On the performance of multiple objective genetic local search on the 0/1 knapsack problem. A comparative experiment.

Andrzej Jaszkiewicz[*]

Working paper
RA-002/2000

July 2000

[*] 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

Multiple objective metaheuristics, e.g. multiple objective evolutionary algorithms, constitute one of the most active fields of multiple objective optimization. Since 1985 a significant number of different methods have been proposed. However, only few comparative studies of the methods were performed on large-scale problems. In this paper, we continue the comparative experiment of Zilter and Thiele described in [21]. We compare performance of multiple objective genetic local search to Strength Pareto Approach on the multiple objective 0/1 knapsack problem. The latter method was the best performer among five multiple objective evolutionary algorithms evaluated in the experiment of Zilter and Thiele. The results of our experiment indicate that multiple objective genetic local search is very effective in generating good approximations to the nondominated set.

Index Terms—Multiple objective optimization, evolutionary algorithms, metaheuristics, knapsack problem.

I. INTRODUCTION

Real-life optimization problems often require taking into account different points of view corresponding to multiple, often conflicting objectives. At the same time, computational complexity of many real-life multiple objective problems makes the use of exact methods practically infeasible. As metaheuristics, e.g. genetic/evolutionary algorithms [6], simulated annealing [12] or tabu search [1] have been successfully applied to many single objective problems their adaptation to multiple objective case seems to be very promising direction of research.

Since the proposition of the Vector Evaluated Genetic Algorithm in 1985 [15] a significant number of different multiple objective metheuristics have been proposed. Unfortunately, although the authors of the methods usually claim that the algorithms can solve large-scale real-life problems, most of the computational experiments reported in the literature are performed on toy problems. For example among thirty test problems described in [20] (Appendix B) only six involve more than two variables. Furthermore, descriptions of few experiments give enough details allowing other researchers for comparison of their new results.

Zilter and Thiele [21] performed a comparative study of five different multiple objective evolutionary algorithms on multiple objective knapsack problem (MOKP). They compared four well-known multiple objective evolutionary algorithms (MOEAs), i.e. Vector Evaluated Genetic Algorithm [15], Haleja's and Lin's Genetic Algorithm [7], Niched Pareto Genetic Algorithm [9] and Nondominated Sorting Genetic Algorithm [16]. On the basis of the experiment the authors proposed a new method called the Strength Pareto Approach (SPEA) that outperformed the other methods.

In this study, we compare the best performer in the Zilter's and Thiele's experiment to multiple objective genetic local search [11]. We use the same set of MOKP instances and similar performance measures. In order to compare results of MOGLS to the results of SPEA we use the sets of approximately nondominated points generated by SPEA available in the Internet [23].

The paper is organized in the following way. In the next section some basic definitions are given. Multiple objective genetic local search algorithm is described in the third section. The fourth section contains description of the Strength Pareto Approach. The design of the experiment is presented in the fifth section. In the sixth section results of the experiment are presented and discussed. The last section contains conclusions.

II. PROBLEM STATEMENT AND BASIC DEFINITIONS

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

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

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

where: *solution* $\mathbf{x} = [x_1, ..., x_I]$ is a vector of *decision variables*, D is the set of feasible solutions.

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 $\mathbf{x}^1$ of dominates the image of $\mathbf{x}^2$.

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

An *approximation to the nondominated set* is a set A of 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*:

$$z_j^* = \max\left\{f_j(\mathbf{x}) \mid \mathbf{x} \in A\right\} \qquad j = 1,...,J.$$

The point $\mathbf{z}_*$ composed of the worst attainable objective function values in the nondominated set is called the *nadir point*.

$\mathbf{z}^{**}(A)$ denotes an approximation of the ideal point based on set A, i.e.

$$z_j^{**} = \max\left\{z_j \mid \mathbf{z} \in A\right\} \qquad j = 1,...,J\,.$$

Range equalization factors ([17], ch. 8.4.2) are defined in the following way:

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

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

Weighted Tchebycheff scalarizing functions are defined in the following way:

$$s_\infty\left(\mathbf{z}, \mathbf{z}^0, \Lambda\right) = \max_j\left\{\lambda_i\left(z_j^0 - z_j\right)\right\} = \max_j\left\{\lambda_i\left(z_j^0 - f_j(\mathbf{x})\right)\right\}, \tag{2}$$

where $\mathbf{z}^0$ is a reference point, $\Lambda = [\lambda_1,...,\lambda_J]$ is a weight vector such that $\lambda_j \geq 0\ \forall j$.

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

minimize α (P2)

s.t.

$\alpha \geq \lambda_i\left(z_j^0 - z_j\right) = \lambda_i\left(z_j^0 - f_j(\mathbf{x})\right) \qquad j = 1,...,J\,,$

$\mathbf{x} \in D\,.$

Each weighted Tchebycheff scalarizing function of this type has at least one global optimum (minimum) belonging to the set of efficient solutions. For each efficient solutions $\mathbf{x}$ there exists a weighted Tchebycheff scalarizing function s such that $\mathbf{x}$ is global optimum (minimum) of s ([17], 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*.

III. MULTIPLE OBJECTIVE GENETIC LOCAL SEARCH

Genetic Local Search (GLS) is a metaheuristic algorithm that hybridizes recombination operators with local search, or more generally, with other local improvement heuristics. Other frequently used names are Memetic Algorithms or Hybrid Genetic Algorithms. It is quite difficult to track the single origin of GLS. To our knowledge, the first description of GLS was published by Ackley [1], but similar algorithms were developed probably completely independently by several authors. Heuristics based on the GLS scheme often prove to be extremely efficient in combinatorial optimization. In particular, methods of this type were able to improve best known results on a number of well studied combinatorial optimization problems, e.g. Travelling Salesperson Problem [13], Graph Coloring Problem [4] and Quadratic Assignment Problem [18]. Thus, in our opinion, development of multiple objective versions of GLS is one of the most promising directions of research.

In this paper, we use multiple objective genetic local search (MOGLS) algorithm proposed by us in [11]. The algorithm was tested on multiple objective Travelling Salesperson Problem and outperformed MOGLS algorithm of Ishibuchi and Murata [10],

MOGLS based on the idea of MOSA method proposed by Ulungu et al. [19] and Pareto ranking-based evolutionary algorithm [3] proposed by Fonseca and Fleming. Below we describe main ideas of the method.

The goal of multiple objective metaheuristics is to generate good approximations to the nondominated set. Of course, the best possible approximation is the whole nondominated set itself. Note that all weighted Tchebycheff scalarizing functions have optima in the nondominated set and each nondominated point is an optimum of a weighted Tchebycheff scalarizing functions (see section II). Thus, finding the whole nondominated set is equivalent to finding optima of all weighted Tchebycheff scalarizing functions. Hence, we reformulate the goal of multiple objective metaheuristics as simultaneous optimization of all weighted Tchebycheff scalarizing functions. In fact, it is enough to consider all weighted Tchebycheff scalarizing functions with normalized weight vectors.

MOGLS implements the idea of simultaneous optimization of all weighted Tchebycheff scalarizing functions with normalized weight vectors by random choice of a scalarizing function optimized in each iteration. In other words, in each iteration, MOGLS tries to improve the value of a randomly selected scalarizing function. A single iteration of MOGLS consists of a single recombination of a pair of solutions and application of heuristic that improves locally the value of the current scalarizing function. An approximation of the ideal point composed of the best known values of each objective is used as the reference point. The reference point changes (improves) during the run of the method. In order to draw at random the scalarizing function, a normalized weight vector is drawn at random by the algorithm presented in Figure 1. The algorithm assures that weight vectors are drawn with uniform probability distribution $p(\Lambda)$, i.e. a distribution for which:

$$\forall \Psi' \subseteq \Psi \quad \int_{\Lambda \in \Psi'} p(\Lambda) d\Lambda \Big/ \int_{\Lambda \in \Psi} p(\Lambda) d\Lambda = V(\Psi') / V(\Psi)$$

where Ψ and Ψ' denote the set of all normalized weights and a subset of it, respectively; $V(\Psi)$ and $V(\Psi')$ are Euclidean hyper-volumes of Ψ and Ψ', respectively. In other words, the probability of drawing a weight vector belonging to Ψ' is proportional to the hyper-volume of Ψ'.

In our opinion, the main reason of the good performance of the single objective GLS is the method combines features of two good solutions. Offsprings produced in this way are good starting point for heuristic improvement. Thus, in each iteration of MOGLS two solutions being good on the current scalarizng function should be recombined. Analogously, MOGLS should combine featerues of solutions that are good on the current scalarizing function. In each iteration, our MOGLS algorithm constructs a temporary population composed of K solutions being the best among known solutions on the current scalarizng function. Then, two different solutions are drawn for recombination from the temporary population with uniform probability. The algorithm of MOGLS is summarized in Figure 2.

The outcome of MOGLS is an approximation to the nondominated set *PE* composed of all potentially nondominated points (and corresponding potentially Pareto-optimal solutions) generated by the method. The set of potentially nondominated points is also called secondary population [9]. Updating the set of potentially nondominated points *PE* with solution **x** consists of:

- adding **f(x)** to *PE* if no point in *PE* dominates **f(x)**,
- removing from *PE* all points dominated by **f(x)**.

In general, the scalarizng functions with random normalized weight vectors should be applied to normalized objective function values (see section II). The objectives could e.g. be normalized with the range equalization factors based on the objectives' ranges in *PE*. As in the case of instances used in this paper all objectives have approximately the same range, we do not use the normalization of objective values in this experiment.

The original idea was to store all generated solutions in the current set of solutions *CS* from which temporary populations are selected. Storing and handling all solutions would, however, be very time and memory consuming. Thus, set *CS* is organized as a queue of size $K \times S$. In each iteration the newly generated solution is added to the beginning of the queue if it is better than the worst solution in the temporary population and different to all solutions in the temporary population. If the size of the queue is bigger than $K \times S$ then the last solution from the queue is removed.

In [11] a method for setting the number of initial solutions was proposed. In this paper, however, we set this parameter to the same values as used by Zilter and Thiele in their experiment [21] (see section V.E).

$$\lambda_1 = 1 - \sqrt[J-1]{rand()}$$

$$\ldots$$

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

$$\ldots$$

$$\lambda_J = 1 - \sum_{l=1}^{J-1} \lambda_l$$

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

Parameters: K – size of the temporary population, S – number of initial solutions, stopping criterion
Initialization:
The set of potentially nondominated points PE:=∅
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**
Improve locally objective f_j starting from solution **x** obtaining **x**'
Add **x**' to the current set of solutions CS
Update set PE with **x**'
Generation of the initial set of solutions:
repeat S times
Draw at random a weight vector Λ
Construct randomly a new feasible solution **x**
Improve locally the scalarizng function $s_\infty\left(\mathbf{z}, \mathbf{z}^{**}(PE), \Lambda\right)$ starting from solution **x** obtaining **x**'
Add **x**' to the current set of solutions CS
Update set PE with **x**'
Main loop:
repeat
Draw at random a weight vector Λ
From CS select N different solutions being the best on scalarizing function $s_\infty\left(\mathbf{z}, \mathbf{z}^{**}(PE), \Lambda\right)$ forming temporary population TP
Draw at random with uniform probability two solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ from TP.
Recombine $\mathbf{x}_1$ and $\mathbf{x}_2$ obtaining $\mathbf{x}_3$
Improve locally the utility function u starting from solution $\mathbf{x}_3$ obtaining $\mathbf{x}_3$'
if $\mathbf{x}_3$' is better on $s_\infty\left(\mathbf{z}, \mathbf{z}^{**}(PE), \Lambda\right)$ than the worst solution in TP and different in the decision space to all solutions in TP **then**
Add $\mathbf{x}_3$' to the current set of solutions CS
Update set PE with $\mathbf{x}_3$'
until the stopping criterion is met

Figure 2. Algorithm of multiple objective genetic local search - MOGLS.

IV. STRENGTH PARETO APPROACH

The Strength Pareto Approach (SPA) is relatively new multiple objective metaheuristic proposed by Zilter and Thiele [21]. The method combines the best features of other multiple objective genetic algorithms with some new techniques. The method was the best performer in experiment run by the authors [21].

The main features of SPEA may be summarized as follows:

- It assigns the fitness to the solutions on the basis of the dominance relation.
- It stores the potentially Pareto-optimal solutions found so far in a secondary population.

- It performs clustering of the potentially Pareto-optimal solutions in order to reduce the number of solutions stored.
- All potentially Pareto-optimal solutions stored in the secondary population participate in the selection.
- It uses a new Pareto-based niching method to preserve diversity of the population.

In the experiment described in [21] Zilter and Thiele use tournament selection with replacement. Further details of the algorithm may be found in [21].

V. EXPERIMENT DESIGN

As was mentioned above the experiment described in this paper is a continuation of the experiment performed by Zilter and Thiele [21]. We use the same problem and the same instances. We compare the approximations to the nondominated sets generated by MOGLS to the results of SPEA. We use similar way of evaluating the approximations to the nondominated sets. The problem instances and the approximations to the nondominated sets generated by SPEA are available in the Internet [23]. Results of our experiment are also available at: www-idss.cs.put.poznan.pl./~jaszkiewicz/mokp/.

A. The multiple objective 0/1 knapsack problem

In this study we use the *multiple objective 0/1 knapsack problem* in the same version as used by Zilter and Thiele in their experiment [21]. The *J*-objective objective 0/1 knapsack is defined by a set of *I* items. With each of the items *J* profits and *J* weights are associated. Furthermore, *J* constraints corresponding to the knapasack capacities are imposed. Formally the problem is defined in the following way: Given a set of *I* items and a set of *J* knapsacks, with:

p_{ji} – profit of item *i* according to knapsack *j*,

w_{ji} – weight of item *i* according to knapsack *j*,

c_j – capacity knapsack *j*,

find a vector $\mathbf{x} = \{x_1,...,x_I\} \in \{0, 1\}^I$, that maximizes

$$f_j(\mathbf{x}) = \sum_{i=1}^{I} p_{ji} x_i, \qquad j = 1,...,J ,$$

s. t.

$$\sum_{i=1}^{I} w_{ji} x_i \le c_j, \qquad j = 1,...,J .$$

We use the same set of 9 instances of the problem that were used by Zilter and Thiele [21]. The instances have 2, 3 and 4 objectives, and 250, 500 and 750 items. The instances were generated randomly with uncorrelated profits and weights. The capacities of knapsacks are equal to half the total weight of items. In result, about 50% of items are expected to be in the optimal solutions.

Allowing variables x_i, i=1,..,I, to take continuous values in <0,1> one obtains *relaxed multiple objective 0/1 knapsack problem*. Note that all feasible solutions of the multiple objective 0/1 knapsack problem are also feasible solutions of the relaxed problem.

B. Adaptation of MOGLS to the multiple objective 0/1 knapsack problem

Alike Zilter and Thiele [21] we base adaptation of MOGLS to the multiple objective 0/1 knapsack problem on the work of Michalewicz and Arabas [14] for single objective case. The solution is naturally encoded by a binary string $\mathbf{x} \in \{0, 1\}^I$. Standard one-point crossover operator is used. As the crossover operator may produce many unfeasible codings Michalewicz and Arabas [14] proposed a *greedy repair procedure* that removes items from an unfeasible solution step by step. The items are removed in the increasing order of the profit/weight ratio. In other words, the method removes items that locally decrease the weight in the knapsack at the lowest cost of the objective function.

As SPEA uses dominance based fitness assignment instead of an objective function, Zilter and Thiele [21] sort the removed items in the increasing order of the maximum profit/weigh ratio over all knapsacks. There is no warranty, however, that this approach decreases the weights at the lowest cost of the solution's fitness.

In each iteration of MOGLS a certain scalarizing function is used. Thus, we use the value of this scalarizing function to guide the greedy repair procedure. Let $\mathbf{x}$ be an unfeasible solution. Let $\mathbf{x}_{\cdot i}$ be the solution obtained by removing item *i* from $\mathbf{x}$. The greedy procedure removes the item with lowest value of the following ratio:

$$\frac{s_\infty\left(\mathbf{f}(\mathbf{x}),\mathbf{z}^{**}(PE),\Lambda\right)-s_\infty\left(\mathbf{f}(\mathbf{x}_{-i}),\mathbf{z}^{**}(PE),\Lambda\right)}{\sum_{j=1}^{J}w_{ji}}=\frac{\max_j\left\{\lambda_i\left(z_j^0-f_j(\mathbf{x})\right)\right\}-\max_j\left\{\lambda_i\left(z_j^0-f_j(\mathbf{x}_{-i})\right)\right\}}{\sum_{j=1}^{J}w_{ji}}=$$

$$\frac{\max_j\left\{\lambda_i\left(z_j^0-f_j(\mathbf{x})\right)\right\}-\max_j\left\{\lambda_i\left(z_j^0-f_j(\mathbf{x})-p_{ji}\right)\right\}}{\sum_{j=1}^{J}w_{ji}}$$

In other words, the method removes items that locally decrease the weights in the knapsacks at the lowest cost of the current scalarizing function.

In addition we observe that only solutions with completely filled knapsacks, i.e. solutions that do not allow insertion of any other elements, can be Pareto-optimal. Thus, we introduce a *greedy insertion procedure* that fills the knapsacks. The procedure is based on the same idea as the repair method. Let **x** be a feasible solution. Let $\mathbf{x}_{+i}$ be the solution obtained by insertion of item *i* to **x**. The greedy insertion procedure inserts the item that fits in all the knapsacks and has the highest value of the following ratio:

$$\frac{s_\infty\left(\mathbf{f}(\mathbf{x}),\mathbf{z}^{**}(PE),\Lambda\right)-s_\infty\left(\mathbf{f}(\mathbf{x}_{+i}),\mathbf{z}^{**}(PE),\Lambda\right)}{\sum_{j=1}^{J}w_{ji}}=\frac{\max_j\left\{\lambda_i\left(z_j^0-f_j(\mathbf{x})\right)\right\}-\max_j\left\{\lambda_i\left(z_j^0-f_j(\mathbf{x}_{+i})\right)\right\}}{\sum_{j=1}^{J}w_{ji}}=$$

$$\frac{\max_j\left\{\lambda_i\left(z_j^0-f_j(\mathbf{x})\right)\right\}-\max_j\left\{\lambda_i\left(z_j^0-f_j(\mathbf{x})+p_{ji}\right)\right\}}{\sum_{j=1}^{J}w_{ji}}$$

In other words, the method inserts items that locally improve the current scalarizing function at the lowest cost of the weights in the knapsacks.

Summarizing, in each iteration, the following operations are performed. At first two solutions are selected and recombined by the standard one-point crossover. If the results of recombination is unfeasible the greedy repair procedure is invoked. Then, if the knapsacks are not completely filled the greedy insertion procedure is invoked.

In the ***Generation of the first approximation of the ideal point*** and ***Generation of the initial set of solutions*** phases of the MOGLS algorithm the greedy insertion procedure is applied to empty solutions.

C. Other methods used in the experiment

In order to test the importance of some elements of the MOGLS two additional methods were used in experiment. The first method is called greedy heuristic (GH). In each iteration of the method a scalarizing function is drawn at random and the greedy insertion procedure is applied to an empty starting solution. In other words, the method corresponds to the ***Generation of the initial set of solutions*** phase of MOGLS algorithm. The algorithm of GH is presented in Figure 3. The method is introduced in order to test the hypothesis that the use of recombination operator is important for the performance of MOGLS.

The second additional method is introduced in order to test the hypothesis that the way the solutions are selected for recombination is important for the performance of MOGLS. The algorithm of the method is almost the same as of the MOGLS (see Figure 2). The only difference is in the selection of the recombined solutions in the ***Main loop***. The two solutions are drawn at random with uniform probability from the whole current set of solutions *CS*. Note that the rules of updating *CS* remain exactly the same and depend on the temporary population. The newly generated solution is added to *CS* if it is better than the worst solution in the temporary population and different to all solutions in the temporary population (see section III). This method is denoted by MOGLS'.

Parameters: stopping criterion
Initialization:
The set of potentially nondominated points PE:=∅
The current set of solutions CS:=∅
Generation of the first approximation of the ideal point:
for each objective f_j
Construct a new empty solution **x** and apply the greedy insertion procedure improving objective f_j obtaining **x'**.
Add **x'** to the current set of solutions CS
Update set PE with **x'**
Main loop:
repeat
Draw at random a weight vector Λ
Construct a new empty solution **x** and apply the greedy insertion procedure the scalarizng function $s_\infty\left(\mathbf{z}, \mathbf{z}^{**}(PE), \Lambda\right)$ obtaining **x'**.
Add **x'** to the current set of solutions CS
Update set PE with **x'**
until the stopping criterion is met

Figure 3. Algorithm of greedy heuristic - GH.

D. Performance measures

One of the performance measures is the coverage of two approximations to the nondominated set. Let A and B be two approximations. The coverage measure is defined as:

$$C(A,B)=\frac{\left|\{\mathbf{z}''\in B\}\mid \exists \mathbf{z}'\in A:\mathbf{z}'\succeq\mathbf{z}''\right|}{|B|}.$$

The value $C(A, B) = 1$ means that all points in B are dominated by or equal to (covered by) some points in A. The value $C(A, B) = 0$ means that none point in B is covered by any point in A. Note that in general $C(A, B) \neq C(B, A)$. The same performance measure was used in [21].

As the second performance measure we use the estimation of the expected value of weighted Tchebycheff scalarizing function on approximation A over the set of normalized weight vectors Λ [8]:

$$R(A)=E\left(s_\infty^*\left(\mathbf{z}^0,A,\Lambda\right)\right)=\int_{\lambda\in\Psi} s_\infty^*\left(\mathbf{z}^0,A,\Lambda\right)p(\Lambda)d\Lambda,$$

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.

The upper approximations of the ideal points of particular instances were used as the reference points $\mathbf{z}^0$. For each instance, the components of the upper approximation of the ideal points were found by individual optimization of each objective in the relaxed multiple objective 0/1 knapsack problem. Note that as all feasible solutions of the multiple objective 0/1 knapsack problem are also feasible solutions of the relaxed problem. Thus, the optimum value of each objective on the relaxed problem is not worse than the optimum value of this objective on the original discrete problem. Individual optimization of objectives in the case of the relaxed problem requires solving relatively easy linear programming problem. In this experiment, Lingo solver was applied [22]. The values of the upper approximations of the ideal points are given in Appendix.

In order to estimate the expected value we use numerical integration, i.e. we calculate average value of the weighted Tchebycheff scalarizing function over a large sample of systematically generated normalized weight vector. We use all normalized weight vectors in which each individual weight takes on one of the following values: $\left\{l/k,\ l=0,...,k\right\}$, where k is a sampling parameter. With a combinatorial argument, we notice that this produces $\binom{k+J-1}{J-1}$ weight vectors. For example, for k=3 and J=3, we obtain the following set of 10 vectors: {[0,0,1], [0,1/3,2/3], [0,2/3,1/3], [0,1,0], [1/3,0,2/3], [1/3,1/3,1/3], [1/3,2/3,0], [2/3,0,1/3], [2/3,1/3,0], [1,0,0]}. The parameter k was set to 200 for bi-objective instances, to 50 for three-objective instances and to 25 for four objective instances. The resulting weight vectors may be found at www-idss.cs.put.poznan.pl./~jaszkiewicz/mokp/.

The above measure was not used in [21]. Instead the volume of the objective space covered by the points in approximation A was used. We do not use volume measure because [21] does not give details of the algorithm used to calculate this value in general

case. Figure 4 illustrates, however, that *R* measure is correlated (but not mathematical equivalent) to the volume between the reference point and the approximation *A*, which in turn is negatively correlated to the volume measure used in [21].

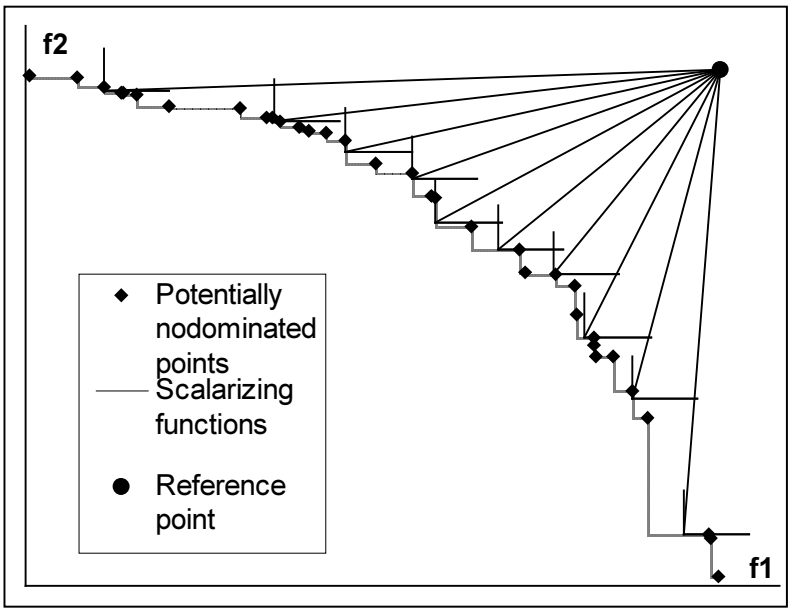

Figure 4. The idea of the expected value of weighted Tchebycheff scalarizing function measure.

Most measures used to evaluate approximations to the nondominated set allow relative comparison of different approximations only. They do not allow assessing the gap between the approximations and the real nondominated set unless this latter set is known. This is clearly the case of the coverage measure. This remark applies also to the volume measure used in [21]. As suggested in [8], lower bound of *R* measure allows estimation of the gap between an approximation and the real nondominated set. In order to calculate the lower bound of *R* measure in the case of the multiple objective 0/1 knapsack problem one can solve problem (P2) for each weighted Tchebycheff scalarizing function used in the numerical integration applying the constraints of the relaxed problem. This requires solving the following linear programming problem:

minimize α (P3)

s.t.

$$\alpha \geq \lambda_i \left(z_j^0 - z_j\right) = \lambda_i \left(z_j^0 - f_j(\mathbf{x})\right) =$$

$$\lambda_i \left(z_j^0 - \sum_{i=1}^{I} p_{ji} x_i \right), \qquad j = 1,...,J$$

$$\sum_{i=1}^{I} w_{ji} x_i \leq c_j, \qquad j = 1,...,J$$

$$0 \leq x_i \leq 1, \qquad i = 1,...,I\ .$$

As all feasible solution of the multiple objective 0/1 knapsack problem are also feasible solutions to the relaxed problem, the optimum value of each weighted Tchebycheff scalarizing function on the relaxed problem is not greater (not worse) than the optimum value of this function on the original discrete problem. Thus, the average of the optimum value of the weighted Tchebycheff scalarizing function on the relaxed problem is not greater (not worse) than the average on the original problem for any set of weight vectors. In this experiment, Lingo solver [22] was applied. Optima of weighted Tchebycheff scalarizing function on the relaxed problem correspond to points in the objective space. The points, in general, do not correspond to feasible solutions of the original problem. The points, however, are used in Figure 7.

The C++ codes used to calculate the two performance measures used in this experiment are available at www-idss.cs.put.poznan.pl./~jaszkiewicz/mokp/.

E. Parameter settings

The number of initial solutions S in both MOGLS and MOGLS' was set in the same way as the population size of SPEA in the experiment of Zilter and Thiele [21] (see Table 1). The size of the temporary population used by MOGLS was set equal to 20, i.e. for each randomly selected scalarizing function s the two recombined solutions were selected from 20 solutions being the best on s. In both MOGLS and MOGLS' the calculations were stopped after $50 \times S$ iterations. In other words, the number of recombinations corresponded to 50 generations of SPEA. Note that the approximations generated by SPEA in the experiment of Zilter and Thiele [21] were obtained after 500 generations. For each instance GH was allowed to construct exactly the same number of solutions as MOGLS and MOGLS'. The only difference is that the solutions were constructed by the greedy insertion procedure applied to empty starting solutions without recombination. For each instance 20 runs of each method were performed. As running times of GH were much higher than the running times of the MOGLS and MOGLS', GH was not run on the biggest instances with 750 items. In the case of MOGLS, the set of potentially nondominated points PE was saved not only at the end of each run but also after $20 \times S$ iterations (which corresponds to 20 generations). Below we present results of two versions of MOGLS. MOGLS20 refers to approximations saved after $20 \times S$ iterations and MOGLS50 refers to approximations saved after $50 \times S$ iterations.

Table 1. Numbers of initial solutions for different instances.

Instance		Number of initial solutions
Number of objectives	**Number of items**	
2	250	150
2	500	200
2	750	250
3	250	200
3	500	250
3	750	300
4	250	250
4	500	300
4	750	350

VI. RESULTS

The average running times of Pentium 350 MHZ CPU are given in Table 2. One can observe that the use of starting solutions obtained by recombination considerably reduces the time needed to obtain feasible solutions with completely filled knapsacks in comparison to GH that starts from empty solutions. MOGLS' is faster than MOGLS performing the same number of iterations. This was caused mainly by the fact that it generated less potentially nondominated points than MOGLS. Thus the time of updating the set of potentially nondominated points PE was lower. [21] does note give details of the running times of SPEA. We expect that MOGLS uses more time per iteration than SPEA. From the other side, MOGLS performed much less recombinations than SPEA (see section V.E).

Table 2. Average running times of the methods

Instance		Average running time [s]			
Number of objectives	**Number of items**	**GH**	**MOGLS'**	**MOGLS20**	**MOGLS50**
2	250	232.95	15.45	9.45	25.95
2	500	1176.40	40.90	33.00	65.15
2	750		81.30	81.00	134.90
3	250	846.05	32.50	20.70	62.50
3	500	1879.80	76.30	61.40	136.65
3	750		151.80	136.80	251.95
4	250	644.10	75.25	44.40	140.60
4	500	2753.16	152.55	101.35	253.70
4	750		290.05	207.45	428.25

The results of the evaluation with the coverage measure C are presented in Figure 5. A simple version of box plots is used. The thick line represents the median value, the upper and lower ends of the box are the upper and lower quartiles, and the ends of the vertical line are minimum and maximum value.

The results indicate that the approximations generated by SPEA are worse than the approximations generated by other methods. In particular points generated by SPEA cover none of the points generated by MOGLS20, MOGLS50 and MOGLS' on 3 and 4 objective instances. MOGLS50 achieves the highest values of the coverage measure. MOGLS20 also outperforms the other methods (naturally, expect of MOGLS50). The comparison of quality of approximations generated by MOGLS' and GH is not obvious.

In general the values of coverage measure *C* decrease with the growth of the number of objectives. It is natural to expect that in high dimensional space it is "easier" for a point to be nondominated. Note, however, that this pattern was not so evident in the results presented in [21].

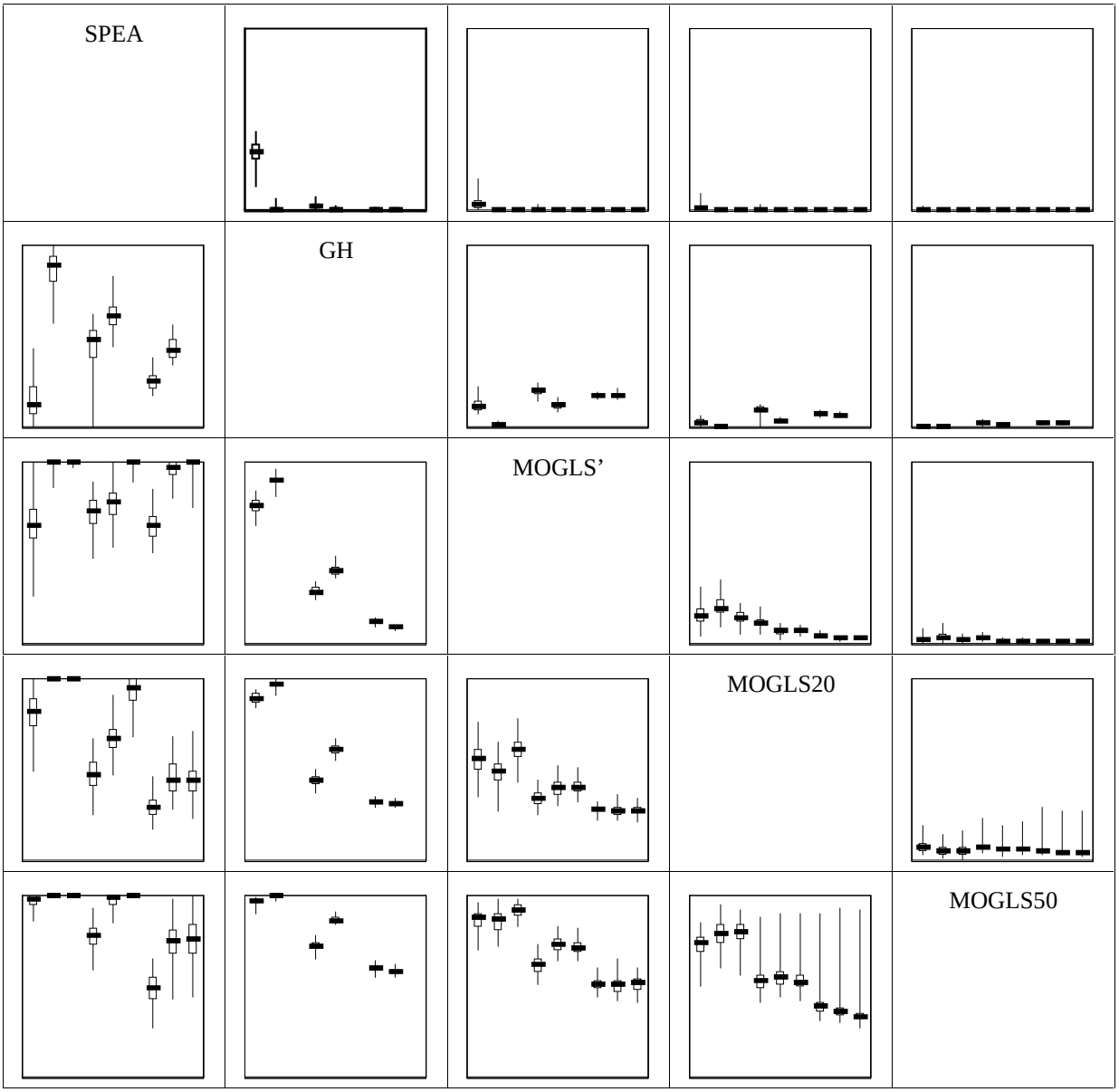

Figure 5. Results of the comparison with *C* measure. Each chart contains nine box plots representing the distribution of *C* values for a certain ordered pair of algorithms; the three box plots to the left relate to 2 knapsacks and (from left to right) 250, 500, and 750 items; correspondingly the three middle box plots relate to 3 knapsacks and the three to the right to 4 knapsacks. The scale is 0 at the bottom and 1 at the top for each chart. Chart in row of algorithm *A1* and column of algorithm *A2* presents values of coverage of approximations generated *A2* by approximations generated by *A1*.

The results of the evaluation with the *R* measure are presented in Figure 6. In general, the results allow drawing similar conclusions with respect to the relative performance of the methods as the results with *C* measure. Note, however, that the dispersion of results is very low. The values of evaluations of approximations generated by different methods almost never

overlap. Thus, *R* measure seems to be very precise tool for distinguishing quality of different methods. Note that the definition of *R* measure was motivated by its sensitivity to any evident improvement of the evaluated approximation (see [8]).

Again, SPEA is clearly the worst performer, while MOGLS50 generated the best approximations. Note that, the gap between SPEA and MOGLS is greater that the gap between MOGLS and the lower bound. Thus, MOGLS was able to generate approximations much closer to the real nondominated set than SPEA. The relative performance of MOGLS' deteriorates with increasing number of objectives. For example, MOGLS' outperformed GH on 2 and 3 objective instances but was outperformed on 4 objective instances.

The comparison of results of MOGLS and GH indicates that the use of recombination operator has crucial influence on the performance of MOGLS. GH method based solely of the greedy insertion procedure not only works much slower but also generates worse results.

The comparison of results of MOGLS and MOGLS' indicates that the recombination of solutions being good on the current scalarazing function is important for the performance of MOGLS. As the relative performance of MOGLS' deteriorates with increasing number of objectives, we may conclude that importance of this element of the algorithm grows with the same factor. MOGLS guarantees that only solutions being good on the current scalarazing function are recombined. Note that in the case of MOGLS' good solutions on the current scalarizing function may also be selected with some probability. The probability, however, decreases with the growing dimensionality of the objective space.

The detailed numerical results of the experiment are available in the Internet at www-idss.cs.put.poznan.pl./~jaszkiewicz/mokp/.

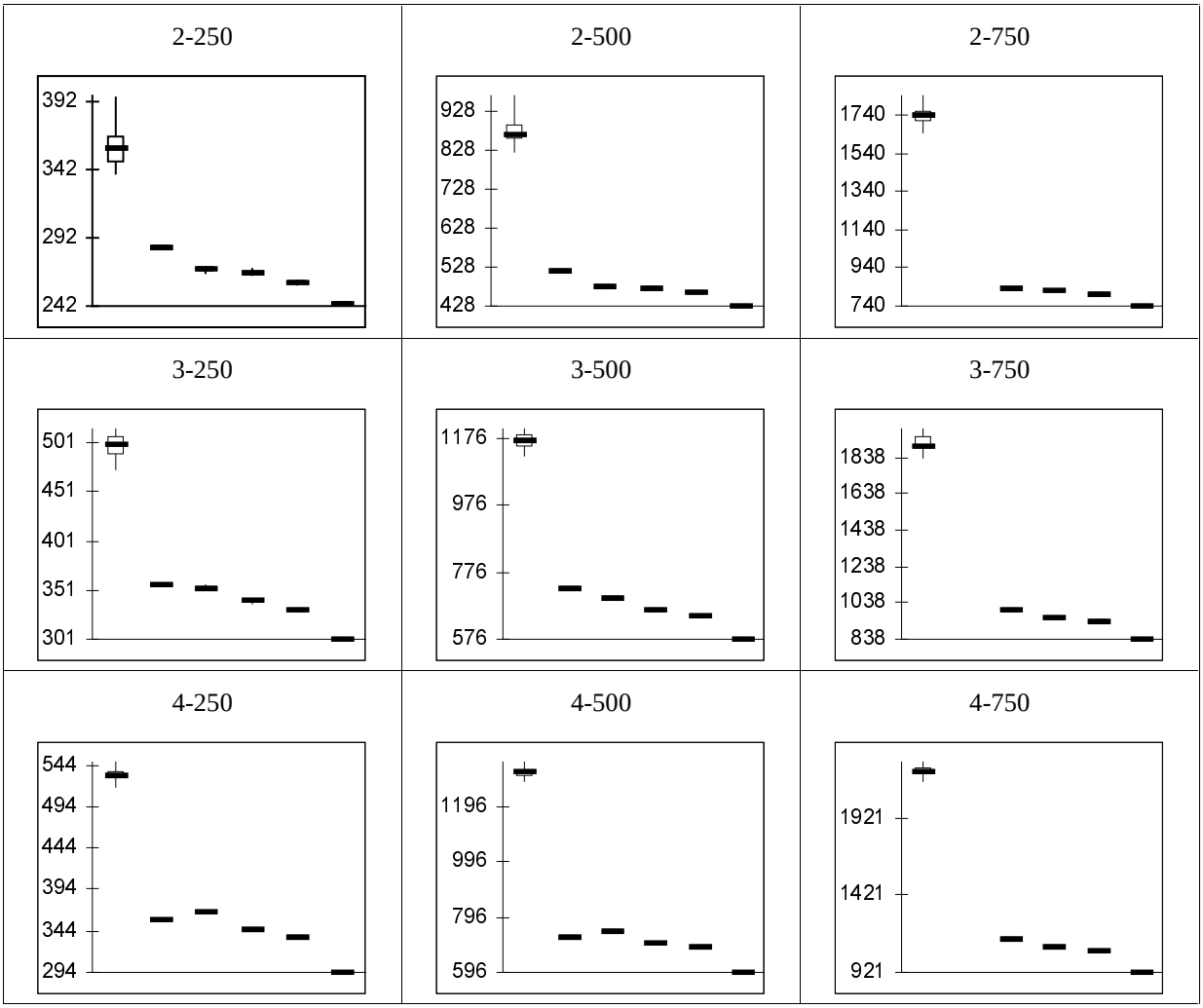

Figure 6. Results of the comparison with *R* measure. Each chart corresponds to different instance. Each chart contains six box plots representing the distribution of *R* measure for (from left to right) SPEA, GH, MOGLS', MOGLS20, MOGLS50 and the lower bound. For each chart the scale is from the lower bound at the bottom to the highest observed value at the top. Note that the box plots are practically invisible for all methods but SPEA because of very low dispersion of the results.

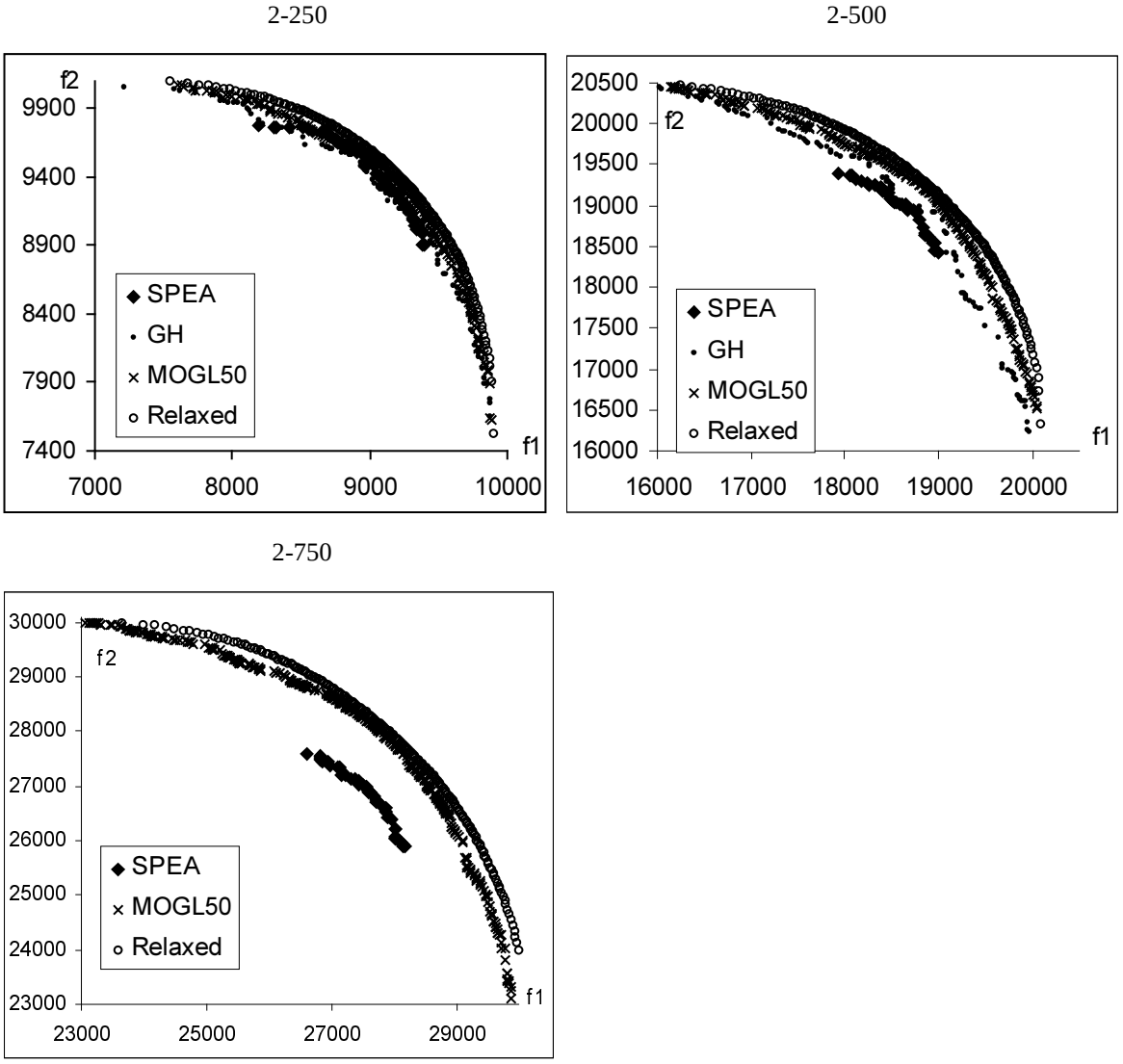

Figure 7. Nondominated points obtained for the bi-objective instances. Each set of points was obtained by a single (first) run of a given method. The set of points denoted by "relaxed" contains unattainable points obtained by optimization of the scalarizing functions on the relaxed problem.

VII. CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

In the presented experiment, multiple objective genetic local search algorithm was compared to Strength Pareto Approach on the multiple objective 0/1 knapsack problem. The results indicate that MOGLS was able to generate much better approximations to the nondominated sets than produced by SPEA – the best performer among five MOGAs evaluated in the experiment of Zilter and Thiele [21]. In addition, MOGLS generated the approximations performing much less recombinations than SPEA.

MOGLS was also compared to two other reference methods. Results of this comparison indicate that the use of the recombination operator and the appropriate selection of the solutions for recombination have significant influence on the performance of MOGLS.

Two different performance measures, based on different ideas, were used in this experiment. On average the results of comparisons with the two measures allow drawing similar conclusions with respect to the relative quality of the approximations generated by different methods. It is worth noting, however, that the results of the expected value of weighted Tchebycheff scalarizing function were characterized by much lower dispersion than the values of the coverage measure.

In this study, two simple repair and insertion greedy procedures were used to improve solutions produced by the recombination operator. We used this approach in order to preserve comparability with the works of Michalewicz and Arabas [14] and Zilter and Thiele [21]. The quality of results produced by MOGLS could probably be further improved by the use of a local search

algorithm that would exchange the items in completely filled knapsacks (see e.g. [2]). This would, however, increase the running time of the algorithm.

The approximations to the nondominated sets generated in this experiment, as well as the C++ codes used to calculate the performance measures are available in the Internet at www-idss.cs.put.poznan.pl./~jaszkiewicz/mokp/. This allows other researchers to compare results of new multiple objective methods.

REFERENCES

[1] Ackley D. H. *A connectionist machine for genetic hillclimbing*. Kluwer Academic Press, Boston, 1987.

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

[3] Fonseca C.M., Fleming P.J. On the performance assessment and comparison of stochastic multiobjective optimizers. *Lecture Notes in Computer Science*, **1141**, Springer Verlag, 584-593, 1996.

[4] Galinier P., Hao J.-K. *Hybrid evolutionary algorithms for graph coloring*. Technical Report, Parc Scientifique Georges Besse, Nimes, 1999.

[5] Glover F. Tabu Search - Part I, *ORSA J. Comput.* Vol. **1**, 190-206, 1989.

[6] Goldberg D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1988.

[7] Hajela P., Lin C.-Y. Genetic search strategies in multicriterion optimal design. *Structural Optimization*. Vol. **4**, pp. 99-107, June 1992.

[8] Hansen P.H, Jaszkiewicz A. Evaluating quality of approximations to the non-dominated set. *Technical Report, Department of Mathematical Modelling, Technical University of Denmark*, IMM-REP-1998-7, 1998.

[9] Horn J., Nafpliotis N., Goldberg D. E. A Niched Pareto Genetic Algorithm for Multiobjective Optimization, In: *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Volume 1, pp. 82-87, Piscataway, New Jersey, June 1994.

[10] Ishibuchi H. Murata T. Multi-Objective Genetic Local Search Algorithm and Its Application to Flowship Scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, **28**, 3, 392-403, 1998.

[11] Jaszkiewicz A. Genetic local search for multiple objective combinatorial optimization, *Technical Report RA-014/98, Institute of Computing Science, Poznan University of Technology*, 1998

[12] Laarhoven P.J.M., Aarts E.H.L. *Simulated Annealing: Theory and Applications,* Reidel, Dordrecht, 1987.

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

[14] Michalewicz Z. Arabas J. Genetic algorithms for the 0/1 knapsack problem. In: Raś, Z.W., Zemankova M. (eds.) *Methodologies for Intelligent Systems (ISMIS'94)*, Berlin, 1994, pp. 134-143.

[15] Schaffer J.D. Multiple objective optimization with vector evaluated genetic algorithms. In: J.J. Grefenstette (ed.), *Genetic Algorithms and Their Applications: Proceedings of the Third International Conference on Genetic Algorithms*, Lawrence Erlbaum, Hillsdale, NJ, 93-100, 1985.

[16] Srinivas N. Deb K. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, *Evolutionary Computation*, **2**(3), pp. 221-248, fall 1994.

[17] Steuer R.E. *Multiple Criteria Optimization - Theory, Computation and Application*, Wiley, New York, 1986.

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

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

[20] Van Veldhuizen D.A. *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, May 1999.

[21] Zitzler E., Thiele L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation*, Vol. **3**, No. 4, pp. 257-271, November, 1999.

[22] http://lindo.com/

[23] http://www.tik.ee.ethz.ch/~zitzler/testdata.html

APPENDIX. THE VALUES OF THE UPPER APPROXIMATIONS OF THE IDEAL POINTS

Instance		Upper approximation of the ideal point
Number of objectives	**Number of items**	
2	250	[9898.86, 10107.3]
2	500	[20096.5, 20494.6]
2	750	[30130.1, 30037.2]
3	250	[9981.99, 10085.5, 9774.19]
3	500	[19705.8, 20082.7, 20437.2]
3	750	[31197.7, 29772.1, 30519.4]
4	250	[9801.73, 9918.88, 10119, 9136.37]
4	500	[19825.8, 19568, 19805.8, 19592.3]
4	750	[29872.5, 29342.1, 30468.6, 30025.2]