA	110.0607	21.113	61.595	0.00	1.50	/
81d	2PPLS	168.5435	4.098	**16.119**	42.50	10.25	8.01
	TPM	**169.3999**	**2.827**	23.631	**75.00**	10.00	/
	PMA	160.0092	10.640	42.617	0.00	6.60	/
82a	2PPLS	**433.5862**	**0.201**	**2.970**	**62.84**	108.15	30.68
	TPM	427.6693	2.190	7.907	17.42	27.00	/
	PMA	431.1933	0.774	4.617	20.45	87.30	/
82b	2PPLS	**233.6640**	**0.191**	**5.143**	**72.16**	73.50	16.06
	TPM	232.8426	1.179	6.040	23.86	28.00	/
	PMA	232.8318	0.612	6.534	35.91	58.60	/
101a	2PPLS	**1287.8008**	**0.494**	**8.865**	**45.83**	107.80	43.88
	TPM	1284.6645	1.340	13.648	19.11	44.00	/
	PMA	1282.0624	1.020	15.550	10.00	90.50	/
101b	2PPLS	**944.1663**	**0.444**	**6.274**	**48.16**	103.75	43.88
	TPM	937.8272	2.324	10.751	19.15	32.00	/
	PMA	939.9055	1.264	17.493	14.96	91.00	/
102a	2PPLS	**305.8453**	**0.395**	**3.450**	**56.51**	60.75	39.63
	TPM	286.7343	9.920	24.121	0.00	6.00	/
	PMA	305.1965	0.675	6.340	33.01	55.80	/
102b	2PPLS	**340.0914**	**0.419**	**6.688**	**64.77**	66.70	37.86
	TPM	329.3906	3.032	26.430	0.00	31.00	/
	PMA	338.8972	0.644	3.737	28.95	61.50	/

For the other instances (the 82c, 82d, 101c, 101d, 102c, 102d, 201a and 201b instances), we have used the outperformance relations to compare the results of 2PPLS with PMA (and not with TPM since we did not get all the results

of TPM for these instances). We have obtained that the proportion of solutions of 2PPLS that are dominated by PMA is very weak and the proportion of solutions of 2PPLS that dominate PMA is always superior to 50%.

Finally, the comparison of the average running times for solving the different groups of instances is given in Table II. We see that the running times of 2PPLS are less or equal than the running times of TPM and PMA (TPM has been executed on a Pentium IV with 1.8 GHz CPUs and 512 MB of RAM, and PMA on a computer with 750 MHz). Those computers are slower than the one that we used (Pentium IV with 3 GHz CPUs), but we have obtained less computation times for most of the instances, and our results are considerably better on the different indicators considered.

Table II
COMPARISON OF THE RUNNING TIMES (IN SECONDS) OF 2PPLS, TPM AND PMA.

Groups of instances	Dimension	2PPLS	TPM	PMA
11	10 x 100	0.56	2.05	4.2
41	40 x 200	7.62	7.69	20.3
42	40 x 400	8.56	8.56	45.7
43	40 x 200	2.34	7.34	14.8
61	60 x 600	20.41	20.27	132.4
62	60 x 600	16.78	20.35	98.8
81	80 x 800	19.85	30.22	165.9
82	80 x 800	16.44	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

V. CONCLUSION AND PERSPECTIVES

We have shown through this paper the effectiveness of the approach based on 2PPLS, VNS and VLSNS to solve the BOSCP. The integration of VNS and VLSNS into 2PPLS allowed to obtain new state-of-the-art results for the BOSCP. VLSNS is particularly suited to multiobjective optimization, since with this approach a set of new potentially efficient solutions can be easily produced from one solution. VNS, for his part, helps to escape from a local Pareto optimum set.

This paper only addresses a special case of MOCO problems, since we only consider biobjective instances of the MOSCP. We however think that with the approach considered high quality results could be obtained for MOSCP with more than two objectives. The only adaptation that could be necessary will be to manage the number of potentially efficient solutions generated that can be very high in case of MOCO problems with more than two objectives.

REFERENCES

[1] R. Ahuja, O. Ergun, J. Orlin, and A. Punnen, "A survey of very large-scale neighborhood search techniques," *Discrete Appl. Math.*, vol. 123, no. 1-3, pp. 75–102, 2002.

[2] P. Hansen and N. Mladenovic, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449 – 467, 2001.

[3] T. Lust and J. Teghem, "Two-phase Pareto local search for the biobjective traveling salesman problem," *Journal of Heuristics*, vol. 16, no. 3, pp. 475–510, 2010.

[4] L. Paquete, M. Chiarandini, and T. Stützle, "Pareto local optimum sets in the biobjective traveling salesman problem: an experimental study," in *Metaheuristics for Multiobjective Optimisation*, X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, Eds. Berlin: Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, 2004, pp. 177–199.

[5] T. Lust and J. Teghem, "The multiobjective multidimensional knapsack problem: a survey and a new approach," *International Transactions in Operational Research*, vol. 19, no. 4, pp. 495–520.

[6] T. Lust and J. Teghem, "The multiobjective traveling salesman problem: a survey and a new approach," in *Advances in Multi-Objective Nature Inspired Computing*, ser. Studies in Computational Intelligence, C. C. Coello, C. Dhaenens, and L. Jourdan, Eds. Springer Berlin / Heidelberg, 2010, vol. 272, pp. 119–141.

[7] A. Jaszkiewicz, "Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 133–143, April 2003.

[8] C. Prins, C. Prodhon, and R. W. Calvo, "Two-phase method and lagrangian relaxation to solve the bi-objective set covering problem," *Annals OR*, vol. 147, no. 1, pp. 23–41, 2006.

[9] Y. Aneja and K. Nair, "Bicriteria transportation problem," *Management Science*, vol. 25, pp. 73–78, 1979.

[10] G. Lan, G. DePuy, and G. Whitehouse, "An effective and simple heuristic for the set covering problem," *European Journal of Operational Research*, vol. 176, no. 3, pp. 1387 – 1403, 2007.

[11] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: methods and applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[12] P. Czyzak and A. Jaszkiewicz, "Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, vol. 7, pp. 34–47, 1998.

[13] M. Laumanns, L. Thiele, and E. Zitzler, "An adaptative scheme to generate the Pareto front based on the epsilon-constraint method," Technischer Bericht, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Tech. Rep. 199, 2004.

[14] M. Hansen and A. Jaszkiewicz, "Evaluating the quality of approximations of the nondominated set," Technical University of Denmark, Lingby, Denmark, Tech. Rep., 1998.