A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the Pareto memetic algorithm

Andrzej Jaszkiewicz[*]

Working paper
RA-003/01

September 2001

[*]Institute of Computing Science
Poznań University of Technology
ul. Piotrowo 3a, 60-965 Poznań, Poland
Jaszkiewicz@cs.put.poznan.pl
www.cs.put.poznan.pl/~and_j

Abstract

The paper describes a comparative study of multiple-objective metaheuristics on the bi-objective set covering problem. Ten representative methods based on genetic algorithms, multiple start local search, hybrid genetic algorithms and simulated annealing are evaluated in the computational experiment. Nine of the methods are well known from the literature. The paper introduces also a new hybrid genetic algorithm called Pareto memetic algorithm. The results of the experiment indicate very good performance of hybrid genetic algorithms, however, no algorithm was able to outperform all other methods on all instances. Furthermore, the results indicate that the performance of multiple-objective metaheuristics may differ radically even if the methods are based on the same single objective algorithm and use exactly the same problem-specific operators.

Keywords: multiple-objective optimization, metaheuristics, set covering problem.

1. Introduction

In recent years one could observe a growing interest in multiple-objective metaheuristics. The goal of such methods is to generate in a single run a set of solutions approximating whole or a part of the nondominated set. Methods of this kind are applied to various computationally hard multiple-objective problems, e.g. combinatorial optimization problems and nonlinear programming problems.

Although, authors of multiple-objective metaheuristics often claim that their methods may effectively solve large-scale problems, a relatively low number of comparative experiments, in particular involving multiple-objective combinatorial optimization (MOCO) problems, has been published in the literature.

In this paper, we compare ten multiple-objective metaheuristics on the bi-objective set covering problem. Nine of the methods are well known from the literature. The methods are: the multiple-objective genetic local search algorithm (MOGLS) proposed by us [14], [16], Ishibuchi's and Murata's multiple-objective genetic local search (IMMOGLS) [12], Serafini's multiple-objective simulated annealing (SMOSA) [23], multiple-objective simulated annealing proposed by Ulungu et al. (MOSA) [28], Pareto simulated annealing (PSA) proposed by us [2], nondominated sorting genetic algorithm (NSGA) [25], controlled elitist non-dominated sorting genetic algorithm (CENSGA) [3], strength Pareto evolutionary algorithm (SPEA) [30], and a simple multiple-objective multiple start local search (MOMSLS) with random weight vectors [15], [16]. In addition, we introduce a new multiple-objective metaheuristic called Pareto memetic algorithm (PMA).

All methods were implemented in C++ with the use of MOMHLib++ library [11]. Their implementations shared significant fractions of common code. The code of all methods used in this experiment is freely available at http://www-idss.cs.put.poznan.pl/~jaszkiewicz/moscp.

Set covering problem is an NP-complete combinatorial optimization problem [6] often used to test single objective metaheuristics. It finds many practical applications especially in crew scheduling [24]. In practice, solutions of set covering problem often cannot be evaluated with a single objective only. For example, in the case of crew scheduling it may be necessary to

take into account both the cost and the quality of work. Other objectives important in the case of crew scheduling are discussed in [20].

The paper is organized in the following way. In the next section, some basic definitions are given. The Pareto memetic algorithm is described in the third section. In the fourth section, other methods used in the experiment are briefly characterized. Adaptation of the methods to multiple-objective set covering problem is presented in the fifth section. In the sixth section, the experiment design is described. The approach used to evaluate quality of obtained results is presented in the seventh section. In the eighth section, results of the experiments are presented and discussed. Concluding remarks are given in the ninth section.

2. Basic definitions

The general multiple-objective optimization (MOO) problem is formulated as:

$$\text{maximize}\{f_1(\mathbf{x}) = z_1, ..., f_J(\mathbf{x}) = z_J\} \qquad \text{(P1)}$$

$$\text{s.t.} \quad \mathbf{x} \in D,$$

where: *solution* $\mathbf{x} = [x_1, ..., x_I]$ is a vector of *decision variables*, D is the set of feasible solutions. If the variables are discrete the MOO problem is called *multiple-objective combinatorial optimization* (MOCO) problem.

The image of a solution $\mathbf{x}$ in the objective space is a *point* $\mathbf{z}^{\mathbf{x}} = [z_1^{\mathbf{x}}, ..., z_J^{\mathbf{x}}] = \mathbf{f}(\mathbf{x})$, such that $z_j^{\mathbf{x}} = f_j(\mathbf{x})$, j=1,..,J.

Point $\mathbf{z}^1$ *dominates* $\mathbf{z}^2$, $\mathbf{z}^1 \succ \mathbf{z}^2$, if $\forall j$ $z_j^1 \geq z_j^2$ and $z_j^1 > z_j^2$ for at least one j. Solution $\mathbf{x}^1$ dominates $\mathbf{x}^2$ if the image of $\mathbf{x}^1$ dominates the image of $\mathbf{x}^2$.

A solution $\mathbf{x} \in D$ is *Pareto-optimal* (*efficient*) if there is no $\mathbf{x}' \in D$ that dominates $\mathbf{x}$. Point being an image of a Pareto-optimal solution is called *nondominated*. The set of all Pareto-optimal solutions is called the *Pareto-optimal set*. The image of the Pareto-optimal set in the objective space is called the *nondominated set* or *Pareto front*.

An *approximation of the nondominated set* is a set A of feasible points (and corresponding solutions) such that $\neg\exists \mathbf{z}^1, \mathbf{z}^2 \in A$ such that $\mathbf{z}^1 \succ \mathbf{z}^2$, i.e. set A is composed of mutually nondominated points.

The point $\mathbf{z}^*$ composed of the best attainable objective function values is called the *ideal point*:

$$z_j^* = \max\{f_j(\mathbf{x}) | \mathbf{x} \in D\} \qquad j = 1, ..., J.$$

Range equalization factors ([26], ch. 8.4.2) are defined in the following way:

$$\pi_j = \frac{1}{R_j}, \; j=1, \ldots, J$$

where R_j is the (approximate) range of objective z_j in the nondominated set, or D or A. Objective function values multiplied by range equalization factors are called *normalized objective function values*.

Weighted linear scalarizing functions are defined in the following way:

$$s_l(\mathbf{z},\Lambda)=\sum_{j=1}^{J}\lambda_j z_j=\sum_{j=1}^{J}\lambda_j f_j(\mathbf{x})$$

where $\Lambda=[\lambda_1,...,\lambda_J]$ is a weight vector such that $\lambda_j \geq 0\ \forall j$.

Weighted Tchebycheff scalarizing functions are defined in the following way:

$$s_\infty(\mathbf{z},\mathbf{z}^0,\Lambda)=\max_j\{\lambda_j(z_j^0-z_j)\}=\max_j\{\lambda_j(z_j^0-f_j(\mathbf{x}))\},$$

where z^0 is a reference point, $\Lambda=[\lambda_1,...,\lambda_J]$ is a weight vector such that $\lambda_j \geq 0\ \forall j$. Each weighted Tchebycheff scalarizing function has at least one global optimum (minimum) belonging to the set of Pareto-optimal solutions. Note, however, that if the optimum is not unique then some of the optima may be dominated, but must have at least one objective component equal to a Pareto-optimal solution. For each Pareto-optimal solution **x** there exists a weighted Tchebycheff scalarizing function *s* such that **x** is a global optimum (minimum) of *s* ([26], ch. 14.8).

Weight vectors that meet the following conditions:

$$\forall j\ \lambda_j \geq 0,\ \sum_{j=1}^{J}\lambda_j=1,$$

are called normalized weight vectors.

Minimization of the weighted Tchebycheff scalarizing function corresponds to a min-max problem. The problem can be, however, transformed to the following one:

$$\begin{aligned}&\text{minimize } \alpha && \text{(P2)}\\ &\text{s.t.}\\ &\alpha \geq \lambda_j(z_j^0-z_j)=\lambda_j(z_j^0-f_j(\mathbf{x})), \qquad j=1,...,J,\\ &\mathbf{x}\in D.\end{aligned}$$

The *bi-objective set covering problem* (BOSCP) is the problem of covering the rows of a *L*-row, *I*-column, zero-one matrix (a_{lj}) by a subset of the columns minimizing two cost-type objectives. Defining x_i=1 if column *i* (with costs $c_i^1,\ c_i^2 > 0$) is selected in the solution and x_i=0, otherwise the BOSCP is:

$$\begin{aligned}&\text{minimize } \{z_1=\sum_{i=1}^{I}c_i^1x_i,\ z_2=\sum_{i=1}^{I}c_i^2x_i\}\\ &\text{s.t.} \quad \sum_{i=1}^{I}a_{li}x_i \geq 1, \quad l=1,...,L\\ &x_i\in\{0,1\}, \quad j=1,...,J\end{aligned}$$

In BOSCP both objectives are minimized, however, the problem can be easily transformed to the form of problem (P1) by changing signs of the objectives. Thus, all definition introduced above remain valid for BOSCP.

3. Pareto memetic algorithm

Hybrid genetic algorithms (HGAs) are relatively new metaheuristics that hybridize recombination operators with local search, or more generally, with other local heuristics. Other frequently used names are memetic algorithms or genetic local search. Methods of this type often perform very well on combinatorial optimization problems, e.g., on travelling salesperson problem [22], graph coloring problem [7], and quadratic assignment problem [27]. They perform also very well on single objective set covering problem [1]. Thus, the development of multiple-objective versions of HGAs is one of the most promising directions of research.

Pareto memetic algorithm introduced in this paper has been developed on the basis of previously proposed multiple-objective genetic local search (MOGLS) algorithm [14], [16]. The general scheme of PMA, MOGLS, as well as of IMMOGLS [12], may be summarized by the algorithm presented in Figure 1. The three algorithms differ by the way in which solutions are drawn for recombination. In the case of MOGLS, a relatively small temporary elite population *TE* composed of a number (e.g. 10) of solutions being the best on the current scalarizing function is selected from the current set of solutions. Then, two solutions are drawn for recombination from the temporary elite population with uniform probability. In this way the solutions recombined are assured to be very good on the current scalarizing function. This process in repeated in each iteration.

Generate the initial set of solutions

repeat

Draw at random a weight vector Λ defining the current scalarizing function

Select probabilistically two previously generated solutions being good on the current scalarizing function

Recombine the two solutions

Apply a local heuristic to the offspring

until the stopping criterion is met

Figure 1. General scheme of the multiple-objective genetic local search algorithm

The selection process used in MOGLS, is however, time consuming. MOGLS was originally applied to multiple-objective traveling salesperson problem using 2-opt local search [16]. In this case the time needed for selection of recombined solutions is meaningless with respect to local search time. In general, however, MOGLS could also be used with relatively simple and fast heuristics (see e.g. [13]). In this case, solutions selection time significantly influences overall running time.

PMA uses another faster approach based on tournament selection to select good solutions for recombination. Let *CS* denote current set of solutions. In each iteration, a sample *T* is drawn at random with repetitions from *CS*. Then, two best solutions in *T* are winners of the tournament and are selected for recombination. The size of *T* is set in order to obtain expected rank induced by the current scalarizing function of recombined solutions similar as in the case of MOGLS. Current scalarizing function *s* induces a rank in set *CS* with the best solution on *s* having rank 1. For simplicity assume that no two solutions in *CS* have equal value of *s*. In the

case of MOGLS, the expected rank of recombined solutions is obviously $|TE|/2+0.5$. Below we analyze expected rank in the case of tournament selection.

Consider a solution **x** having rank r and selected for the tournament. There are r-1 better solutions and $|CS|-r+1$ not better solutions (including **x**). The probability P_1 that **x** is the best solution in T under condition that **x** is selected for the tournament is equal to the probability of drawing $|T|$-1 times a solution not better than **x**:

$$P_1 = \left(\frac{|CS|-r+1}{|CS|}\right)^{|T|-1} = \left(1-\frac{r-1}{|CS|}\right)^{|T|-1}.$$

The probability P_2 that **x** is drawn for the tournament is equal to:

$$P_2 = 1-\left(\frac{|CS|-1}{|CS|}\right)^{|T|},$$

thus, the probability P_3 that **x** is the best solution in the tournament is:

$$P_3 = P_1 P_2 = \left(1-\left(\frac{|CS|-1}{|CS|}\right)^{|T|}\right)\left(1-\frac{r-1}{|CS|}\right)^{|T|-1},$$

and the expected rank of the best solution in the tournament $Er1$ is:

$$Er1 = \sum_{r=1}^{|CS|} r\left(1-\left(\frac{|CS|-1}{|CS|}\right)^{|T|}\right)\left(1-\frac{r-1}{|CS|}\right)^{|T|-1} = \left(1-\left(\frac{|CS|-1}{|CS|}\right)^{|T|}\right)\sum_{r=1}^{|CS|} r\left(1-\frac{r-1}{|CS|}\right)^{|T|-1}.$$

In a similar way one may calculate expected rank of the second best solution. In this case, the probability P_4 that a solution **x** having rank r and selected for the tournament is the second best solution is calculated from Bernoulli distribution and is equal to:

$$P_4 = \left(1-\frac{r-1}{|CS|}\right)^{|T|-2}\left(\frac{r-1}{|CS|}\right)(|T|-1).$$

Thus, the expected rank of the second best solution $Er2$ is equal to:

$$Er2 = \left(1-\left(\frac{|CS|-1}{|CS|}\right)^{|T|}\right)\sum_{r=1}^{|CS|} r\left(1-\frac{r-1}{|CS|}\right)^{|T|-2}\left(\frac{r-1}{|CS|}\right)(|T|-1)$$

In result, the expected rank of recombined solutions Er is equal to:

$$Er = \frac{Er1+Er2}{2} = \frac{\left(1-\left(\frac{|CS|-1}{|CS|}\right)^{|T|}\right)}{2}\sum_{r=1}^{|CS|} r\left(\left(1-\frac{r-1}{|CS|}\right)^{|T|-1} + \left(1-\frac{r-1}{|CS|}\right)^{|T|-2}\left(\frac{r-1}{|CS|}\right)(|T|-1)\right).$$

The above formula, although quite complicated, is within typical ranges of $|CS|$ and Er, i.e. $|CS| >> Er \geq 5$, well approximated by the following formula:

$$Er \approx \frac{3|CS|}{2|T|}.$$

In PMA, the expected rank of recombined solutions is used as a parameter of the method and the size of the tournament sample T is set in the following way:

$$|T| \approx \frac{3|CS|}{2Er}.$$

In general, low values of Er result in very high selection pressure but may cause a premature convergence. The algorithm of PMA is summarized in Figure 2.

The set of potentially Pareto-optimal solutions PP is the outcome of the method and is an approximation of the nondominated set. It is empty at the beginning and is updated whenever a new solution is generated by the local heuristic. Updating the set of potentially Pareto-optimal solutions with solution $\mathbf{x}$ consists of:

- adding $\mathbf{f}(\mathbf{x})$ to PP if no point in PP dominates $\mathbf{f}(\mathbf{x})$,
- removing from PP all points dominated by $\mathbf{f}(\mathbf{x})$.

In this experiment we used weighted Techbycheff scalarizing functions in PMA and other hybrid genetic algorithms. The scalarizing functions used a reference point based on the set of potentially Pareto-optimal solutions defined in the following way (note that in the case of BOSCP signs of the objectives were changed, see section 2):

$$z_j^{**}(PP) = \max\{z_j \mid \mathbf{z} \in PP\} + 0.1(\max\{z_j \mid \mathbf{z} \in PP\} - \min\{z_j \mid \mathbf{z} \in PP\}) \qquad j = 1,\ldots,J.$$

Set CS is organized as a queue of size $N \times S$, where S is the number of initial solutions and N is a parameter of the method typically equal to $2Er$-1 (a value assuring compatibility with approach used in MOGLS). New solutions are added to the beginning of the queue. If the size of the queue is bigger than $N \times S$ then the last solution from the queue is removed. In our experiment, however, a value of $N = 2Er$-1 allowed to store all generated solution in CS until the end of the algorithm run.

Parameters: *Er* – expected rank of recombined solutions, *S* – number of initial solutions or *K* - parameter used in condition (1), stopping criterion

Initialization:

The set of potentially Pareto-optimal solutions *PP*:=∅

The current set of solutions *CS*:=∅

Generation of the first approximation of the ideal point:

for each objective f_j

Construct a new feasible solution **x** by a randomized algorithm

Apply a local heuristic optimizing objective f_j to solution **x** obtaining **x**'

Add **x**' to the current set of solutions *CS*

Update set *PP* with **x**'

Generation of the initial set of solutions:

repeat

Draw at random a weight vector Λ

Construct a new feasible solution **x** by a randomized algorithm

Apply a local heuristic optimizing $s(\mathbf{z},...,\Lambda)$ to solution **x** obtaining **x**'

Add **x**' to the current set of solutions *CS*

Update set *PP* with **x**'

until *S* initial solutions are generated or condition (1) is fulfilled

Main loop:

repeat

Draw at random a weight vector Λ

From *CS* draw at random with uniform probability a sample *T* of $3|CS|/2Er$ solutions

Recombine the best and the second best solution on $s(\mathbf{z},...,\Lambda)$ obtaining $\mathbf{x}_1$

Apply a local heuristic optimizing $s(\mathbf{z},...,\Lambda)$ to solution $\mathbf{x}_1$ obtaining $\mathbf{x}_1$'

if $\mathbf{x}_1$' is better on $s(\mathbf{z},...,\Lambda)$ than the second best solution in *T* **then**

Add $\mathbf{x}_1$' to the current set of solutions *CS*

Update set *PP* with $\mathbf{x}_1$'

until the stopping criterion is met

Figure 2. The algorithm of Pareto memetic algorithm. $s(\mathbf{z},...,\Lambda)$ **stands for either** $s_\infty(\mathbf{z},\mathbf{z}^{**}(PP),\Lambda)$ **or** $s_l(\mathbf{z},\Lambda)$**. The former version is used in this experiment**

The initial solutions are generated by local optimization of randomly selected scalarizing functions. The number of the initial solutions *S* is an additional parameter of the method. We use also another approach that allows automatically stopping generation of initial solutions.

Single objective hybrid genetic algorithms often start by generation of a number of solution by local optimization. In the multiple-objective case, we propose to stop generating the initial solutions when average quality of K best solutions in the set of initial solutions over all scalarizing functions is the same as the average quality of solutions generated by the local heuristic used for optimization of these functions. Precisely, the generation of initial solutions is stopped when the following condition is met (minimization of the scalarizing function is assumed):

$$\frac{1}{|CS|}\sum_{x\in CS}\left(\bar{s}_x\left(B\left(K,CS,\mathbf{x},s_x\right)\right)-s_x(\mathbf{x})\right)\leq 0\,, \tag{1}$$

where $\mathbf{x}$ is an initial solution being obtained by local optimization of a scalarizing function s_x (note that $\mathbf{x}$ does not need to be the best solution on s_x in the set CS because solutions obtained by local optimization of other functions may be better on s_x), $B(K,CS,\mathbf{x},s_x)\subseteq CS$ is the set of K best solutions of function s_x different from $\mathbf{x}$ and $\bar{s}_x(B(K,CS,\mathbf{x},s_x))$ is the average value of s_x in $B(K,CS,\mathbf{x},s_x)$:

$$\bar{s}_x\left(B\left(K,CS,\mathbf{x},s_x\right)\right)=\frac{\sum_{\mathbf{y}\in B(K,CS,\mathbf{x},s_x)} s_x(\mathbf{y})}{K}\,.$$

In order to draw at random a weight vector the algorithm presented in Figure 3 is used. The algorithm uniformly samples the whole set of normalized weight vectors [17]. Note that by changing the algorithm for drawing the weight vectors one can concentrate the method on a subregion of the nondominated set [13].

In real-life applications the ranges of objectives usually differ a lot, often by many levels of magnitude. So, the normalized weight vectors should be applied to normalized objective values (see section 2). The range equalization factors are updated on-line from the ranges of particular objectives in the set of potentially Pareto-optimal solutions.

$$\lambda_1=1-\sqrt[J-1]{\text{rand}()}\,,\ \ldots,\ \lambda_j=\left(1-\sum_{l=1}^{j-1}\lambda_l\right)\left(1-\sqrt[J-1-j]{\text{rand}()}\right),\ \ldots,\ \lambda_J=1-\sum_{l=1}^{J-1}\lambda_l$$

Figure 3. Algorithm for generating random normalized weight vectors. Function rand() returns a random value from the range <0,1> with uniform probability

4. Other methods tested in the experiment

Because of limited space only basic information about other algorithms tested in the experiment is given. Further details may be found in the literature.

As it was mentioned in the previous section, PMA normalizes the objective values with range equalization factors calculated on-line from the ranges of particular objectives in the set of potentially Pareto-optimal solutions. Although descriptions of some algorithms do not refer to this issue, we use objective values normalization in all algorithms that use scalarizing functions or distance measures in the objective space.

4.1. Hybrid genetic algorithms

Three hybrid genetic algorithms have been tested in the experiment. The Pareto memetic algorithm (PMA) has been described in section 3. Multiple-objective genetic local search algorithm (MOGLS) has also been briefly described in that section.

Ishibuchi's and Murata's multiple-objective genetic local search (IMMOGLS) [12] was the first multiple-objective hybrid genetic algorithm described in the literature. As it was already mentioned in section 3, its general scheme is the same as of PMA and MOGLS and is presented in figure Figure 1. The method stores relatively small number of solutions in a standard genetic population. New population replaces the old one but some elite solutions are preserved. The solutions are selected for recombination according to the roulette wheel scheme, with fitness depending on the current scalarizing function. This results in some pressure towards recombination of solutions good on the current scalarizing function but, in general, the solutions recombined are worse on this function that in the case of MOGLS and PMA.

4.2. Multiple-objective multiple start local search (MOMSLS)

The method corresponds to the *Generation of the initial set of solutions* phase of PMA and other hybrid genetic algorithms. The method repeatedly draws a scalarizing function and uses a local heuristic to optimize the function starting from a random solution.

4.3. Simulated annealing algorithms

The first multiple-objective version of simulated annealing has been proposed by Serafini [23]. The method is called Serafini's multiple-objective simulated annealing (SMOSA). The algorithm of the method is almost the same as the algorithm of single objective SA. The method uses a single current solution. Serafini considered a number of multiple-objective rules for acceptance of new solutions. In this experiment, a new neighborhood solution is accepted with an acceptance rule that may be interpreted as the local use of the weighted linear scalarizing function. In each iteration, the weight vector used in the acceptance rule is modified randomly. Each weight is allowed to change by no more than 10%.

Ulungu et al. [28] proposed a method called multiple-objective simulated annealing (MOSA). The method uses a number of generating solutions exploring different regions of the nondominated set. With each of the solutions a predefined weight vector is associated. The weight vectors do not change during the run of the method. The acceptance rule of new solutions is the same as in the case of SMOSA.

Pareto simulated annealing (PSA) [2] also uses a sample of generating solutions. Weight vectors associated with the generating solutions are modified in each iteration in order to induce a repulsion mechanism assuring dispersion of the solutions over all regions of the nondominated set. The method uses the same acceptance rule as the two above algorithms.

Summarizing, all simulated annealing algorithms use locally scalarizing functions in acceptance rules, and use different weight vectors, either predefined or updated during the run of the algorithm, in order to assure dispersion of the search over all regions of the nondominated set.

4.4. Genetic algorithms

Nondominated sorting genetic algorithm (NSGA) [25] is a relatively simple Pareto-ranking based multiple-objective GA. Its scheme is very similar to the single objective GA with generations replacements. The only difference is in fitness assignment that is based on a ranking induced by the dominance relation. In each iteration, NSGA assigns rank 1 to all solutions nondominated in the current population. Then, the nondominated solutions are temporarily removed from the population and the next rank is assigned to the solutions nondominated in the remaining part of the population. The process is continued until all solutions in the population are ranked. The method uses fitness sharing in order to avoid convergence of all solutions to a small region of the nondominated set caused by the genetic drift.

Controlled elitist non-dominated sorting genetic algorithm (CENSGA) [3] is a modification of NSGA that preserves good solutions from the current population. In order to assure diversification of the population it preserves not only solutions nondominated in the current population but also some dominated solutions. CENSGA uses also more effective way for rank assignment.

In strength Pareto evolutionary algorithm (SPEA) [30] both solutions from the current population and from the set of potentially Pareto-optimal solutions participate in the selection which is also based on Pareto ranking. In addition, the method uses a new Pareto-based niching method to preserve diversity of the population.

Original versions of NSGA and CENSGA do not use the external set of potentially Pareto-optimal solutions and outcomes of the methods are the final populations. In our case, we used, however, such set in all methods. The set was updated with each newly generated solution.

5. Adaptation of the methods to multiple-objective set covering problem

Metaheuristics are often used to solve single objective set covering problem. Various adaptations of metaheuristics to this problem are described for example in [1], [4], [19] and [21]. Thus, we base our adaptation of multiple-objective metaheuristics to BOSCP on these works. Note, that although we use bi-objective instances in this experiment, the adaptation may be used in the case of general multiple-objective set covering problem.

Solutions encoding

In our case, a solution $\mathbf{x}$ is encoded as a set of columns, i.e. $\mathbf{x} \subseteq C$, where C is the set of all columns in matrix (a_{lj}) (see section 2). Of course, this encoding is equivalent to a binary representation using I binary variables or genes (see e.g. [1]), but this binary representation is not used directly in our implementation.

Generation of initial solutions

Initial solutions are constructed in a way similar to that proposed in [4]. At first, we consider each row $l \in 1,\ldots,L$, starting from a randomly selected row. If l is not covered yet, a column i that covers l is drawn at random. A column i covers row l if $a_{lj}=1$. Solutions obtained in this way may include many redundant columns, i.e. columns that can be removed without violating feasibility. Thus, in the next step the redundant columns are removed iteratively until no redundant column remains in the solution. In the case of methods that use scalarizing

functions (simulated annealing algorithms, hybrid genetic algorithms and MOMSLS), in each iteration, the redundant column with the highest ratio of:

$$\frac{\text{scalarizing function value improvement caused be removal of the column}}{\text{the number of non zero elements in the column}}$$

is removed. In the case of genetic algorithms, in each iteration, a randomly selected redundant column is removed.

Neighborhood operator

We use a neighborhood operator that is guided by a scalarizing function. At first, a randomly selected column is removed from the solution. In result an unfeasible solution is obtained. Then, the solution is repaired in greedy manner by inserting columns with the lowest ratio of:

$$\frac{\text{scalarizing value decline caused by insertion of the column}}{\text{the number of uncovered rows covered by the column}}$$

The column removed in the first step is not considered by the greedy procedure, thus, the neighborhood operator always produces a new solution.

Local search

Local search is guided by a scalarizing function. We use the steepest version of local search, i.e. the whole neighborhood of the current solution is tested and the best local move is performed. Local search ends when no improving move may be found in the neighborhood of the current solution. In order to test the whole neighborhood removal of each column is considered in the neighborhood operator.

Recombination operator

We use a recombination operator that produces a single offspring from two recombined solutions (parents). The operator is based on the idea of distance preserving crossover [22], i.e. it preserves elements common to both parents. At first, we place in the offspring all columns that are included in both recombined solutions. The columns that appear in one of the parents only are placed in the offspring with 50% probability. This, in general, does not guarantees covering of all rows. Then, all uncovered rows with randomly selected columns. After that, rows are selected for mutation with 5% probability. The mutation consists in drawing a random column covering the mutated row and adding it to the solution. In the case of genetic algorithms, redundant columns are then removed in random order. In the case of hybrid genetic algorithms, local search assures removal of redundant columns.

6. Computational experiment design

Evaluation of metaheuristics should involve at least two criteria – the quality of obtained results and running time. The way we evaluate the quality of results is described in section 7. In order to measure running time two main approaches are usually used: the number of generated solutions/number of function evaluations (see e.g. [30]) or CPU time (see e.g. [1]). The advantage of CPU time is that it takes into account differences in time needed to generate new solutions by different operators and any CPU time overload introduced by a given method (e.g. time needed for selection of solutions for recombination). CPU time has also the highest practical importance among running time measures. The disadvantage of CPU time is that it is influenced by computer, programming language and implementation style characteristics. In our case, however, we were able to test all methods on the same computer, implemented in the same language and with the use of the same library. In result, most lines

of code used by each method were shared with other methods. Thus, we decided to use CPU time measure in this experiment.

We used 44 instances of BOSCP generated by Gandibleux and available at http://www.univ-valenciennes.fr/ROAD/MCDM/ListMOSCP.html. The instances differ not only by size but also by other characteristics. The instances have from 10 to 200 rows and from 100 to 1000 columns. There are four sets of the instances. In the case of instances from set A, costs were generated randomly and independently. In the case of instances from set B, only coefficients of the first objective were generated randomly while the coefficients of the second objectives were set in the following way:

$$c_i^2 = c_{I-i+1}^1, \qquad i=1,...,I.$$

In the case of instances from set C, columns are divided into several subsets. All columns from one subset have the same coefficients on both objectives. Set D of instances combines characteristics of sets B and C.

Because of limited space we do not report all results, but only results obtained for 12 representative instances. All results, as well as all generated approximations, are available at http://www-idss.cs.put.poznan.pl/~jaszkiewicz/moscp.

Table 1. Characteristics of the test instances

Instance	Number of rows	Number of columns	Set
2scp41A	40	200	A
2scp41B			B
2scp41C			C
2scp41D			D
2scp81A	80	800	A
2scp81B			B
2scp81C			C
2scp81D			D
2scp201A	200	1000	A
2scp201B			B
2scp201C			C
2scp201D			D

On each instance, each method was run 10 times. Each method was allowed to run for the same time. Since the instances differ significantly it would not be a good approach to use the same CPU time for all instances. Thus, at first, PMA was run ten times with parameters described below and average running time of PMA was used as the stopping criterion for all other methods.

The parameters of all methods were selected experimentally on the basis of the best choice principle. We did not tune, however, the parameters to particular instances, but set the parameters in order to obtain a relatively good performance on all instances.

The number of initial solutions was set in the same way for all hybrid genetic algorithms. We used the automatic approach described in section 3. Since, the number of initial solutions obtained in this way is not deterministic (although usually is characterized by a low dispersion [16]) we decided to use the same number of initial solutions in all runs of the three HGAs. The condition (1) was tested only once and then the same number of initial solutions was used in all other runs of PMA, MOGLS and IMMOGLS on the same instance. The parameter K

was set to 10. The size of temporary elite population used by MOGLS was set to 15. In the case of PMA, the expected rank *Er* was set to 7.5. PMA was allowed to perform ten times more recombinations than the number of initial solutions. In the case of IMOMGLS, the size of genetic population was set equal to the number of initial solutions and the elite size was set equal to 10% of the population size.

In all multiple-objective genetic algorithms we used populations of size 100 which is quite typical value used in such methods [29]. The neighborhood distance used in NSGA was set to 0.2. The reduction rate in CENSGA was set to 0.8. The nondominated population size of SPEA was set to 100, and clustering was performed when the size of the population exceeded 150.

In all simulated annealing algorithms the starting temperature was set to 0.05 and final temperature to 0.0005. After each temperature plateau the temperature was multiplied 0.9. In result, 44 temperature plateaus were obtained. To each temperature plateau 1/44 of the average running time of PMA was allocated. The number of generating solutions used in PSA and the number of predefined weight vectors used in MOSA was set to 8. The weights change coefficient used in PSA was set to 0.0003.

7. Quality evaluation

As the quality measure of approximations of the nondominated set we use average best value of the weighted Tchebycheff scalarizing function over a set of systematically generated normalized weight vectors [8], [15], [14]. We use all normalized weight vectors in which each individual weight takes on one of the following values: $\left\{ l/k,\ l = 0,\ldots,k \right\}$, where k is a sampling parameter. The set of such weight vectors is denoted by Ψ_s and defined mathematically as:

$$\Psi_s = \left\{ \Lambda = [\lambda_1,\ldots,\lambda_J] \in \Psi \mid \lambda_j \in \left\{ 0, 1/k, 2/k, \ldots, k-1/k, 1 \right\} \right\},$$

where Ψ is the set of all normalized weight vectors.

The measure is calculated as:

$$R(A) = \frac{\sum_{\Lambda \in \Psi_s} s_\infty^*\left(\mathbf{z}^0, A, \Lambda\right)}{\left|\Psi_s\right|},$$

where $s_\infty^*\left(\mathbf{z}^0, A, \Lambda\right) = \min_{\mathbf{z} \in A}\left\{ s_\infty\left(\mathbf{z}, \mathbf{z}^0, \Lambda\right) \right\}$ is the best value achieved by function $s_\infty\left(\mathbf{z}, \mathbf{z}^0, \Lambda\right)$ on approximation A.

In our experiment we used sampling parameter k equal to 100, i.e. 101 weight vectors were used.

In the case of 16 instances with up to 40 rows and 400 columns we found exact ideal points optimizing each objective individually using Lingo [9] and Frontline Premium [10] IP solvers. The ideal points were used as reference points $\mathbf{z}^0$ in the quality measure. In the case of other instances, we used upper approximations of the ideal points obtained by optimization of individual objectives on relaxed version of BOSCP. The relaxed version was obtained by allowing variables x_i to take continuous values from range <0,1>. Before applying the quality measure all objective values were normalized using objective ranges observed during individual optimization of the objectives, i.e. the range one of the objectives objective was equal to the difference between its best value found by optimization of this objective and the value of this objective found by optimization of the other objective.

The measure achieves the best (minimum) value on the nondominated set. However, the best value may be found without generating the whole nondominated set if one is able to find the optimum value of the weighted Tchebycheff scalarizing function for each weight vector in set Ψ_s, which requires solving problem (P2). We used to this end the mentioned above two solvers. Because of time requirements we were able to find the best values of the measure only for 16 instances with up to 40 rows and 400 columns. An interesting observation was that instances from sets C and D were more difficult for the solvers than instances from sets A and B. The average running time of the solvers was about 4 times higher on instances from sets C and D. We have also noticed that instances from sets C and D contain much fewer nondominated points (cf. Figure 5 and Figure 6). In the case of other 28 instances, we have found lower bounds on the quality measure by solving relaxed version of problem (P2) for each weight vector. The relaxed version was obtained by allowing variables x_i to take continuous values from range <0,1>.

8. Results and discussion

Table 2 presents running times and numbers of solutions generated by particular algorithms on the 12 representative instances. All calculations were run on a 400 MHz PC. Note that in the case of HGAs and MOMSLS we count only local optima (we do not count intermediate solutions generated during local search), so, the numbers of generated solutions are naturally much lower than in the case of other algorithms. In the case of simulated annealing, we count accepted solutions only. Thus, the results should rather be used to compare algorithms from the same class.

Results of the experiment on the 12 representative instances are presented in Table 3 and Figure 4. Detailed numerical results for all instances are available at http://www-idss.cs.put.poznan.pl/~jaszkiewicz/moscp. Each chart in Figure 4 corresponds to a different instance. Each chart contains ten results representing the distribution of R for (from left to right) the best possible value or lower bound, PMA, MOGLS, IMMOGLS, MOMSLS, SPEA, NSGA, CENSGA, PSA and MOSA. For each method we present average value and ± standard deviation range. For each chart the scale is from the best possible value or lower bound at the bottom to the highest observed value at the top. We did not include results of SMOSA in Figure 4 because average results of SMOSA were much worse than of other algorithms with very high standard deviations. More detailed analysis indicated that in most runs SMOSA gave results comparable to results of other simulated annealing algorithm (PSA and MOSA). However, in about 15% of runs of SMOSA the results were enormously poor. It indicates that sometimes the single solution used by the algorithm gets trapped in poor regions of the solution space. Other simulated annealing algorithms are less vulnerable to such situations due to the use of several generating solutions.

The results indicate that instances from sets C and D are more difficult also for multiple-objective metaheuristics (see section 7). In general, the gap between results of the metaheuristics and the best possible value or lower bound is higher in case of these instances.

Table 2. Numbers of generated solutions and running times on the test instances

		Instance			
		2scp41A	**2scp41B**	**2scp41C**	**2scp41D**
Number of initial solutions in HGAs		110	110	60	50
Running time [s]		27.5	26.4	13.9	13.4
Number of generated solutions	**PMA**	1210	1210	660	550
	MOGLS	1139	1067	566	496
	IMMOGLS	695	662	422	402
	MOMSLS	426	426	310	284
	SPEA	39580	39690	21430	20320
	NSGA	31630	30000	12270	9820
	CENSGA	27220	23340	11800	10600
	PSA	88749	89307	29774	31894
	MOSA	80792	86143	22745	31321
	SMOSA	166252	126512	38488	46837
		Instance			
		2scp81A	**2scp81B**	**2scp81C**	**2scp81D**
Number of initial solutions in HGAs		140	150	20	30
Running time [s]		285.4	306.4	25.6	46.2
Number of generated solutions	**PMA**	1540	1650	220	330
	MOGLS	1535	1643	178	274
	IMMOGLS	842	902	224	278
	MOMSLS	413	491	46	88
	SPEA	189330	209020	15780	34080
	NSGA	160170	192660	7410	18290
	CENSGA	131350	145550	9280	16730
	PSA	613388	684540	35223	72135
	MOSA	587273	637394	12926	39385
	SMOSA	827919	850063	50898	77904
		Instance			
		2scp201A	**2scp201B**	**2scp201C**	**2scp201D**
Number of initial solutions in HGAs		140	130	70	40
Running time [s]		1060.9	937.0	487.8	261.4
Number of generated solutions	**PMA**	1540	1430	770	440
	MOGLS	1571	1431	773	429
	IMMOGLS	1206	1172	639	386
	MOMSLS	677	604	391	201
	SPEA	87240	72410	29930	16760
	NSGA	73190	70890	32600	24980
	CENSGA	31320	17080	12890	12360
	PSA	891138	817970	217459	53073
	MOSA	799390	730129	218638	40693
	SMOSA	1311530	1079046	327267	195030

Taking into account all results hybrid genetic algorithms, especially PMA and MOGLS, are the best performers in the experiment. The performance of the two methods is very similar. Thus, the new faster selection mechanism introduced in PMA does not deteriorate quality with respect to MOGLS. PMA usually generates slightly more solutions than MOGLS but in some case the opposite situation is observed. Since, PMA uses a faster selection mechanism one could expect that PMA should be able to generate more solutions than MOGLS. The reduction of selection time has, however, very low influence on the overall running time in the case of BOSCP because most time is spent on local optimization. The advantage of PMA

selection mechanism should be more evident if a simpler and faster heuristic is used. IMMOGLS performs worse than PMA and MOGLS on all instances but 2scp81C and 2scp81D. In most cases, IMMOGLS was able to generate fewer solutions than PMA and MOGLS in the same time. In our opinion, this is because IMOMGLS recombines, in general, worse solutions, and thus the offspring are further from local optima than in the case of PMA and MOMGLS. The results indicate that selection of very good solutions for recombination is crucial for the performance of the two latter algorithms on BOSCP.

MOMSLS performs worse than all HGAs. The method was able to generate much fewer solutions than HGAs, because local search needs more time to achieve a local optimum when started from an initial solution. The results show that hybridization of recombination operator with local search is very beneficial in the case of BOSCP.

The performance of PSA and MOSA is very similar and significantly depends on instance size. The methods gave the best results on 2scp201C and 2scp201D instances, but performed poorly on 2scp41C and 2scp41D instances. This is probably due to the difficulty in finding parameters appropriate for all instances. We expect that performance on smaller instances could be significantly improved if the parameters, especially starting and final temperature, were tuned for each instance. Thus, although PSA and MOSA are capable of giving high quality results, they are less robust than HGAs. Note that since we use normalized objective values the above phenomenon cannot be explained by different ranges of objectives in different instances. SMOSA always accepted in the same time many more solutions than PSA and MOSA, and PSA, in most cases, accepted mores solutions than MOSA. This is because, SMOSA and PSA modify the weights used in scalarizing function in each iteration. Furthermore, SMOSA makes greater changes to the weights in a single iteration.

Excluding SMOSA, multiple-objective genetic algorithms are the worst performers in the experiment. Only on 2scp41C and 2scp41D instances CENSGA gives results comparable to PMA and MOGLS. In most cases, however, the three GAs give the worst results. In addition, dispersion of the quality of results is higher than in the case of other methods. The simplest algorithm NSGA gives the worst results among the three GAs. Thus, the extensions to the basic idea of Pareto-ranking resulted in improved performance of the newer algorithms, but did not make the new algorithm competitive to other multiple-objective metaheuristics on BOSCP. Furthermore, one may note that SPEA performs relatively well on instances from sets A and B, while CENSGA performs better on instances from sets C and D. Note that CENSGA puts more emphasis on diversification (see section 4.4) which apparently pays off on more difficult instances. In most cases, SPEA generates the highest number of solutions among GAs.

Table 3. Average values and ± standard deviation of the quality of results

	Instance			
	2scp41A	2scp41B	2scp41C	2scp41D
PMA	0.1097 ± 0.0010	0.0821 ± 0.0008	0.1269 ± 0.0089	0.0888 ± 0.0033
MOGLS	0.1108 ± 0.0019	0.0830 ± 0.0010	0.1346 ± 0.0070	0.0915 ± 0.0041
IMMOGLS	0.1202 ± 0.0027	0.0906 ± 0.0020	0.1501 ± 0.0069	0.1067 ± 0.0041
MOMSLS	0.1273 ± 0.0015	0.0975 ± 0.0011	0.1957 ± 0.0099	0.1303 ± 0.0112
SPEA	0.1315 ± 0.0103	0.1202 ± 0.0110	0.1659 ± 0.0351	0.0977 ± 0.0098
NSGA	0.1737 ± 0.0230	0.1369 ± 0.0076	0.1849 ± 0.0525	0.1218 ± 0.0235
CENSGA	0.1342 ± 0.0081	0.1208 ± 0.0085	0.1353 ± 0.0106	0.0926 ± 0.0075
PSA	0.1196 ± 0.0022	0.0879 ± 0.0018	0.1411 ± 0.0068	0.1040 ± 0.0041
MOSA	0.1191 ± 0.0018	0.0884 ± 0.0012	0.1529 ± 0.0175	0.1119 ± 0.0039
SMOSA	0.2457 ± 0.2015	0.1164 ± 0.0902	0.1697 ± 0.1414	0.1487 ± 0.1206
Best possible value	0.1051	0.0779	0.1073	0.0668
	Instance			
	2scp81A	2scp81B	2scp81C	2scp81D
PMA	0.0844 ± 0.0003	0.0815 ± 0.0003	0.0205 ± 0.0021	0.0400 ± 0.0044
MOGLS	0.0844 ± 0.0002	0.0816 ± 0.0006	0.0208 ± 0.0023	0.0486 ± 0.0075
IMMOGLS	0.0903 ± 0.0007	0.0913 ± 0.0013	0.0209 ± 0.0020	0.0402 ± 0.0052
MOMSLS	0.0958 ± 0.0013	0.0977 ± 0.0016	0.0264 ± 0.0021	0.0751 ± 0.0030
SPEA	0.1260 ± 0.0121	0.1174 ± 0.0150	0.1441 ± 0.0726	0.2050 ± 0.1070
NSGA	0.1880 ± 0.0243	0.1733 ± 0.0143	0.2406 ± 0.0666	0.2361 ± 0.1291
CENSGA	0.1525 ± 0.0193	0.1258 ± 0.0104	0.1691 ± 0.0696	0.1908 ± 0.0334
PSA	0.0844 ± 0.0003	0.0833 ± 0.0010	0.0203 ± 0.0006	0.0488 ± 0.0022
MOSA	0.0852 ± 0.0006	0.0860 ± 0.0013	0.0207 ± 0.0010	0.0491 ± 0.0029
SMOSA	0.2151 ± 0.2665	0.1345 ± 0.1399	0.6923 ± 1.0885	0.6369 ± 1.2544
Lower bound	0.0800	0.0771	0.0101	0.0137
	Instance			
	2scp201A	2scp201B	2scp201C	2scp201D
PMA	0.0813 ± 0.0010	0.0659 ± 0.0006	0.1649 ± 0.0086	0.4967 ± 0.0351
MOGLS	0.0812 ± 0.0009	0.0662 ± 0.0006	0.1659 ± 0.0047	0.4705 ± 0.0158
IMMOGLS	0.0900 ± 0.0013	0.0727 ± 0.0008	0.1906 ± 0.0047	0.5296 ± 0.0249
MOMSLS	0.0941 ± 0.0010	0.0764 ± 0.0012	0.2245 ± 0.0110	0.6944 ± 0.0205
SPEA	0.1338 ± 0.0080	0.1089 ± 0.0122	0.2526 ± 0.0168	0.7228 ± 0.0843
NSGA	0.2024 ± 0.0215	0.1534 ± 0.0179	0.2711 ± 0.0301	0.7590 ± 0.1263
CENSGA	0.1754 ± 0.0136	0.1559 ± 0.0209	0.2384 ± 0.0162	0.6666 ± 0.0504
PSA	0.0831 ± 0.0009	0.0665 ± 0.0004	0.1506 ± 0.0044	0.4146 ± 0.0051
MOSA	0.0833 ± 0.0007	0.0677 ± 0.0007	0.1504 ± 0.0041	0.4226 ± 0.0136
SMOSA	0.3919 ± 0.3934	0.2112 ± 0.2988	0.2491 ± 0.2584	1.0697 ± 0.8540
Lower bound	0.0590	0.0513	0.0552	0.0789

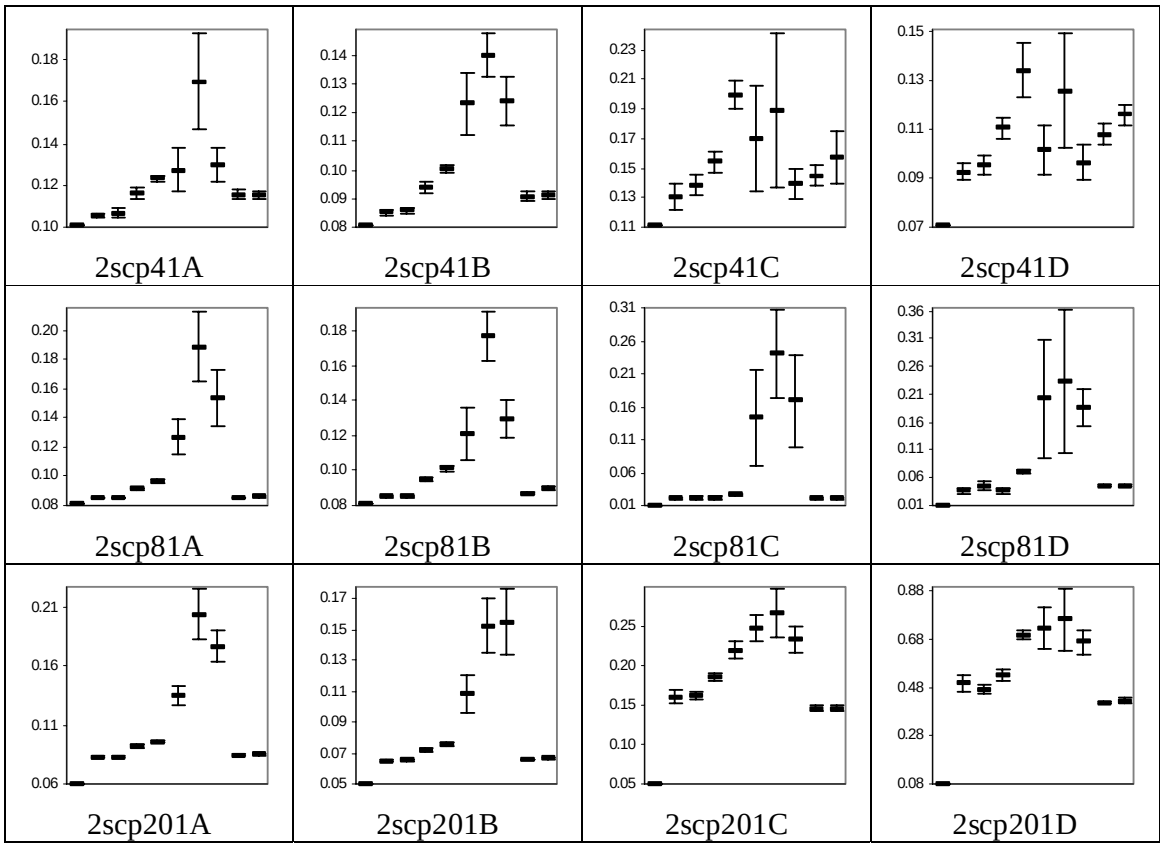

Figure 4. Results of the experiment in graphical form

In Figures 5-8 some exemplary approximations for four different instances are presented (each of them was generated in the first run of a given algorithm). We did not include all approximations in order to preserve clarity of the figures. The points referred to as "exact" are obtained by local optimization of weighted Tchebycheff scalarizing functions with IP solvers (see section 7). Note, that the points may be dominated (see section 2), which apparently happens in many cases. The points referred to as "relaxed" are points obtained by local optimization of weighted Tchebycheff scalarizing functions on the relaxed version of BOSCP with continuous variables. One may note, that there is a very large gap between the relaxed points and any of the approximations in the case of 2scp201D instance. It is not clear whether this gap is mainly due to the poor performance of metaheuristics on this instance or due to the gap between Pareto fronts of the original and relaxed problems. We have noted, however, that in the case of this instance solutions of the relaxed problem contain many fractional values. Thus, it is possible that the solutions are far from feasible solutions with binary values.

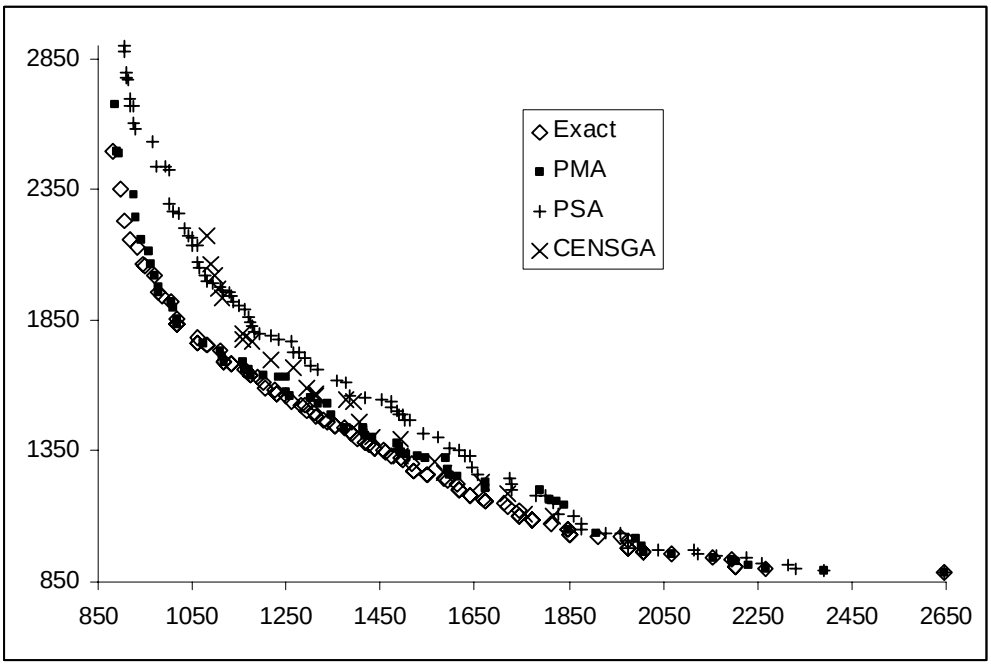

Figure 5. Exemplary approximations obtained for 2scp41A instance

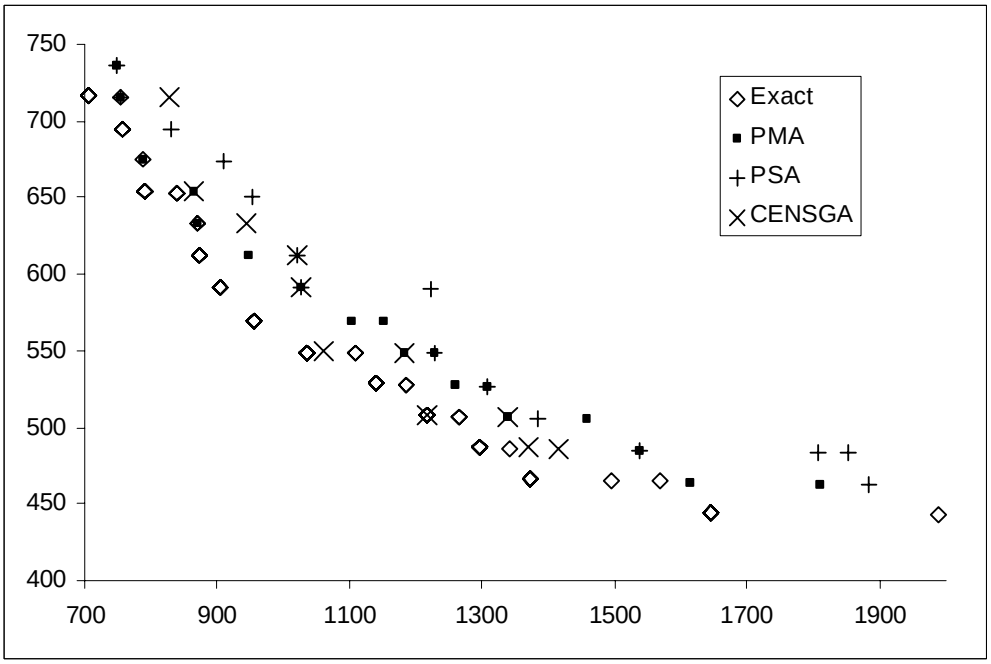

Figure 6. Exemplary approximations obtained for 2scp41C instance

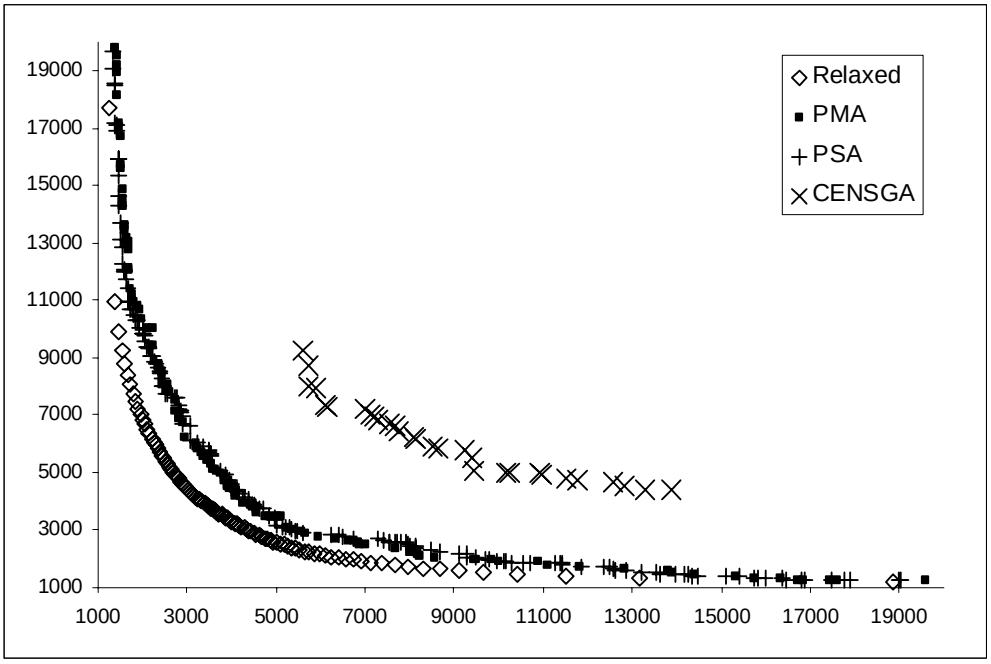

Figure 7. Exemplary approximations obtained for 2scp201B instance

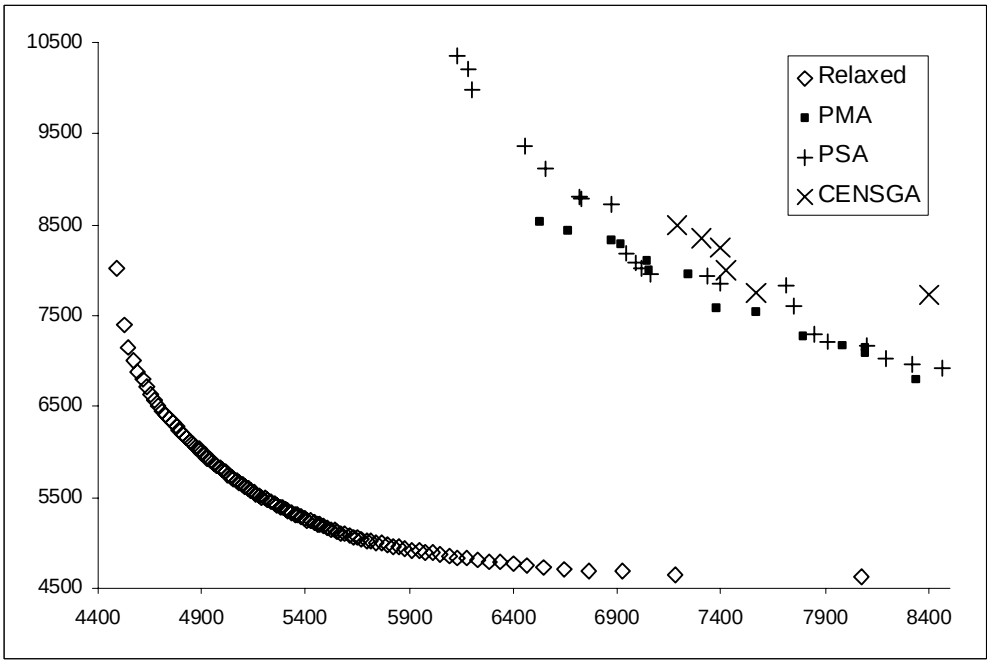

Figure 8. Exemplary approximations obtained for 2scp201D instance

9. Conclusions

Ten multiple-objective metaheuristics have been compared on the bi-objective set covering problem. The main conclusions of the experiment are:

- instances with repeating values of objectives coefficients are more difficult for both IP solvers and metaheuristics,

- PMA and MOGLS are the best performers in the experiment, although, they were not able to outperform all other methods on all instances.
- comparison of PMA and MOGLS with IMMOGLS and MOMSLS indicates that the use of recombination operator and recombination of good solutions is crucial for the performance of the former two algorithms,
- MOSA and PSA are capable of giving high quality results but are very sensitive to parameters setting,
- the use of a single generating solution in SMOSA results in very high dispersion of the quality of results,
- Pareto-ranking based genetic algorithms perform, in general, worse than other algorithms,
- the new mechanisms introduced in SPEA and CENSGA improve the results of basic Pareto-ranking based evaluation mechanism used in NSGA,
- CENSGA performs relatively well on small, difficult instances with repeating values of objectives coefficients.

Furthermore, the results indicate that the performance of multiple-objective metaheuristics may differ radically even if the methods are based on the same single objective algorithm and use exactly the same problem-specific operators. Thus, the elements of the methods that are specific to multiple-objective case have substantial influence on their performance.

In the future we would like to consider more advanced adaptations of the metaheuristics to the set covering problem taking into account the most effective approaches proposed in single objective case, e.g. [21]. We plan also to extend the experiments to instances with more than two objectives and to include other multiple-objective metaheuristics, e.g. methods based on tabu search [5] or Pareto-ranking based memetic algorithms [18].

10. References

[1] Beasly, J.E., Chu, P.C. (1996), *A Genetic Algorithm for the Set Covering Problem*. European Journal of Operational Research, **94**, 392-404.

[2] Czyżak P., Jaszkiewicz A. (1998), Pareto simulated annealing – a metaheuristic technique for multiple-objective combinatorial optimisation, *Journal of Multi-Criteria Decision Analysis*, **7**, 34-47.

[3] Deb K. , Goel T. (2001), Controlled elitist non-dominated sorting genetic algorithms for better convergence, in: E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, D. Corne (eds.) *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, **1993**, Springer, Berlin, 67-81.

[4] Eremeev, A.V. (1999), A genetic algorithm with a non-binary representation for the set covering problem, in: *Proc. of Operations Research (OR'98)*. Springer-Verlag, 175-181.

[5] Gandibleux X., Fréville A. (2000), Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: the two objectives case, *Journal of Heuristics*, **6**, 361-383.

[6] Garey M.R., Johnson D.S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.F. Freeman and Co., San Francisco.

[7] Galinier P., Hao J.-K. (1999), *Hybrid evolutionary algorithms for graph coloring*. *Journal of Combinatorial Optimization*, **3**, 4, 379-397.

[8] Hansen M.P., Jaszkiewicz A. (1998), *Evaluating the quality of approximations to the nondominated set*, Working paper, Institute of Mathematical Modelling Technical University of Denmark, IMM-REP-1998-7.

[9] http://lindo.com/

[10] http://www.frontsys.com/

[11] http://www-idss.cs.put.poznan.pl/~jaszkiewicz/MOMHLib/

[12] Ishibuchi H. Murata T. (1998), Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling, *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews*, **28**, 3, 392-403.

[13] Jaszkiewicz A. (2001), Comparison of local search-based metaheuristics on the multiple objective knapsack problem, *Foundations of Computing and Decision Sciences*, **26**, 1, 99-120.

[14] Jaszkiewicz A., Hapke M., Kominek P. (2001), Performance of multiple objective evolutionary algorithms on distribution system design problem – computational experiment, in: E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, D. Corne (eds.) *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, **1993**, Springer, Berlin, 241-255.

[15] Jaszkiewicz A. (2001), Multiple objective metaheuristic algorithms for combinatorial optimization, Habilitation thesis, 360, Poznan University of Technology, Poznan.

[16] Jaszkiewicz A. (to appear). Genetic local search for multiple objective combinatorial optimization, *European Journal of Operational Research*.

[17] Jaszkiewicz A. (to appear). On the performance of multiple objective genetic local search on the 0/1 knapsack problem. A comparative experiment, *IEEE Transactions on Evolutionary Computation*.

[18] Knowles J.D., Corne D.W. (2000), M-PAES: A Memetic Algorithm for Multiobjective Optimization, in *2000 Congress on Evolutionary Computation*, volume 1, Piscataway, New Jersey, IEEE Service Center, 325-332.

[19] Kwan, A.S.K., Kwan, R.S.K., Wren, A. (1999), Driver scheduling using genetic algorithms with embedded combinatorial traits, in: Wilson, N.H.M. (Ed.), *Computer-Aided Transit Scheduling*, Springer-Verlag, 81-102.

[20] Lourenço H.R., Paixão J.P., Portugal R. (2001), *The Crew-Scheduling Module in the GIST System*, Working Paper 547, Departament d'Economia i Empresa, Universitat Pompeu Fabra, Portugal.

[21] Marchiori E., Steenbeek A. (2000), An Evolutionary Algorithm for Large Scale Set Covering Problems with Application to Airline Crew Scheduling, in *Real World Applications of Evolutionary Computing*. Springer-Verlag, Lecture Notes in Computer Science, **1083**, 367-381.

[22] Merz P., Freisleben B. (1997), Genetic Local Search for the TSP: New Results, in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, IEEE Press, 159-164.

[23] Serafini P. (1992), Simulated annealing for multiple objective optimization problems, in: *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, Taipei 19-24.07, vol. 1, 87-96.

[24] Smith B. M., Wren A., A Bus Crew Scheduling System Using a Set Covering Formulation, *Transportation Research*, **22A**, 97-108, 1988.

[25] Srinivas N., Deb K. (1994), Multiple objective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation*, **2**, 2, 221-248.

[26] Steuer R.E. (1986), *Multiple Criteria Optimization – Theory, Computation and Application*, Wiley, New York.

[27] Taillard É. D. Comparison of iterative searches for the quadratic assignment problem, *Location science*, **3**, 87-105, 1995.

[28] Ulungu E.L., Teghem J., Fortemps Ph., Tuyttens D. (1999), MOSA method: a tool for solving multiobjective combinatorial optimization problems, *Journal of Multi-Criteria Decision Analysis*, **8**, 221-236.

[29] Van Veldhuizen D.A. (1999), *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, PhD thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

[30] Zitzler E., Thiele L. (1999), Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Evolutionary Algorithm, *IEEE Transactions on Evolutionary Computation*, **3**, 4, 257-271.