Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 34 (2007) 3458–3470

computers & operations research

www.elsevier.com/locate/cor

MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem

Maria João Alves[a,*], Marla Almeida[b]

[a]*Faculty of Economics, University of Coimbra/INESCC, Av. Dias da Silva, 165, 3004-512 Coimbra, Portugal*
[b]*Rua 25 de Abril, 3140-222 Montemor-o-Velho, Portugal*

Available online 20 March 2006

Abstract

This paper presents a new multiobjective genetic algorithm based on the Tchebycheff scalarizing function, which aims to generate a good approximation of the nondominated solution set of the multiobjective problem. The algorithm performs several stages, each one intended for searching potentially nondominated solutions in a different part of the Pareto front. Pre-defined weight vectors act as pivots to define the weighted-Tchebycheff scalarizing functions used in each stage. Therefore, each stage focuses the search on a specific region, leading to an iterative approximation of the entire nondominated set.

This algorithm, called MOTGA (Multiple objective Tchebycheff based Genetic Algorithm) has been designed to the multiobjective multidimensional 0/1 knapsack problem, for which a dedicated routine to repair infeasible solutions was implemented. Computational results are presented and compared with the outcomes of other evolutionary algorithms.

Keywords: Genetic algorithms; Multiple objective programming; Knapsack problem

1. Introduction

Since the first developments on multiobjective metaheuristics in the 1980's, many metaheuristic approaches, namely evolutionary algorithms, have been proposed for dealing with multiobjective problems. In the recent years there has been a growing interest in this field, and many multiobjective metaheuristics have been developed and proved to be successful in practice. Most of the research on this area has been concentrated on evolutionary algorithms. Deb [1] and Coello [2] report several multiobjective evolutionary approaches and the site [3] includes a vast list of bibliographic references on this field. Also, the overview of Jones et al. [4] indicates that 70% of the articles utilize genetic algorithms as primary metaheuristic technique, 24% simulated annealing, and 6% tabu search. Most of the multiobjective metaheuristic algorithms are *generating* methods since they aim at generating an approximation of the entire set of the nondominated (Pareto-optimal or efficient) solutions, while few of them are *interactive* methods (e.g. [5–7]).

Concerning evolutionary algorithms, two classes of approaches can be basically distinguished: those that are based on the dominance relation and those that are based on scalarizing functions. The former type, which has been more

* Corresponding author.
E-mail address: mjalves@fe.uc.pt (M.J. Alves).

0305-0548/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2006.02.008

often considered, uses the dominance relation to primarily base the fitness assignment. They usually incorporate other sub-criteria to provide a more fine-grained discrimination of the individuals (solutions). The second type of approaches considers scalarizing functions that temporarily aggregate the several objective functions. Depending on the approach, a scalarizing function can be stable during several generations or can vary from one iteration to the next of the algorithm. In the design of a *generating* algorithm, the scalarizing functions are just tools used in the search for approximate nondominated solutions, so their use has not to do with modeling the decision maker's preferences in the form of a single aggregate function [8].

In this paper, we propose a new multiobjective genetic algorithm, which is based on the Tchebycheff scalarizing function. For short, we shall designate by MOTGA this Multiple Objective Tchebycheff-based Genetic Algorithm.

The main purpose of MOTGA is to create a good approximation set of the nondominated solution set, assuring both convergence and diversity of the obtained solutions. The current version of the algorithm is devoted to the multiobjective multidimensional 0/1 knapsack problem (MOMKP) since a specialized routine to repair infeasible solutions has been implemented. Nevertheless, the algorithm employs a general schema that can be easily adapted to other multiobjective linear programs with binary variables.

The MOMKP has been widely studied by other authors and several experimental comparisons of different meta-heuristic approaches on MOMKP have been reported in the literature. Some problem instances and results of those studies are available in the Internet. These benchmarks can be very useful to validate new meta-heuristics.

Zitzler and Thiele [9] performed a comparative study of five different multiple objective evolutionary algorithms on MOMKP, and also introduced a new method called Strength Pareto Evolutionary Algorithm (SPEA) that outperformed the other methods. Knowles and Corne [10] used the same set of MOMKP instances to compare the Memetic Pareto Archived Evolution Strategy (M-PAES), of their own, with the Random Directions Multiple Objective Genetic Local Searcher (RD-MOGLS) algorithm of Jaszkiewicz [11]. The authors concluded that, although both algorithms performed well, M-PAES was found to be superior overall on those experiments, and its performance improved relatively to RD-MOGLS with increasing problem size (number of items). Nonetheless, RD-MOGLS seemed capable of generating solutions over a wider range in each of the objectives than M-PAES, and on the four-objective problems there was no discernible difference in performance between the two algorithms.

Also, several comparative studies of different algorithms on the same set of MOMKP instances were made by Jaszkiewicz. In [8], Jaszkiewicz compared five algorithms: the Multiple-Objective Genetic Local Searcher (MOGLS), proposed by the author, M-PAES [10], the Serafini's Multiple Objective Simulated Annealing (SMOSA) [12], the Multiple Objective Simulated Annealing (MOSA) of Ulungu et al. [13] and the Pareto Simulated Annealing (PSA) developed by Czyzak and Jaszkiewicz [14]. The author concluded that the results of those experiments indicated the best performance for MOGLS and PSA on the tested MOMKP instances. He also pointed out that the methods based on scalarizing functions were able to produce points in all regions of the nondominated set, whereas the use of the dominance relation assured convergence towards the nondominated set but did not guarantee dispersion over all regions of the set. Jaszkiewicz presented other comparative experiments: in [15], MOGLS was compared with SPEA, and in [16] a comparison was made between MOGLS, SPEA, M-PAES and Ishibuchi and Murata's multiple-objective genetic local search (IMMOGLS) [17]—in both comparative experiments the author concluded that MOGLS outperformed the other methods; in [18] it was made a comparison of the computational efforts needed to generate solutions of MOMKP of approximately the same quality by two kinds of methods: multiple and single objective metaheuristics.

An improved version of the SPEA algorithm, called SPEA2, was further developed by Zitzler et al. [19]. It has been tested on different problems, including a subset of the MOMKP set proposed in [9], and the authors concluded that SPEA2 performed better than its predecessor on all problems.

In our study, we consider the same set of MOMKP instances to test MOTGA and compare it with the following algorithms: SPEA (which had the highest performance in [9]), SPEA2 (which has revealed to be superior to the previous one, but the comparison will be restricted to the few instances that were considered in [19]), and MOGLS (which has proved to be very successful for solving MOMKP and presented the best performance in the experiments reported in [15] and [16]).

The paper is organized as follows: in Section 2, the MOMKP is defined and some basic concepts concerning multiobjective (integer) programming are introduced. The algorithm we propose (MOTGA) is described in Section 3. Section 4 shows computational results and an experimental comparison with other approaches. Finally, some concluding remarks are given in Section 5.

2. Basic definitions

The multiobjective multidimensional 0/1 knapsack problem (MOMKP), with p objective functions, m constraints and n binary decision variables (items) can be stated as follows:

$$\max \quad f_k(\mathbf{x}) = \sum_{j=1}^{n} q_{kj} x_j, \quad k = 1, \ldots, p \tag{MOMKP}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_{ij} x_j \leqslant c_i, \quad i = 1, \ldots, m,$$
$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n,$$

where q_{kj} is the performance measure (e.g. profit) of item j according to the objective k, w_{ij} is the coefficient (e.g. weight) of item j in the knapsack constraint i, and c_i is the capacity of the knapsack i. It is assumed that all coefficients q_{kj}, w_{ij}, and c_i are nonnegative. Let X denote the feasible region and Z its image on the objective space, i.e. the set of points $\mathbf{z} \in \Re^p$ such that $z_k = f_k(\mathbf{x})$, $k = 1, \ldots, p$, $\mathbf{x} \in X$.

As usually in multiobjective programming, a feasible solution that cannot be improved in any objective without degradation in another is called *nondominated*, *efficient* or *Pareto-optimal* solution. Although the terms *efficient* and *Pareto-optimal* are more often used for points $\mathbf{x} \in X$ and the term *nondominated* for points $\mathbf{z} \in Z$, they can also be used interchangeably.

The dominance relation is defined as follows: $\mathbf{z}^2$ *dominates* $\mathbf{z}^1$, $\mathbf{z}^2 \succ \mathbf{z}^1$, iff $z_k^2 \geqslant z_k^1$ for all k and $z_k^2 > z_k^1$ for at least one k. Hence, $\mathbf{z}^1 \in Z$ is nondominated ($\mathbf{x}^1 \in X$ is efficient or Pareto-optimal) iff there is no other $\mathbf{z}^2 \in Z$ that dominates $\mathbf{z}^1$. A solution $\mathbf{z}^1 \in Z$ is said to be *weakly nondominated* iff there is no other $\mathbf{z}^2 \in Z$ such that $z_k^2 > z_k^1$ for all k.

The MOMKP is a particular case of the multiobjective integer linear program, inheriting its mathematical properties. Since X is nonconvex, the problem admits not only *supported* Pareto-optimal solutions but also *unsupported* Pareto-optimal solutions, i.e. solutions that do not belong to frontier of the convex hull of the feasible region. These solutions cannot be captured by optimizing weighted-sums of the objective functions (see, e.g. [20] or [21, Chapter 14]). Therefore, reference point scalarizing functions (which project reference points onto the nondominated solution set) seem more suitable to deal with multiobjective integer programming problems, and they have been widely used in the development of methods (namely interactive methods) for this type of problems. For more information on this topic, we refer to Wierzbicki [22] and the references therein.

The Tchebycheff scalarizing function (whose theoretical foundations for multiobjective integer linear programming have been assigned to Bowman [23]) integrates this type. Let $\|\mathbf{z}^0 - \mathbf{f}(\mathbf{x})\|_{\boldsymbol{\lambda}}$ denote the $\boldsymbol{\lambda}$-weighted Tchebycheff distance from $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_p(\mathbf{x}))$ to $\mathbf{z}^0$, a reference point on the objective space such that $\mathbf{z}^0 \geqslant \mathbf{f}(\mathbf{x})$, $\forall \mathbf{x} \in X$. Hence, $\|\mathbf{z}^0 - \mathbf{f}(\mathbf{x})\|_{\boldsymbol{\lambda}} = \max_{k=1,\ldots,p} \{\lambda_k (z_k^0 - f_k(\mathbf{x}))\}$. The optimal solution of $\min_{\mathbf{x} \in X} \|\mathbf{z}^0 - \mathbf{f}(\mathbf{x})\|_{\boldsymbol{\lambda}}$ is at least a weakly nondominated solution, and the parameterization on $\boldsymbol{\lambda} \in \Lambda = \{(\lambda_1, \ldots, \lambda_p)\colon \lambda_k \geqslant 0, \forall k, \sum_k \lambda_k = 1\}$ of $\min_{\mathbf{x} \in X} \|\mathbf{z}^0 - \mathbf{f}(\mathbf{x})\|_{\boldsymbol{\lambda}}$ generates all the nondominated solutions (for a proof, see [21, Chapter 14]).

The reference point may represent aspiration levels for the objective function values. The *ideal solution* $\mathbf{z}^*$, which is composed of the maxima individual values for the objective functions, has been widely used as reference point: $z_k^0 = z_k^* = \max_{\mathbf{x} \in X} \{f_k(\mathbf{x})\}$, $k = 1, \ldots, p$.

Let us denote by $\mathrm{F}(\boldsymbol{\lambda}, \mathbf{x})$ the $\boldsymbol{\lambda}$-weighted Tchebycheff distance to the ideal solution, i.e. $F(\boldsymbol{\lambda}, \mathbf{x}) = \|\mathbf{z}^* - \mathbf{f}(\mathbf{x})\|_{\boldsymbol{\lambda}}$. As mentioned above for a generic $\mathbf{z}^0$, the minimization of $F(\boldsymbol{\lambda}, \mathbf{x})$ over X may yield weakly nondominated solutions. However, they can be avoided by optimizing the augmented Tchebycheff function, which can be formulated as: $\min_{\mathbf{x} \in X} \{F(\boldsymbol{\lambda}, \mathbf{x}) - \rho \sum_k f_k(\mathbf{x})\}$ with ρ small positive. This advantage of the augmented Tchebycheff function is rather more relevant to an exact method than to a heuristic one. Therefore, for simplicity reasons, we consider in our algorithm the simple weighted Tchebycheff function instead of the augmented one.

The minimization of $F(\boldsymbol{\lambda}, \mathbf{x})$ over X can also be stated in the following way:

$$\begin{aligned} \min \quad & \alpha \\ \text{s.t.} \quad & \lambda_k (z_k^* - f_k(\mathbf{x})) \leqslant \alpha, \quad k = 1, \ldots, p, \\ & \mathbf{x} \in X, \\ & \alpha \geqslant 0. \end{aligned} \tag{$T(\boldsymbol{\lambda})$}$$

Besides, let $T_{\mathrm{LR}}(\boldsymbol{\lambda})$ denote the linear relaxation of $T(\boldsymbol{\lambda})$, that is the program resulting by discarding from X the integer conditions on $\mathbf{x}$ and considering just $\mathbf{0} \leqslant \mathbf{x} \leqslant \mathbf{1}$. $T_{\mathrm{LR}}(\boldsymbol{\lambda})$ is used in the algorithm proposed in the next section for computing initial solutions. $F(\boldsymbol{\lambda}, \mathbf{x})$ is used to fitness assignment during the genetic process.

3. The MOTGA algorithm

MOTGA aims at generating a set of potentially nondominated solutions (an *approximation set*) that provides a good approximation of the *true* nondominated solution set (also known as *Pareto front*) of the multiobjective problem.

The MOTGA algorithm works with two populations: the internal population, P, which participates in the genetic process, and the external population, P_{nd}, which stores all the potentially nondominated solutions found through the algorithm. The outcome of the algorithm is P_{nd}, which is called the nondominated population.

The algorithm makes use of a set of pre-defined weights $\boldsymbol{\lambda}$ well dispersed over the weight space Λ, which act as *pivots* to generate other weight vectors that are inputted into the weighted Tchebycheff scalarizing function. The different weight vectors generated will guide the search towards different parts of the nondominated set, throughout a multi-stage process with many stages as the number of *pivot* weight vectors defined. Each stage focuses on a particular region of the Pareto front, so that the merge of the results produced in all stages could provide a good approximation of the entire nondominated solution set.

We have been considering $2p + 1$ *pivot* weight vectors (where p is the number of objective functions) defined as in the *Interval Criterion Weights* method of Steuer [21]: $\boldsymbol{\lambda}^1 = (1, 0, \ldots, 0), \ldots, \boldsymbol{\lambda}^p = (0, 0, \ldots, 1)$, $\boldsymbol{\lambda}^{p+1} = (1/p^2, v, \ldots, v), \ldots, \boldsymbol{\lambda}^{2p} = (v, v, \ldots, 1/p^2)$, $\boldsymbol{\lambda}^{2p+1} = (1/p, 1/p, \ldots, 1/p)$, with $v = (p+1)/p^2$. Therefore, each run of the algorithm executes $2p + 1$ stages, each one using a different *pivot* weight vector $\boldsymbol{\lambda}^r$ $(r = 1, \ldots, 2p + 1)$. Note, however, that other dispersed weight vectors could be used as *pivots*.

Let us consider the rth stage: G iterations (generations) of a genetic process are performed, in which the individuals are assessed by a fitness evaluation function—the weighted Tchebycheff function—which varies from one iteration to the next according to a parameter vector of weights that slightly changes "around" the *pivot* vector $\boldsymbol{\lambda}^r$. Thus, in the iteration i of the stage r, a weight vector $\boldsymbol{\lambda}^{r(i)}$ is drawn at random in a neighborhood of $\boldsymbol{\lambda}^r$ in the following way: $\delta_1, \delta_2, \ldots, \delta_p$ are randomly generated in the interval $]-\Delta, \Delta[$, with Δ a small positive value, to produce $\boldsymbol{\mu} = (\lambda_1^r + \delta_1, \lambda_2^r + \delta_2, \ldots, \lambda_p^r + \delta_p)$; after setting to 0 all the negative components of $\boldsymbol{\mu}$, this vector is normalized in order to $\sum_{k=1}^{p} \mu_k = 1$; the resulting vector is then assigned to $\boldsymbol{\lambda}^{r(i)}$. The weight vector $\boldsymbol{\lambda}^{r(i)}$ is used in the genetic selection process, in which the fitness of a solution $\boldsymbol{x}$ is given by $F(\boldsymbol{\lambda}^{r(i)}, \mathbf{x}) = \max_{k=1,\ldots,p} \{\lambda_k^{r(i)}(\bar{z}_k^* - f_k(\boldsymbol{x}))\}$, where $\bar{\mathbf{z}}^*$is the ideal solution of the linear relaxation of the multiobjective problem. Since $F(\boldsymbol{\lambda}^{r(i)}, \mathbf{x})$ is a minimizing function, a lower value implies an higher probability of $\mathbf{x}$ for reproduction.

In our study, we have been considering $\Delta = 1/(2p^2)$. Fig. 1 shows the *pivot* weight vectors used in bi-objective problems, and also the area of weight vectors that is "covered" by each stage (i.e. the weight vectors that can be generated in each stage) considering $\Delta = 1/(2p^2)$. Note that this value of Δ implies that any weight vector belonging to the weight space (Λ) can be generated, because the areas of weight vectors "covered" by the stages are contiguous (see Fig. 1). This is not true, however, in cases with more than 2 objective functions.

To start the search within each stage, the internal population is reset, and a set of N initial solutions is generated in the following way: N weight vectors $\boldsymbol{\beta}^{t \in \Lambda}$ $(t = 1, \ldots, N)$ are randomly generated, and the linear program $T_{\mathrm{LR}}(\boldsymbol{\beta}^t)$ is solved for all $\boldsymbol{\beta}^t (t = 1, \ldots, N)$; the optimal solution of $T_{\mathrm{LR}}(\boldsymbol{\beta}^t)$ is a nondominated solution for the linear relaxation of the multiobjective problem (the closest one to $\bar{\mathbf{z}}^*$, according to the $\boldsymbol{\beta}^t$-weighted Tchebycheff metric); this solution is then rounded and repaired (if it is infeasible) and introduced in the internal population P. At this point, we should stress that other ways for defining an initial population were designed and experimented. For instance, one of them consisted in picking a subset (of size N) of solutions from the external population P_{nd} (except at the first stage). In this case, the first stage used the central *pivot* weight vector $\boldsymbol{\lambda}^{2p+1} = (1/p, 1/p, \ldots, 1/p)$ (instead of $\boldsymbol{\lambda}^1$) to try to generate a "central" population P_{nd}. Different ways for selecting solutions from P_{nd} were then experimented: (a) selecting the N best solutions of P_{nd} according to the weighted Tchebycheff function for the *pivot* weight vector used in that stage; (b) performing a tournament selection evaluated by the same function; (c) selecting randomly N solutions from P_{nd}. However, these modifications seemed to not improve the algorithm, as the final approximation sets produced for the tested MOMKP instances were, in general, worse than those obtained with the adopted process explained above.

In sum, the achievement of convergence is the basis on adopting the type of strategy we propose, where several stages are performed, each one mainly concentrated on a particular region of the Pareto front. The preoccupation of

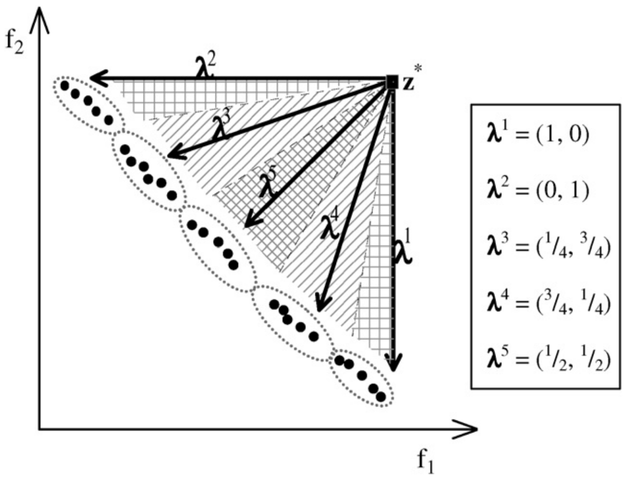

Fig. 1. *Pivot* weight vectors used in bi-objective problems and the area of weight vectors "covered" by each stage, considering $\Delta = 1/(2p^2)$.

diversity is also expressed in two ways: (1) by considering random weight vectors ($\boldsymbol{\beta}$) to generate an initial population for each stage; (2) by slightly changing the weight vector throughout one stage (around a *pivot*), which is used to guide the search for potentially nondominated solutions in a particular region. This avoids a bias towards solutions that are close to the optimum of one predefined weighted Tchebycheff function.

An overall description of MOTGA can be stated as follows:

```
Parameters:     $N$—size of the internal population $P$.
                $G$—number of generations in each stage.
                probC, probM—probabilities of crossover and mutation.
Initialization: Initialize the nondominated population: $P_{\mathrm{nd}} := \emptyset$
                Determine $\bar{\mathbf{z}}^*$ (the ideal solution for the linear relaxation of the problem).
                Define the pivot weight vectors: $\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^{2p+1}$
For each $\boldsymbol{\lambda}^r$ $(r = 1, \ldots, 2p+1)$ do   (* execute stage $r$*)
    $P := \emptyset$
    Compute the initial population $P$, of size $N$:
        Draw at random $N$ weight vectors from $\Lambda$ : $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^N$
        For each $\boldsymbol{\beta}^t$ $(t = 1, \ldots, N)$ do
            • optimize the linear program $T_{\mathrm{LR}}(\boldsymbol{\beta}^t)$, obtaining a continuous solution $\tilde{\mathbf{x}}^t$
            • round $\tilde{\mathbf{x}}^t$ to obtain $\mathbf{x}^t$
            • apply the routine repair_improve($\mathbf{x}^t, \boldsymbol{\beta}^t$) to repair $\mathbf{x}^t$, if it is infeasible, and to try to improve it
            • insert $\mathbf{x}^t$ in $P$
        End for
    Update $P_{\mathrm{nd}}$ with the nondominated solutions of $P$.
    Repeat G times   (* Genetic process of stage $r$, using the pivot weight vector $\boldsymbol{\lambda}^r$ *)
        Generate $\boldsymbol{\lambda}^{r(i)}$ in a neighborhood of $\boldsymbol{\lambda}^r$
        Execute a genetic iteration, guided by $\boldsymbol{\lambda}^{r(i)}$:
            • apply tournament selection in $P$, using $F(\boldsymbol{\lambda}^{r(i)}, \mathbf{x})$ to fitness assignment, forming a temporary
            population $P'$,
            • apply crossover in $P'$ with probability probC; replace in $P'$ the parents by the offspring,
            • apply mutation in $P'$ with probability probM,
            • apply the routine repair_improve($\mathbf{x}, \boldsymbol{\lambda}^{r(i)}$) to each new $\mathbf{x} \in P'$,
            • $P := P'$
        Update $P_{\mathrm{nd}}$ with the new nondominated solutions of $P$.
    End Repeat
End For
Output: $P_{\mathrm{nd}}$.
```

Let us now present the *genetic process* with more details.

The *genetic iteration* i of the stage r is guided by $\boldsymbol{\lambda}^{r(i)}$ and it is characterized by the application in the current population P of tournament selection, and usual operators of crossover and mutation for binary strings (solutions $\mathbf{x}$). In the selection scheme, $N/2$ tournaments are played between two solutions of P and the better solution according to $F(\boldsymbol{\lambda}^{r(i)}, \mathbf{x})$ is chosen and copied to a temporary population P'. A single-point crossover is then applied to strings of P' in the following way: pairs of strings (parent solutions) are successively picked at random from P' (without replacement); a total of $probC \times N/2$ pairs of solutions are picked; for each pair, a random crossover point is chosen and the corresponding sub-strings are exchanged; the two new strings are inserted into an offspring population, P_c. Afterwards, the offspring solutions in P_c are inserted into P' replacing their parents (which are discarded from P'). Thus, P' will include the offspring of reproduced solutions, and also the selected solutions (i.e. solutions obtained from the selection schema) that were not picked for reproduction. The bit-wise mutation operator is then applied to $probM \times N$ strings of P'. The infeasible solutions in P' are repaired and all the new solutions are attempted to be improved, using the routine *repair_improve* (with parameter $\boldsymbol{\lambda}^{r(i)}$, which provides a direction for the movements). P' is the new current population, so it is assigned to P. Note that the size of the current population P remains constant and equal to N, from one generation to the next (but repeated solutions may exist). The nondominated population P_{nd} is then updated with the new nondominated solutions of P, which means copying any new $\mathbf{x}$ of P to P_{nd} if no solution of P_{nd} dominates or is equal to it, and removing from P_{nd} all the solutions dominated by $\mathbf{x}$.

Both in the generation of an initial population and during the genetic process, the routine *repair_improve* is called to repair an infeasible solution and also to attempt to improve it. This routine implements a greedy method for the MOMKP, which removes/adds items from/to the solution, step by step. The weight vector inputted as parameter is used to guide these movements. The mechanism of repairing an infeasible solution consists of successively removing items (i.e. flipping the corresponding variables from 1 to 0) until all the capacity constraints are fulfilled. On the other side, the mechanism of improving a solution consists of adding items that do not violate any constraint. The algorithm of *repair_improve* for the MOMKP is the following:

```
Input: $\mathbf{x}$—solution; $\boldsymbol{\lambda}$—weight vector
(* Repair an infeasible solution *)
While $\mathbf{x}$ is not feasible do
    $V := \{i \mid \text{constraint } i \text{ is violated}\}$
    For each $x_j = 1$ compute $Q_j = \sum_{k=1}^{p} \lambda_k q_{kj}$ and $W_j = \sum_{i \in V} w_{ij}$
    Change from 1 to 0 the $x_j$ corresponding to the $\min_j Q_j/W_j$
End While
(* Improvement *)
    Rank all the variables $x_j = 0$ by decreasing order of $Q_j = \sum_{k=1}^{p} \lambda_k q_{kj}$.
    Consider this ranking to change variables, from 0 to 1, which keep satisfied all the constraints.
Output: $\mathbf{x}$—repaired and improved solution
```

The steps to *repair* an infeasible solution intend to remove items that most benefit the fulfillment of violated constraints (highest W_j's) and least decrease the values of the objective functions, measured by a weighted-sum of objective values (lowest Q_j's). The steps to *improve* a feasible solution intend to add items that most benefit the same weighted-sum of the objective functions, provided that the solution remains feasible. Note that the weight vector $\boldsymbol{\lambda}$ is now used to produce a weighted-sum instead of a weighted Tchebycheff distance. This change was considered in order to create a faster routine, because the determination of the item to remove that least deteriorates the value of the Tchebycheff scalarizing function requires more computations. Fortunately, experimental computations have shown that this adaptation also produces good results. Finally, note that any new solution generated by MOTGA is attempted to be improved even if it is originally feasible, because any new solution is inputted into the routine *repair_improve*.

4. Comparative experiment

MOTGA was implemented using the Delphi developer for Windows. It has been tested on several randomly generated MOMKP instances, and also on nine MOMKP instances first used in [9] (whose data are available on the site [24]) and further considered in other studies.

In this section we present computational results for those 9 instances, which have 2, 3, 4 objectives and 250, 500, 750 variables. They will be denoted by 2-250, 2-500, ..., 4-750, where the first index represents the number of knapsacks (i.e. the number of objective functions and constraints, as $p = m$ in all instances), and the second index represents the number of items (i.e. the number of variables, n).

We compare the results of MOTGA with the following evolutionary algorithms: SPEA [9], SPEA2 [19] (whose results are available on the site [24]) and MOGLS [15] (whose results are available on the site [25]). Although Zitzler et al. [19] have claimed that SPEA2 outperforms SPEA, we have also included SPEA in our comparison because the experimentation with SPEA2 on MOMKP presented in [19] is restricted to the three instances with 750 items (2-750, 3-750 and 4-750).

4.1. Performance measures

We have adopted three types of performance measures to evaluate the approximations sets produced by the different evolutionary algorithms, and thus give a numerical comparison of them.

(i) *Strict coverage of two sets*, which computes the fraction of solutions of one set dominated by solutions of another set. Let A and B be two sets of solutions. The strict coverage measure is defined as

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

The value $C(A, B) = 1$ means that all points of B are dominated by points of A. The value $C(A, B) = 0$ means that none point of B is dominated by any point of A. The coverage measure has been widely used in previous experiments (e.g. [9,15,16]), considering $\mathbf{z}' \succeq \mathbf{z}''$ instead of $\mathbf{z}' \succ \mathbf{z}''$. In this study we disaggregate the two measures, computing the strict coverage and the number of equal points.

The measure C can be used to show that the outcomes of one algorithm dominate the outcomes of another algorithm, but it does not tell how much better this is. Moreover, a good coverage value does not mean a good approximation of the whole Pareto front because C does not give any indication about diversity and number of solutions.

Thus, two other performance measures are considered. One of them (ii) estimated how "far" the solutions of one approximation set are from the true nondominated solutions. This measure is restricted to cases for which the Pareto front is known. The other one (iii) is obtained through an aggregation of weighted distances between the ideal solution and the members of an approximation set.

(ii) *Generational distance* [26] is a value representing how "far" an approximation set A is from the Pareto front, P^* (assuming that it is known):

$$GD(A) = \frac{\sqrt{\sum_{i=1}^{|A|} d_i^2}}{|A|},$$

where d_i is the (Euclidean) distance between the point $\mathbf{z}_i \in A$ and the nearest member of P^*.

This measure will be considered only for the instances 2-250 and 2-500 for which P^* is known.

(iii) *Average value of the weighted Tchebycheff scalarizing function on an approximation set A over $\Psi \subset \Lambda$, a set of systematically generated uniformly dispersed weight vectors* [8,15,16,18]:

$$R(A) = \frac{\sum_{\lambda \in \Psi} \min_{\mathbf{z} \in A} \|\mathbf{z}^* - \mathbf{z}\|_{\lambda}}{|\Psi|},$$

where $\min_{\mathbf{z} \in A} \|\mathbf{z}^* - \mathbf{z}\|_{\lambda}$ is the Tchebycheff distance, weighted by $\boldsymbol{\lambda}$, between $\mathbf{z}^*$ and the nearest member of A.

The definition and the number of weight vectors in Ψ depend on a parameter K, such that Ψ contains $\binom{K+p-1}{p-1}$ vectors:

$$\Psi = \left\{ \boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p) \left| \sum_{j=1}^{p} \lambda_j = 1 \text{ and } \lambda_j \in \left\{0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}, 1\right\} \right. \right\}.$$

In our study, we set the parameter K as in [15]: $K = 200$, 50 and 25, respectively for $p = 2$, 3 and 4. As before, we use $\bar{\mathbf{z}}^*$ (the ideal solution for the linear relaxation of the problem) in place of $\mathbf{z}^*$.

Table 1
Average number of (potentially) nondominated solutions

	2-250	2-500	2-750	3-250	3-500	3-750	4-250	4-500	4-750
MOTGA	98.7	165.8	249.9	498.5	1067.0	1406.6	1043.0	2131.0	3094.2
MOGLS	170.7	250.5	259.1	1439.4	2385.5	3031.8	3885.1	5643.4	7190.7
SPEA	55.9	33.9	34.5	389.3	389.7	312.2	960.0	1014.4	1075.7
P^*	567	1427	—	—	—	—	—	—	—

Besides $R(A)$, which gives an average value, we have also considered

$$R^-(A) = \min_{\lambda \in \Psi} \left\{ \min_{\mathbf{z} \in A} \|\mathbf{z}^* - \mathbf{z}\|_\lambda \right\}$$

and

$$R^+(A) = \max_{\lambda \in \Psi} \left\{ \min_{\mathbf{z} \in A} \|\mathbf{z}^* - \mathbf{z}\|_\lambda \right\},$$

which give the minimum and the maximum distance to the ideal solution over Ψ, respectively.

4.2. Methodology and parameter setting

To assess the overall performance of MOTGA and compare it with the other algorithms, MOTGA was executed 20 times on each test problem. The following parameter values were considered: $probC = 0.8$, $probM = 0.1$; N, the size of the internal population, was set to 20, 30 and 40, respectively for problems with 250, 500 and 750 items; G, the number of iterations in each stage, was set to 20, which yields a total of $20 \times (2p + 1)$ generations during a run of the algorithm.

For SPEA and SPEA2, the first 20 outcome files for each instance were taken from [24]. According to [9], 500 generations were simulated per optimization run of SPEA, and the probabilities of crossover and mutation were fixed to 0.8 and 0.01, respectively. The population size was chosen to be dependent on the problem size, which was set between 150 (for the problem 2-250) and 350 (for the problem 4-750). In case of SPEA2 [19], for which the experimentation was restricted to the instances with 750 items, the population size was set to 250 for $p = 2$, 300 for $p = 3$, and 400 for $p = 4$; the mutation probability of 0.06 was considered for flipping each bit.

In case of MOGLS, the 20 outcome files of MOGLS50 for each instance were taken from [25]. MOGLS50 is one of two experiments of MOGLS, the other being MOGLS20, and they just differ in the number of iterations, i.e., $50 \times S$ and $20 \times S$, respectively; S denotes the number of initial solutions and it was set in the same way as the population size of SPEA [9]. Therefore, we have considered MOGLS50 for our comparison study, which we will just refer to MOGLS in the rest of the text.

4.3. Results and discussion

The average number of potentially nondominated solutions obtained by each algorithm for each instance, in the 20 runs experiment, is given in Table 1. In cases for which the Pareto front (P^*) is known (instances 2-250 and 2-500) the respective number of nondominated solutions is also presented in the last row of Table 1. We omit in this table the average number of potentially nondominated solutions obtained using the SPEA2 [19], because this method sets a priori the size of the archive (nondominated population), so the number of solutions is constant over time in all runs (250, 300 and 350, respectively for the instances 2-750, 3-750 and 4-750). We observe that MOGLS is the algorithm that presents the largest average size of the approximation sets for all instances.

Table 2 shows the average running times of MOTGA and MOGLS. The times of MOGLS are those presented in [15] (for MOGLS50), which were recorded on a computer Pentium 350 MHz. In case of MOTGA, we present the average times on a Pentium IV 3.2 GHz (256MB of RAM) and also on a Pentium II 350 MHz (64 MB of RAM), in order to

Table 2
Average running times (s)

Computer	Algorithm	Problem								
		2-250	2-500	2-750	3-250	3-500	3-750	4-250	4-500	4-750
Pentium IV, 3.2 GHz	MOTGA	1.5	7.2	19.5	2.7	12.8	33.4	4.2	18.2	51.9
Pentium II, 350 MHz	MOTGA	9.1	34.5	92.5	14.3	62.1	124.8	21.5	88.9	244.8
	MOGLS	25.9	65.1	134.9	62.5	136.6	251.9	140.6	253.7	428.2

Table 3
Average values of the C measure between MOTGA and MOGLS

	2-250	2-500	2-750	3-250	3-500	3-750	4-250	4-500	4-750
C (MOTGA, MOGLS)	0.843	0.973	0.979	0.504	0.818	0.878	0.280	0.430	0.554
C (MOGLS, MOTGA)	0.045	0.000	0.000	0.085	0.004	0.002	0.046	0.013	0.005

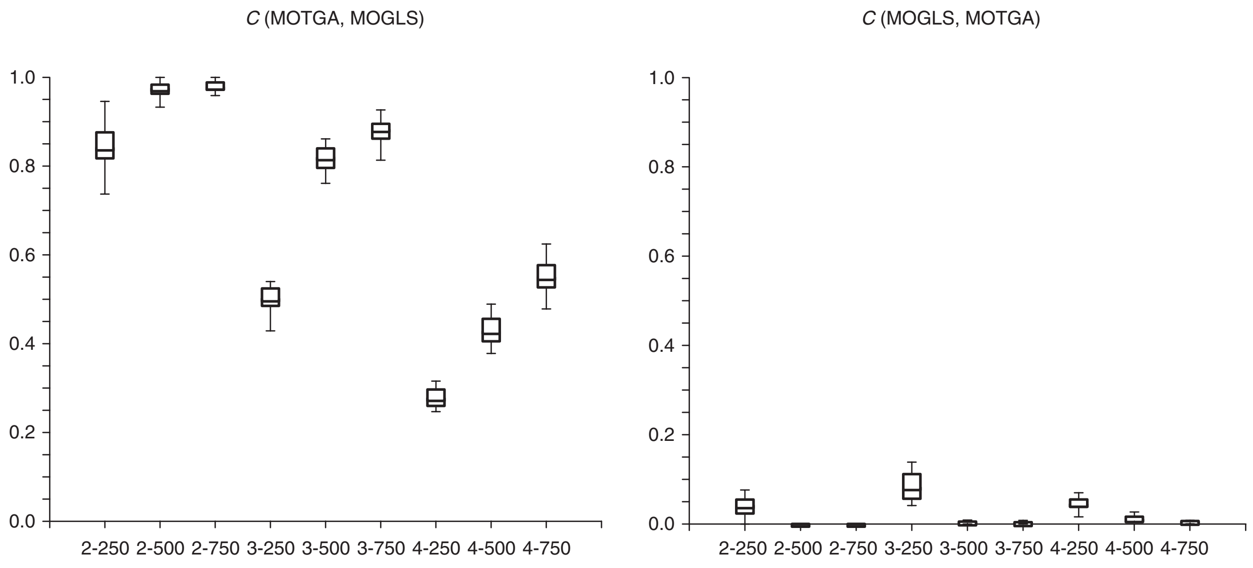

Fig. 2. Box-plots representing the distribution of the coverage (C) values between the outcomes of MOTGA and the outcomes of MOGLS.

enable a more accurate comparison with MOGLS. It can be observed in Table 2 that the average times of MOTGA are lower than the corresponding ones of MOGLS, in all instances.

MOTGA was assessed in pairs with the other algorithms using the *strict coverage* (C) measure. For an ordered pair of algorithms (A, B) there is a sample of $20C$ values for each instance, according to the 20 runs experiment. In these comparisons, all the solutions of SPEA are dominated by solutions of MOTGA and none solution of MOTGA is dominated by any solution of SPEA. This is still true when we compare MOTGA with SPEA2. In what concerns the comparisons with MOGLS, we observe that the average number of solutions of MOGLS dominated by solutions of MOTGA varies from 28% to 97.9%. On the other side, the average number of solutions of MOTGA dominated by solutions of MOGLS lies between 0% and 8.5%. It can also be mentioned that there are no equal solutions in all these comparisons. The average C values obtained from the comparisons between MOTGA and MOGLS are presented in Table 3. Fig. 2 shows the box-plots representing the distribution of the 20 values of C(MOTGA, MOGLS) and C(MOGLS, MOTGA) for each problem.

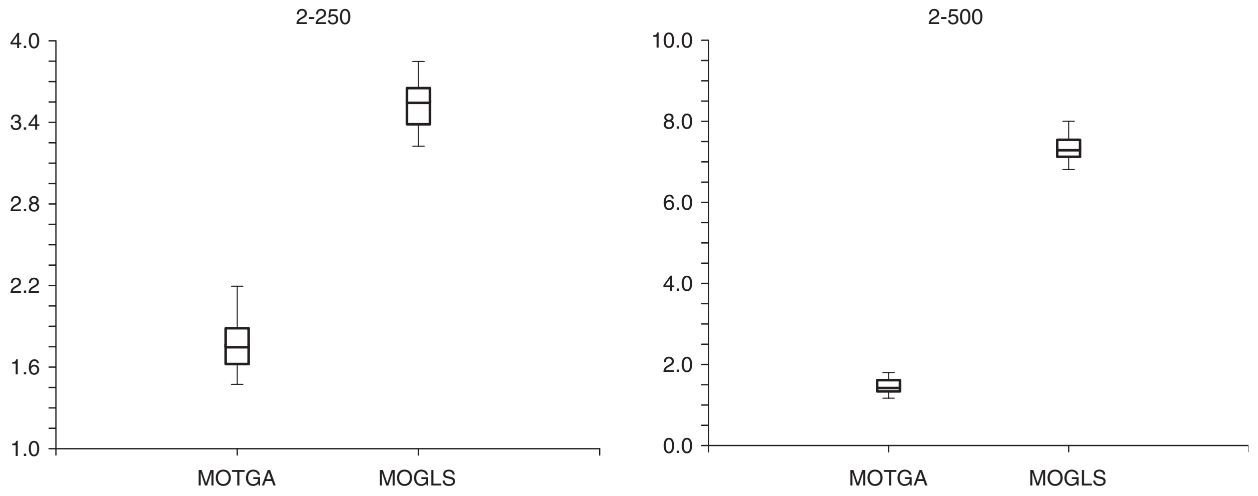

Fig. 3. Box-plots representing the distribution of the generational distance (*GD*) values.

Table 4
R, R^+, R^- average values

	2-250			2-500			2-750			3-250			3-500		
	R	R^+	R^-	R	R^+	R^-	R	R^+	R^-	R	R^+	R^-	R	R^+	R^-
MOTGA	258	369	31	446	640	29	765	1101	27	334	444	81	627	823	174
MOGLS	264	370	20	472	659	34	816	1138	28	339	449	40	662	858	91
SPEA2	–	–	–	–	–	–	1047	1222	862	–	–	–	–	–	–
P^*	250	357	4	439	626	2	–	–	–	–	–	–	–	–	–

	3-750			4-250			4-500			4-750		
	R	R^+	R^-	R	R^+	R^-	R	R^+	R^-	R	R^+	R^-
MOTGA	902	1217	226	346	494	90	681	992	216	1039	1411	394
MOGLS	961	1280	18	346	484	64	704	1026	133	1091	1485	77
SPEA2	1336	2023	1036	–	–	–	–	–	–	1742	3086	1174

In the following assessments, we will omit the performance values of SPEA as they were worse than the corresponding ones of the other algorithms in all the tested cases.

The average *generational distance* (*GD*) values for the instances 2-250 and 2-500 (the only instances for which the Pareto front is known) are the following: MOTGA: 1.80 and 1.47; MOGLS: 3.53 and 7.37, respectively. Fig. 3 shows the box-plots representing the distribution of the *GD* values obtained in the 20 runs for these problems.

Table 4 shows the average R, R^+ and R^- values (rounded to whole numbers) of MOGLS and MOTGA, for all instances, and SPEA2 for the instances with 750 items. The corresponding measures for the Pareto front (P^*) on the first two instances are also presented. Note that the R^- values of P^* are larger than 0 just because $\bar{\mathbf{z}}^*$ was used instead of $\mathbf{z}^*$, and the components of $\bar{\mathbf{z}}^*$ are upper bounds for the true optimal objective function values (components of $\mathbf{z}^*$). It can be observed in Table 4 that, in general, the R and R^+ average values of MOTGA are better (lower) than the ones of MOGLS, but none algorithm outperforms the other according to R^-. Fig. 4 shows the box-plots representing the distribution of the R values obtained in the 20 runs experiment on the instances with 750 items.

Figs. 5 and 6 show the potentially nondominated solutions found in the first five runs of each algorithm on the instances 2-500 and 2-750, respectively. The outcomes of the first 5 runs were unified and then the nondominated solutions within this union set were picked. These are the points plotted in Figs. 5 and 6. The Pareto front of the instance 2-500 is also represented in Fig. 5.

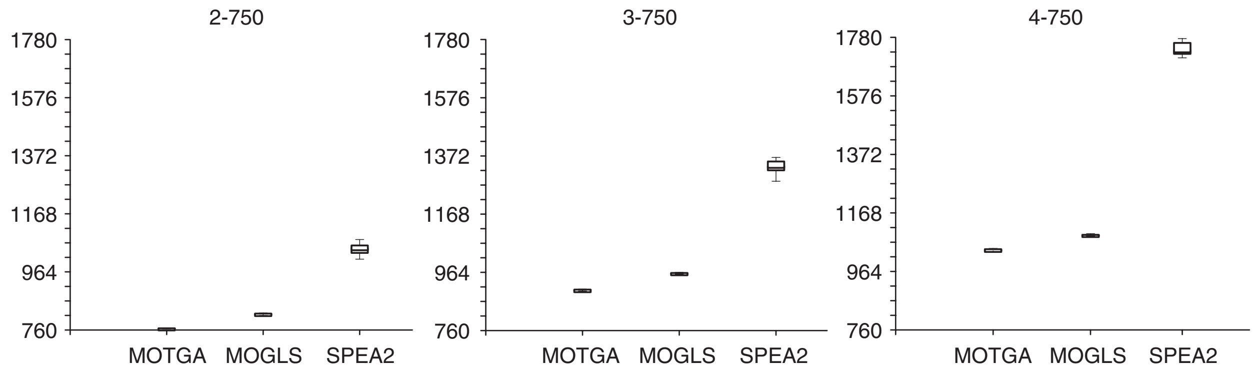

Fig. 4. Box-plots representing the distribution of the R values on the instances with 750 items.

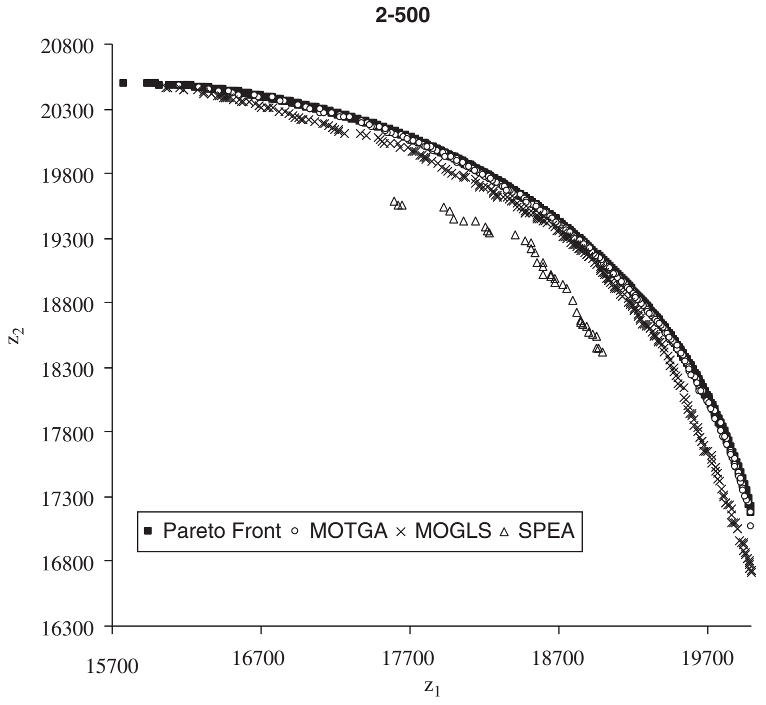

Fig. 5. Potentially nondominated solutions obtained in five runs of each algorithm on the instance 2-500, and the corresponding Pareto front.

5. Conclusions

In this paper we presented a new genetic algorithm for the multiobjective multidimensional knapsack problem, which is based on the use of the Tchebycheff scalarizing function.

The new algorithm, called MOTGA, was compared with the evolutionary algorithms SPEA [9], MOGLS [15] and SPEA2 [19] on a test set of nine multiobjective knapsack instances. In these experiments, MOTGA performed well as it produced high quality solutions, namely when compared with the other algorithms. The number of potentially nondominated solutions obtained for each instance was not very high (namely when compared with MOGLS), but the solutions seem to be well dispersed and show good quality indicators.

In bi-objective problems, the graphical representation of the solutions on the objective function space shows that the generated approximations sets provide, in fact, good approximations of the Pareto front. As the number of objectives

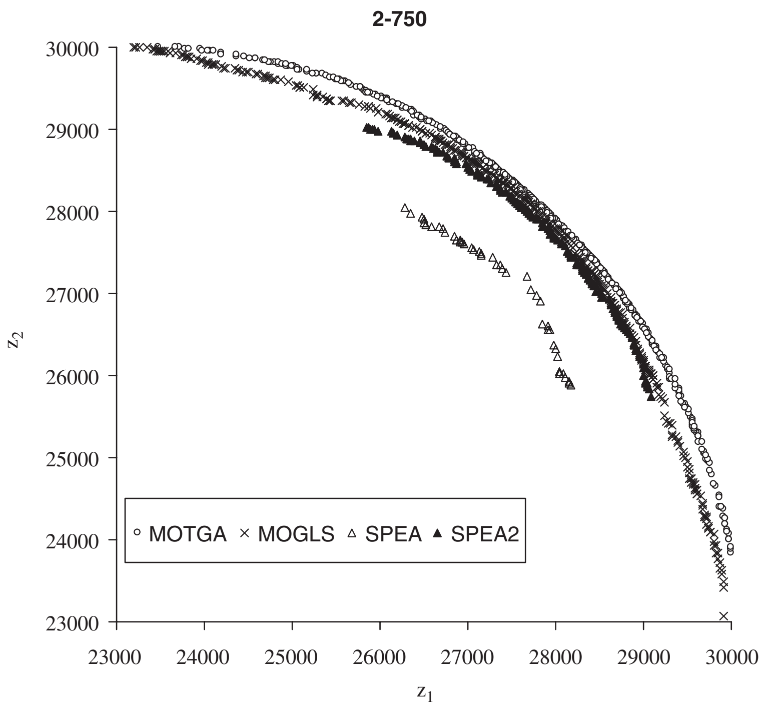

Fig. 6. Potentially nondominated solutions obtained in five runs of each algorithm on the instance 2-750.

increases, the evaluation of an approximation set becomes more difficult. We adopted different types of quality measures: the strict coverage measure (*C*), which indicates the proportion of solutions of one approximation set that are dominated by another set, the generational distance (*GD*), a performance indicator when the Pareto front is known, and the *R*, R^+, R^- measures, which give indications on the relative merit and also on the absolute merit of each outcome set. MOTGA presented good values to these quality measures, as they are, in general, better than in the other algorithms. Despite this fact, we believe that further quality measures should be established in order to provide more accurate evaluations, namely for problems with more than two objective functions. Furthermore, many other experiments must be made to sustain the conclusions on the performance of MOTGA, namely by considering knapsack instances with more constraints, where the reestablishment of the feasibility of solutions could be more difficult to achieve. Also, this algorithm should be extended to other multiobjective combinatorial problems.

Acknowledgments

The authors would like to thank to an anonymous referee for his/her useful comments and suggestions that helped to improve the paper.

This work was supported in part by the Foundation for Science and Technology (FCT) under Grants POSI/SRI/37346/2001 and POCTI/GES/37707/2001.

References

[1] Deb K. Multi-objective optimization using evolutionary algorithms. New York: Wiley; 2001.

[2] Coello Coello CA, Van Veldhuizen DA, Lamont GB. Evolutionary algorithms for solving multi-objective problems. Dordrecht: Kluwer Academic Publishers; 2002.

[3] http://www.lania.mx/~ccoello/EMOO/EMOObib.html

[4] Jones DF, Mirrazavi SK, Tamiz M. Multi-objective meta-heuristics: an overview of the current state-of-art. European Journal of Operational Research 2002;137(1):1–9.

[5] Alves MJ, Clímaco J. An interactive method for 0–1 multiobjective problems using simulated annealing and tabu search. Journal of Heuristics 2000;6(3):385–403.
[6] Phelps SP, Köksalan M. An interactive evolutionary metaheuristic for multiobjective combinatorial optimization. Management Science 2003;49(12):1726–38.
[7] Ulungu EL, Teghem J, Ost Ch. Efficiency of interactive multi-objective simulated annealing through a case study. Journal of the Operational Research Society 1998;49:1044–50.
[8] Jaszkiewicz A. Comparison of local search-based metaheuristics on the multiple-objective knapsack problem. Foundations of Computing and Decision Sciences 2001;26(1):99–120.
[9] Zitzler E, Thiele L. Multiple objective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 1999;3(4):257–71.
[10] Knowles JD, Corne DW. A comparison of diverse approaches to memetic multi-objective combinatorial optimization. In: Proceedings of the 2000 genetic and evolutionary computation conference workshop program. Las Vegas, Nevada, 2000. p. 103–8.
[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] Serafini P. Simulated annealing for multiple objective optimization problems. In: Proceedings of the tenth international conference on multiple criteria decision making, Taipei, vol. 1; 1992. p. 87–96.
[13] Ulungu EL, Teghem J, Fortemps Ph, Tuyttens D. MOSA method: a tool for solving multiobjective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis 1999;8:221–36.
[14] Czyzak P, Jaszkiewicz A. Pareto simulated annealing—a metaheuristic technique for multiple objective combinatorial optimisation. Journal of Multi-Criteria Decision Analysis 1998;7:34–47.
[15] Jaszkiewicz A. On the performance of multiple objective genetic local search on the 0/1 knapsack problem. A comparative experiment. Technical Report RA-002/2000, Institute of Computing Science, Poznan University of Technology, 2000.
[16] Jaszkiewicz A. On the performance of multiple objective genetic local search on the 0/1 knapsack problem—a comparative experiment. IEEE Transactions on Evolutionary Computation 2002;6(4):402–12.
[17] Ishibuchi H, Murata T. Multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man and Cybernetics 1998;28(3):392–403.
[18] Jaszkiewicz A. On the computational efficiency of multiple objective metaheuristics. The knapsack problem case study. European Journal of Operational Research 2004;158:418–33.
[19] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm. Zuerich, TIK-Report Nr. 103, Computer Engineering and Networks Lab (TIK). Swiss Federal Institute of Technology, Zurich, 2001.
[20] Chankong V, Haimes YY. Multiobjective decision making: theory and methodology. New York: North Holland; 1983.
[21] Steuer R. Multiple criteria optimization: theory, computation and application. New York: Wiley; 1986.
[22] Wierzbicki AP. Reference point approaches. In: Gal T, Stewart TJ, Hanne T, editors. Multiple criteria decision making: advances in MCDM models, algorithms, theory and applications. Boston: Kluwer; 1999. p. 1–39 [Chapter 9].
[23] Bowman, Jr VJ. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In: H. Thiriez, S. Zionts editors, Multiple criteria decision making, Lecture Notes in Economics and Mathematical Systems, vol. 130. Berlin: Springer; 1976. p. 76–86.
[24] http://www.tik.ee.ethz.ch/∼zitzler/testdata.html
[25] http://www-idss.cs.put.poznan.pl./∼jaszkiewicz/mokp/
[26] Veldhuizen DV, Lamont GB. Multiobjective evolutionary algorithm test suites. In: Carroll J et al., editors. Proceedings of the 1999 ACM symposium on applied computing. San Antonio, TX: ACM, 1999. p. 351–7.