Experimental analysis of design elements of scalarizing function-based multiobjective evolutionary algorithms

Mansoureh Aghabeig¹ · Andrzej Jaszkiewicz¹

Published online: 28 November 2018
© The Author(s) 2018

Abstract

In this paper, we systematically study the influence of the main design elements of scalarizing function-based multiobjective evolutionary algorithms (MOEAs) on the performance of these algorithms. Such algorithms proved to be very successful in multiple computational experiments and practical applications. Well-known examples of this class of MOEAs are Jaszkiewicz's multiobjective genetic local search and multiobjective evolutionary algorithm based on decomposition (MOEA/D). The two algorithms share the same common structure and differ in two aspects, i.e., the selection of parents for recombination and the selection of weight vectors of scalarizing functions. Using three different multiobjective combinatorial optimization problems, i.e., the multiobjective symmetric traveling salesperson problem, the traveling salesperson problem with profits, and the multiobjective set covering problem, we show that the design element with the highest influence on the performance is the choice of a mechanism for parents selection, while the selection of weight vectors, either random or evenly distributed, has practically negligible influence if the number of evenly distributed weight vectors is sufficiently large.

Keywords Metaheuristics · Multiobjective evolutionary algorithms · Combinatorial optimization · Traveling salesperson problem · Set covering problem

1 Introduction

In many areas, an optimal decision should take into consideration two or more conflicting objectives. A problem with multiple objectives is called a *multiobjective combinatorial optimization problem* (MOOP), if it has two characteristics. First, the decision variables are discrete, and second, the set of feasible solutions is finite. Combinatorial optimization finds applications in many real-world problems, e.g., scheduling, time tabling, production, facilities design, and routing (Yu 2013).

In multiobjective optimization, there is usually no solution for which all objectives have optimal values, so the preferences of the decision maker (DM) need to be taken

into account in order to select the best compromise solution. It is a generally accepted assumption that the DM's preferences are compatible with the dominance relation. Under this assumption, the most preferred solution belongs to the Pareto set. Thus, the goal of most multiobjective optimization algorithms is finding the Pareto set, or a good approximation of it, for further exploration by the DM.

Evolutionary algorithms are a promising option for solving multiobjective problems (Deb 2001). They process a population of candidate solutions in each iteration, so they are able to search for multiple (approximately) Pareto solutions concurrently in a single run.

In evolutionary algorithms, in each intermediate iteration, a selection process is performed in which good members of the population have a higher probability of survival (and generating offspring) and worse members have a higher probability of elimination. Thus, a mechanism for evaluation of intermediate solutions is necessary. In the single-objective case, intermediate solutions can naturally be evaluated with the value of the single-objective function. In the multiobjective case, there is no such obvious evaluation mechanism, so different mechanisms are used in different methods.

Communicated by V. Loia.

✉ Andrzej Jaszkiewicz
andrzej.jaszkiewicz@put.poznan.pl
Mansoureh Aghabeig
mansoureh.aghabeig@yahoo.com

¹ Institute of Computing Science, Faculty of Computing,
Poznan University of Technology, ul. Piotrowo 2,
60-965 Poznan, Poland

One frequently used type of evaluation mechanism is the Pareto dominance-based evaluation. A typical example is Pareto ranking used, e.g., in the NSGA2 algorithm (Deb et al. 2002). Pareto dominance-based evaluation is also used in PAES (Knowles and Corne 1999) and SPEA (Zitzler and Thiele 1999). Although Pareto dominance-based mechanisms have the advantage of not requiring the transformation of a MOOP into a single-objective problem, they may suffer from other weaknesses, such as losing selection pressure with the increasing number of objectives, needing an additional mechanism for preserving diversity, and difficult hybridization with local search (Neri et al. 2012).

Another type of evaluation mechanism is scalarizing function-based evaluation. In mechanisms of this type, a multiobjective optimization problem is transformed into a family of parametric single-objective optimization problems. In each problem, a nonnegative weight vector defines a single scalarizing function. Typically used scalarizing functions are described in the next section.

In the extreme case, two or more objectives may be combined with a single scalarizing function and the problem may be further treated as a single-objective problem, see, e.g., Abualigah et al. (2018). However, in the case of scalarizing function-based MOEAs, multiple scalarizing functions with various weight vectors are used to generate an approximation of the whole or a part of the Pareto front.

Two well-known examples of multiobjective evolutionary algorithms based on scalarizing functions are JMOGLS Jaszkievicz (2002a) and MOEA/D Zhang and Li (2007). These methods proved to be very successful in multiple computational experiments (Gong et al. 2012; Ishibuchi et al. 2015; Ke et al. 2013; Zhang et al. 2010; Li et al. 2014; Ding and Wang 2013; Liu et al. 2014; Jaszkievicz 2002b, 2003; Mei et al. 2011; Ishibuchi et al. 2011; Sindhya et al. 2011; Kafafy et al. 2012) and practical applications (Konstantinidis and Yang 2011; Sengupta et al. 2012, 2013; Carvalho et al. 2012; Trivedi et al. 2015). However, none of these computational experiments had the same goal as our study, which is understanding of the importance of the main design elements differentiating JMOGLS and MOEA/D.

Although the ways JMOGLS and MOEA/D presented in the original papers were very different, the two methods have a similar structure. In fact, they differ only in two main elements: the selection of the weight vectors defining scalarizing functions, and the selection of parents for recombination.

The main contribution of this paper is a clear understanding which of the two above-mentioned design elements has the main influence on the performance of JMOGLS and MOEA/D. In order to answer this question, we propose a new method that is an intermediate method between JMOGLS and MOEA/D and differs from each of them in just one of the design elements. In the computational experiment, we use three different multiobjective combinatorial optimiza-

tion problems, i.e., the multiobjective symmetric traveling salesperson problem, the traveling salesperson problem with profits, and the multiobjective set covering problem. The use of three different problems allows us to make more general conclusions. Wherever possible, we use instances from well-known libraries, such as multiobjective symmetric traveling salesperson problem instances from Lust's library¹ (Lust and Teghem 2010) or multiobjective set covering problem instances from another library of the same author² (Lust and Tuytens 2014). Some other instances have been generated for the purpose of this experiment, in a clearly described way. The most important result of the computational experiment is that the design element with the highest influence on the performance of the scalarizing function-based MOEAs is the choice of a mechanism for parents selection, while the selection of weight vectors, either random or evenly distributed, has practically negligible influence.

The rest of this paper is organized as follows. In the next section, some basic definitions are given. In Sect. 3, a short description of JMOGLS and MOEA/D is given. An intermediate method between JMOGLS and MOEA/D, called Evenly distributed MOGLS (EMOGLS), is introduced in Sect. 4. The computational experiments and discussion of the obtained results are presented in Sects. 5 and 6, respectively. The paper ends with conclusions and potential directions for future research.

2 Basic definitions

The multiobjective optimization problem (MOOP) is an optimization problem which involves multiple objective functions and in mathematical terms can be formulated as:

$$\begin{aligned} \text{"minimize"} [f_1(x) = z_1, \dots, f_J(x) = z_J] \\ \text{s.t. } x \in D \end{aligned}$$

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

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

Point z^1 dominates z^2 , $z^1 \succ z^2$, if $\forall_j z_j^1 \leq z_j^2$ and $z_j^1 < z_j^2$ for at least one j . A solution is a *Pareto solution*, if there does not exist another feasible solution which dominates it. The image of a Pareto solution in the objective space is called a *non-dominated point*. The set of all Pareto solutions is called the *Pareto set*, and the image of the Pareto set in the objective space is called the *non-dominated frontier* or

¹ <http://www.desir.lip6.fr/~lustt/Research.html#MOTSP>.

² <http://www.desir.lip6.fr/~lustt/Research.html#MOSCP>.

the *Pareto front*. Two solutions are *mutually non-dominated* if neither of them dominates the other and their images in the objective space are different. In this paper, a set of mutually non-dominated solutions generated by a multiobjective evolutionary algorithm is called a *Pareto archive*.

Weighted linear scalarizing functions are defined in the following way:

$$s_1(z, \Lambda) = \sum_{j=1}^J \lambda_j z_j$$

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

Each weighted linear scalarizing function has at least one global optimum belonging to the Pareto set (Steuer 1985).

Weighted Chebyshev scalarizing functions are defined in the following way:

$$s_\infty(z, z^*, \Lambda) = -\max_j (\lambda_j (z_j^* - z_j))$$

where z^* is a reference point and $\Lambda = [\lambda_1, \dots, \lambda_J] \forall \lambda_j \geq 0$ is a weight vector.

Each weighted Chebyshev scalarizing function has at least one global optimum belonging to the Pareto set. For each Pareto solution x , there exists a weighted Chebyshev scalarizing function s_∞ such that x is a global optimum of s_∞ (Steuer 1985).

The above properties suggest an advantage of weighted Chebyshev scalarizing functions, since every Pareto solution may be obtained by optimizing this type of function. However, this property holds only if an exact optimum solution of the function can be obtained. In practice, when heuristic methods are used, linear scalarizing functions often perform better (Zhang and Li 2007; Jaszkievicz 2002a).

The two classes of functions can also be combined, producing *mixed scalarizing functions*, defined in the following way:

$$s_m(z, z^*, \Lambda) = w_1 s_1(z, \Lambda) + w_\infty s_\infty(z, z^*, \Lambda)$$

where w_1 defines the weight of the linear scalarizing function and w_∞ defines the weight of the Chebyshev scalarizing function. The sum of these two weights should equal one.

3 JMOGLS and MOEA/D algorithms

The main idea of scalarizing function-based multiobjective algorithms is as follows: if we optimized all weighted Chebyshev scalarizing functions defined by all possible weight vectors, we would obtain the true Pareto set. Unfortunately, implementing this idea in practice is impossible, since the set of all weight vectors is infinite, and, in many cases, there

exists no exact method for finding the optimum solution of a scalarizing function within a realistic time frame. However, we can still approximate the Pareto set by heuristic optimization of a set of various scalarizing functions defined by a set of well-distributed weight vectors.

From another point of view, JMOGLS and MOEA/D are based on the single-objective genetic local search (sGLS) algorithm (Kolen and Pesch 1994; Jaszkievicz and Kominek 2000). In each iteration of sGLS, two solutions (parents) are chosen for recombination from a population of solutions being relatively good on the objective function. The offspring is generated by a recombination of the parents and then improved by a local search.

A single iteration in both JMOGLS and MOEA/D is almost the same as a single iteration in sGLS, i.e., in each iteration the algorithms select two solutions which are relatively good on the current scalarizing function, and the offspring is then improved by a local search guided by the same function. However, for each iteration, a different weight vector, and thus a different scalarizing function, is selected. Furthermore, both JMOGLS and MOEA/D use special mechanisms for parents selection. These mechanisms are a necessity, as the populations used in multiobjective algorithms are relatively large and contain solutions that are dispersed over various regions of the objective space, only some of them being good on the current scalarizing function (while in single-objective algorithms all solutions in the population are usually relatively good on the single-objective function). In short, JMOGLS and MOEA/D use two specific mechanisms: one for choosing weight vectors, and another for choosing two parents which are relatively good on the current scalarizing function. The two mechanisms differ in each method.

JMOGLS was proposed by Jaszkievicz (2002a) and further developed in Jaszkievicz (2004). It is based on the algorithm proposed by Ishibuchi and Murata (1998). Both methods choose weight vectors defining the scalarizing functions at random, but JMOGLS uses an aggressive tournament selection (instead of roulette wheel selection, used by Ishibuchi and Murata) to select very good solutions for recombination. The selection is aggressive in the sense that a relatively large number of solutions take part in the tournament, and only the best and second best solutions (according to the current scalarizing function) are selected as parents.

The original version (Jaszkievicz 2002a) used a so-called temporary population, selected in each iteration, to achieve an aggressive selection. The tournament selection was proposed in the updated version of JMOGLS (Jaszkievicz 2004). Since the tournament selection is less time-consuming than the original mechanism, it is what we chose to use in this paper.

MOEA/D, proposed by Zhang and Li (2007), generates a finite set of evenly distributed weight vectors defining a set of scalarizing functions. Zhang et al. interpret it as a decom-

position of the MOOP into a number of single-objective subproblems corresponding to particular weight vectors, giving rise to its name.

Note that the two methods do not simply perform an independent optimization of a number of scalarizing functions. In each iteration, the parents are selected from a common population, so a parent could have been obtained with the use of another scalarizing function defined by a weight vector different (but usually similar) to the one currently used. In other words, solutions obtained during the optimization of a given scalarizing function help optimize other, similar scalarizing functions.

3.1 General structure of JMOGLS and MOEA/D

As mentioned above, the general structure of both JMOGLS and MOEA/D is the same. It is described in Algorithm 1.

In each iteration of the initial phase, a weight vector is chosen and used as the basis for defining a scalarizing function. A new feasible solution is then generated and improved by a local search based on the current scalarizing function. Finally, the Pareto archive is updated with the new solution.

A weight vector is also chosen in each iteration of the main phase, after which two solutions that are relatively good on the scalarizing function defined by the chosen weight vector are selected as parents. A new solution (offspring) is generated by a recombination of the parents, and afterward improved by a local search. At the end of each iteration, the Pareto archive is updated with the new offspring.

Algorithm 1 General Structure

Parameter $\downarrow$: Number of Initial Solutions K
 Parameter $\downarrow$: Stopping criterion
 Parameter $\uparrow$: Pareto archive $\hat{\mathcal{A}}$

--| Initial Phase

$\hat{\mathcal{A}} := \emptyset$

for ($k = 0; k < K; k++$) do

 GetWeight($\Lambda \uparrow$)

 Construct a new feasible solution X

 LocalSearch($s(z, \Lambda) \downarrow, X \uparrow$)

 Update($X \downarrow, \hat{\mathcal{A}} \uparrow$)

end for

--| Main Phase

repeat

 GetWeight($\Lambda \uparrow$)

 GetParents($s(z, \Lambda) \downarrow, X_1 \uparrow, X_2 \uparrow$)

 Recombine($X_1 \downarrow, X_2 \downarrow, X_3 \uparrow$)

 LocalSearch($s(z, \Lambda) \downarrow, X_3 \uparrow$)

 Update($X_3 \downarrow, \hat{\mathcal{A}} \uparrow$)

until the stopping criterion is met

Algorithm 2 GetWeight (JMOGLS)

Parameter $\uparrow$: Weight vector Λ

Draw at random a normalized weight vector Λ

Algorithm 3 GetWeight (MOEA/D, EMOGLS)

Parameter $\uparrow$: Weight vector Λ

if (it is the first time) then

 Generate N evenly distributed weight vectors: $\Psi_s = \{\Lambda^1, \dots, \Lambda^N\}$

end if

Take the next weight vector from Ψ_s . If there is no next weight vector, use the first one

Algorithm 4 GetParents (JMOGLS, EMOGLS)

Parameter $\downarrow$: Scalarizing function $s(z, \Lambda)$

Parameter $\downarrow$: Pareto archive $\hat{\mathcal{A}}$

Parameter $\downarrow$: Size of tournament T

Parameter $\uparrow$: Parents X_1, X_2

Select at random T solutions from $\hat{\mathcal{A}}$ for tournament

Choose the first X_1 and second X_2 best solutions in the tournament as parents

Algorithm 5 GetParents (MOEA/D)

Parameter $\downarrow$: Number of neighborhood weight vectors N

Parameter $\downarrow$: Evenly distributed weight vectors Ψ_s

Parameter $\uparrow$: Parents X_1, X_2

if (it is the first time) then

 for each weight vector i in Ψ_s , let $B(i) = \{N \text{ closest weight vectors to } i\}$

end if

Randomly select two indexes k, l from $B(i)$ and use related solutions as parents

3.2 Selection of weights

As mentioned above, JMOGLS and MOEA/D differ in the way they choose the weight vector in each iteration. JMOGLS draws a weight vector at random in each iteration using the algorithm proposed in Jaskiewicz (2002a), whereas in MOEA/D a finite set of evenly distributed weight vectors is generated when the algorithm starts. Then, in each iteration, MOEA/D chooses the next weight vector from this set.

3.3 Selection of parents

JMOGLS and MOEA/D also differ in the way they select parents for recombination. Note that by selection of parents we mean the whole process influencing the final choice of

parents. This process includes the choice of the set (population) from which the parents are selected, the mechanism for updating this population and the mechanism for final selection of parents from this population.

In JMOGLS, two parents are chosen by tournament selection from the whole Pareto archive $\hat{\mathcal{A}}$. In each iteration, a sample of size T is drawn at random from the Pareto archive. Then, two solutions (parents) which are the best on the current scalarizing function are selected from the sample. The size T of the tournament sample is determined in a way which ensures that the two selected solutions have a specified expected rank (Jaszkiewicz 2004), by which we mean the position of the solution in an order induced by the current scalarizing function s in the whole Pareto archive, with the best solution for s having a rank of 1. As shown in Jaszkiewicz (2004), the expected rank Er of the best solution in the sample of T randomly selected solutions is well approximated by:

$$Er \approx \frac{3|\hat{\mathcal{A}}|}{2T}$$

So, the larger the size of the tournament sample compared to the size of the Pareto archive, the better the solutions selected in the tournament.

In MOEA/D, a single solution is associated with each of the evenly distributed weight vectors. Furthermore, a neighborhood relation among evenly distributed weight vectors (and thus corresponding subproblems) is defined based on their Euclidean distance in the weight space. More precisely, a neighborhood of a given weight vector is composed of a number of its closest weight vectors. In the original version of MOEA/D, the two parents are selected from a subset of solutions corresponding to the neighbor weight vectors. In this paper, we use a newer version of MOEA/D (Zhang et al. 2009), in which parents are selected from either the set of solutions corresponding to all subproblems or the subset of solutions corresponding to the neighbor subproblems with some probability. Furthermore, the new version of MOEA/D updates only a limited number of solutions in each iteration, while in the original version all solutions of the neighborhood subproblems are updated.

Zhang and Li (2007) argue that this way of selecting parents is faster than the mechanism used in JMOGLS. Though this is indeed true, in JMOGLS and MOEA/D the vast majority of CPU time is spent on local search, while the time needed to select parents is practically negligible.

Note that the expected rank in JMOGLS plays a role similar to the size of the neighborhood in MOEA/D. The lower the expected rank and the lower the size of the neighborhood, the better, on average, the solutions for the current scalarizing function, selected for recombination.

4 Evenly distributed MOGLS

As stated above, our goal is to experimentally assess which of the two elements differentiating JMOGLS and MOEA/D has a greater influence on performance and which versions of these elements yield better results. However, if we observed differences in the performance of the two methods, we would not know which of the two different elements is the main source of these differences. Therefore, we propose an intermediate method, called Evenly distributed MOGLS (EMOGLS), which is different in just one element from both JMOGLS and MOEA/D. EMOGLS selects weight vectors from a set of evenly distributed weight vectors similarly to MOEA/D, but it chooses the solutions for recombination in the same way as JMOGLS.

We could also consider another intermediate combination of the design elements, i.e., a method that would select weight vectors like JMOGLS and select solutions for recombination like MOEA/D. We do not see, however, any straightforward way to implement such a combination, since in MOEA/D the selection of solutions for recombination is strongly linked with the existence of evenly distributed weight vectors and association of solutions with these weight vectors.

Evenly distributed weight vectors were also used in some other MOGLS algorithms (Murata et al. 2001; Ishibuchi et al. 2009), but these methods differ from JMOGLS and MOEA/D in other aspects.

5 Computational experiment

In order to experimentally assess the influence of different elements in JMOGLS and MOEA/D, we compare the algorithms on instances of three different multiobjective combinatorial problems, i.e., multiobjective symmetric traveling salesperson problem (TSP), traveling salesperson problem with profits, and multiobjective set covering problem. To avoid the influence of implementation details, all methods were implemented in Java, sharing as much of the code as possible.

5.1 Multiobjective symmetric TSP

Given N cities (nodes) and the traveling costs (distances) $c_{i,l}^j$ ($i \neq l$) between each pair of distinct cities, the multiobjective traveling salesperson problem consists in finding a circular path visiting each city exactly once. In other words, the goal is to find a permutation p of the cities that minimizes the following objectives ($j = 1, \dots, J$):

$$\text{“minimize” } z_j(p) = \sum_{i=1}^{N-1} c_{p(i), p(i+1)}^j + c_{p(N), p(1)}^j$$

In this paper, we use the symmetric version of the multiobjective traveling salesperson problem (MSTSP), where: $c_{i,l}^j = c_{l,i}^j$ for $1 \leq i, l \leq N$.

5.2 Multiobjective TSP with profit

An extension of TSP is TSP with profit (TSPWP) (Feillet et al. 2005). It is formulated as follows: given a set of N cities and profits associated with each city find a sub-tour of the cities which minimizes the tour length and maximizes the collected profit.

TSPWP is multiobjective in nature (Jozefowicz et al. 2008). However, it is usually thought of as a single-objective problem and solved by an aggregation of the two objectives (Feillet et al. 2005). TSPWP is a problem with heterogeneous objectives, i.e., the objectives are defined by functions of different mathematical forms.

5.3 Multiobjective set covering problem (MOSCP)

MOSCP consists in covering the rows of an L -row, I -column, zero-one matrix in which elements are denoted by a_{li} , $l = 1, \dots, L$, and $i = 1, \dots, I$, with a subset of the columns minimizing J cost-type objectives (Jaskiewicz 2003). Define $x_i = 1$ if column i (with cost $c_i^j > 0$, $j = 1, \dots, J$) is selected in the solution, and $x_i = 0$ otherwise; MOSCP is formulated as follows:

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

5.4 Quality indicators

In this paper, we use the following quality measures:

- R measure (Hansen and Jaskiewicz 1998; Jaskiewicz 2002a) evaluates a Pareto archive $\hat{A}$ with the average value of weighted Chebyshev scalarizing functions over a set of normalized weight vectors. It is calculated as follows:

$$R(\hat{A}) = \frac{\sum_{\Lambda \in \Psi_s} \min_{z \in \hat{A}} s_\infty(z, z^*, \Lambda)}{|\Psi_s|},$$

where Ψ_s is the set of evenly distributed weight vectors generated with the procedure described in Jaskiewicz (2002a).

- Hypervolume (HV) (Zitzler et al. 2003), which indicates the area in the objective space that is dominated by at least one solution of the archive. HV of a given Pareto archive $\hat{A}$ is the Lebesgue measure of the set $\bigcup_{z \in \hat{A}} H(z, r_*)$, where $r_* \in \mathbb{R}^J$ is a reference point dominated by each point in the archive and $H(z, r_*)$ is a hypercuboid defined by points z and r_* .

5.5 Adaptation of the methods to MSTSP

We use the 2-opt local search with two-edge exchange move, first proposed by Croes (1958) and applicable to TSP and many related problems. It consists in testing all pairs of non-adjacent edges in the tour in order to find the best pair of edges $\langle a, b \rangle$ and $\langle c, d \rangle$, such that replacing them with edges $\langle a, c \rangle$ and $\langle b, d \rangle$ results in a shorter tour.

Since local search is the most time-consuming part of each method, we use a speed-up technique, namely candidate lists, in 2-opt local search in the main phase of each method. This technique is able to reduce the running time significantly with only a very small degradation of the quality of the retrieved solutions (Lust and Jaskiewicz 2010). There are several ways of making the candidate lists. In this paper, we use the population of initial solutions improved by the local search to make a candidate list for each node. Specifically, the candidate list of a node a contains all nodes connected to a in at least one of the initial solutions. Then, we just consider the pairs of edges $\langle a, b \rangle$ and $\langle c, d \rangle$ such that c is in the candidate list of a or d is in the candidate list of b .

For the recombination of solutions, we use the distance-preserving crossover (DPX) operator (Freisleben and Merz 1996). DPX generates an offspring by putting the edges which are common in both parents to the offspring. The offspring is then completed by randomly selected edges which are not present in any of the parents. As a result, the generated offspring has the same distance (measured by the number of different edges) to both of its parents.

In preliminary experiments with MSTSP, we observed that the best results are obtained with linear scalarizing functions. Consequently, we used this type of function for this problem, similarly to Jaskiewicz (2002a) and Zhang and Li (2007).

5.6 Adaptation of the methods to TSPWP

In TSPWP, we use a local search which performs moves of four different types. In each iteration of the local search, all moves of every type are tested and the best move is performed.

- Edge exchange: this move works exactly like the two-edge exchange used in MSTSP. It can change the length

objective but cannot change profit, so other types of moves are necessary.

- Node insertion: in this move, a node which is not present in the current tour is added to the best position according to the current scalarizing function in the tour.
- Node deletion: in this move, a node is deleted from the tour.
- Node exchange: in this move, a node present in the tour is exchanged for another node that is not present in the tour. The new node replaces the old one in the same position in the tour.

Algorithm 6 DPX operator for TSPWP

Parameter ↓: The parents X_1, X_2

Parameter ↑: The offspring X_3

$comSet :=$ set of common edges and nodes in X_1 and X_2

$remSet :=$ set of remaining nodes not present in $comSet$

$fragmentSet := comSet$

$expectedNumberOfNodes :=$ average number of nodes in X_1 and X_2

$addProbability := \frac{expectedNumberOfNodes - |comSet|}{|remSet|}$

for each node $a \in remSet$ **do**

add a to $fragmentSet$ with probability $addProbability$

end for

Combine the edges and nodes from $fragmentSet$ randomly creating a circular path

As the recombination operator, we use an extended version of the DPX operator, in which we collect both common nodes and common edges between two parents. We then randomly add some of remaining nodes to obtain the expected number of nodes, equal to the average number of nodes in the parents. The fragments (edges and nodes) are then combined randomly, creating a circular path. The extended version of the DPX operator is given in Algorithm 6.

In preliminary experiments with TSPWP, we observed that the best results are achieved through mixed scalarizing functions with weights 0, 999 for the Chebyshev scalarizing function, and 0, 001 for the linear scalarizing function. Thus, these mixed scalarizing functions were used for this problem.

Since cost and profit objectives may have very different ranges, we normalize their values using certain approximate ranges of the objectives in the Pareto front. Specifically, the approximate ranges were retrieved at the beginning of each method, by running local search with two scalarizing functions with weight vectors (0.999, 0.001) and (0.001, 0.999).

5.7 Adaptation of the methods to MOSCP

In MOSCP, the local search is performed based on a neighborhood operator, which is guided by a scalarizing function

and defined as follows (Jaszkiewicz 2004): first, a randomly selected column is removed from the current solution, which leads to an unfeasible solution. The solution is then repaired in a greedy manner by inserting columns with the lowest ratio of:

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

The column removed in the first step is not considered by the greedy procedure; therefore, the neighborhood operator always produces a new solution. The whole neighborhood of the current solution is tested, and the best local move is performed. The recombination operator is also based on the distance-preserving crossover idea. An offspring is generated as follows: first, all columns common to both parents are inserted into the offspring. Then, the columns which appear in only one of the parents are inserted into the offspring with 50% probability. Since this procedure cannot guarantee covering all rows, in the last step, all uncovered rows are covered with randomly selected columns.

5.8 Experiment design

We present the average values of the quality indicators for 10 executions for each method and each instance of a given problem. We compare four methods: Multiobjective Multiple Start Local Search (MOMSLS), JMOGLS, EMOGLS, and MOEA/D for MSTSP, TSPWP, and MOSCP. To avoid influence of implementation differences, all methods were implemented in Java sharing as much of the code as possible.

MOMSLS is a simple method employing multiple runs of local search. Each run starts with a random initial solution and uses a scalarizing function with a random weight vector. In other words, MOMSLS is similar to the initial phases of JMOGLS and MOEA/D and is therefore a natural reference for JMOGLS and MOEA/D. The use of recombination in these methods should ensure better performance than that of MOMSLS.

Two different types of instances of MTSP have been used:

- *Euclidean instances* in this group, the distances between the edges correspond to the Euclidean distances between points randomly located in a plane with uniform distribution. Euclidean and Kro instances are included in this group.
- *Cluster instances* the points are randomly clustered in a plane and the distances between points correspond to their Euclidean distance.

For two- and three-objective instances of MSTSP, we use the instances which were proposed in Lust's library³ (Lust and Teghem 2010). As mentioned in Sect. 5.2, in TSPWP the first objective is the length of the tour, while the second objective is the collected profit. In our experiment, the first objective comes from either Euclidean or Cluster instances, and the profits are generated randomly from a uniform distribution in a given range.

We used bi-objective instances of MOSCP from Lust's library⁴ (Lust and Tuytens 2014). We generated three-objective instances of MOSCP by combining two bi-objective instances. Two objectives came from the first instance, and the third objective was the first from the second instance. The instances are available from the authors upon request.

For a fair comparison, the number of weights in MOEA/D and EMOGLS, and the number of initial solutions in JMOGLS, were set the same way in all methods. The number of iterations was also the same in all methods. As a consequence, the same number of local search runs and recombinations was performed in all methods. The number of iterations in MOMSLS was also the same as the number of iterations in JMOGLS, EMOGLS, and MOEA/D. By one iteration, we mean one run of local search (MOMSLS and initial phases of other methods), or one recombination and one run of local search (main phases of JMOGLS, EMOGLS, and MOEA/D).

The parameters of each method were set experimentally, based on the best choice principle. The parameters setting for particular instances are listed in Table 1. The size of the neighborhood in MOEA/D is set to 20, the probability of choosing parents from a subset of solutions corresponding to the neighbor weight vectors is set to 0.9, and the number of solutions which will be updated in each iteration is set to 2. The expected rank value in JMOGLS and EMOGLS is set to 10, 5, 4 for instances KroAB100, ClusterAB300, and EuclideanAB500, respectively. For instances KroABC100 and ClusterABC300, it is set to 10 and 8, respectively. For all other instances, the expected rank is set to 10.

The number of weight vectors used in the quality measure R was set to 1000 for all bi-objective instances, and to 7562 for all three-objective instances. The reference points are defined by the minimum values of each objective in the reference sets.

6 Results and discussion

Tables 2, 3, 4, 5 and 6 present the mean and the standard deviation of quality indicators values obtained by MOMSLS,

Table 1 Parameter settings

Problem	Number of generations	Number of weight vectors
MSTSP-2obj	50	101
MSTSP-3obj	5	3403
TSPWP	17	301
MOSCP-2obj	17	301
MOSCP-3obj	5	3403

JMOGLS, EMOGLS, and MOEA/D on MSTSP, TSPWP, and MOSCP instances. The best results for each instance are highlighted.

The main observations are:

- In all cases, MOMSLS performs worst compared to all other methods. This confirms that the use of recombination to generate a starting solution for local search heavily influences the performance of MOEAs.
- For all test instances, the best results were obtained by either JMOGLS or EMOGLS. It is also apparent that JMOGLS and EMOGLS work very similarly.
- MOEA/D never obtains the best values compared to other tested methods. Furthermore, MOEA/D results are substantially worse compared to JMOGLS and EMOGLS results.

In order to test the statistical significance of the differences, we performed the nonparametric Wilcoxon signed-rank test, with $\alpha = 0.05$. We found, in all cases, that MOEA/D was significantly worse than JMOGLS and EMOGLS. The two latter methods did not differ in any cases except of the ClusterABC300 instance of MSTSP, for which EMOGLS was slightly better. As can be seen in Table 3, the difference between EMOGLS and JMOGLS in this instance was, however, much smaller than the difference between MOEA/D and any other method.

The main observation from the whole experiment is that the main design element which influences the performance of scalarizing function-based multiobjective evolutionary algorithms is the choice of the mechanism for parents selection. MOEA/D and EMOGLS differ only in this mechanism, and their performance differs significantly.

Furthermore, since EMOGLS performs better than MOEA/D, and the two methods differ only in parents selection, we may conclude that the selection mechanism used in EMOGLS (and JMOGLS) is a better choice for the problems considered in this paper. In our opinion, this is mainly due to the fact that in JMOGLS and EMOGLS parents are selected from a larger population of solutions, i.e., from the Pareto archive, while in MOEA/D the population is bounded by the predefined number of weight vectors. Selecting parents from

³ <http://www.desir.lip6.fr/~lustt/Research.html#MOTSP>.

⁴ <http://www.desir.lip6.fr/~lustt/Research.html#MOSCP>.

Table 2 Results for 2-obj instances of MSTSP

Instance	Quality	MOMSLS	JMOGLS	EMOGLS	MOEA/D
KroAB100	<i>R</i>	10765.39 (7.92)	10408.17 (11.27)	10405.71 (8.81)	10508.75 (26.52)
	<i>HV</i>	21.71E+09 (5.72E+06)	21915.43E+06 (3.71E+06)	21915.56E+06 (4.32E+06)	21.85E+09 (7.94E+06)
ClusterAB300	<i>R</i>	27187.44 (17.24)	26221.41 (31.70)	26212.33 (47.61)	26612.22 (26.04)
	<i>HV</i>	211.35E+09 (2.20E+07)	2125.61E+08 (5.63E+07)	2125.42E+08 (6.07E+07)	211.88E+09 (2.77E+07)
EuclideanAB500	<i>R</i>	51015.59 (18.71)	49117.52 (55.62)	49119.42 (45.35)	49921.83 (49.81)
	<i>HV</i>	5.79E+11 (3.51E+07)	583.55E+09 (1.39E+08)	583.61E+09 (1.13E+08)	5.81E+11 (9.15E+07)

Table 3 Results for 3-obj instances of MSTSP

Instance	Quality	MOMSLS	JMOGLS	EMOGLS	MOEA/D
KroABC100	<i>R</i>	12708.28 (4.84)	12358.69 (5.29)	12353.63 (5.34)	12454.55 (4.2)
	<i>HV</i>	3.57E+15 (5.97E+11)	3633.83E+12 (7.87E+11)	3633.71E+12 (7.21E+11)	3.61E+15 (4.55E+11)
ClusterABC300	<i>R</i>	17026.97 (3.72)	16723.42 (3.05)	16701.25 (5.3)	16837.89 (4.23)
	<i>HV</i>	11.27E+16 (1.28E+13)	1130.35E+14 (1.86E+13)	1130.59E+14 (6.75E+12)	11.29E+16 (8.96E+12)

Table 4 Results for instances of TSPWP

Instance	Quality	MOMSLS	JMOGLS	EMOGLS	MOEA/D
KroAProfit100	<i>R</i>	0.16 (3.59E-04)	0.1587 (1.20E-04)	0.1589 (1.81E-04)	0.159 (1.56E-04)
	<i>HV</i>	4.42E+08 (1.31E+06)	467.97E+06 (4.47E+05)	467.25E+06 (5.86E+05)	46.44E+07 (7.88E+05)
ClusterAProfit300	<i>R</i>	0.156 (3.41E-04)	0.1445 (2.47E-04)	0.1446 (3.11E-04)	0.151 (3.34E-04)
	<i>HV</i>	3.01E+09 (6.47E+06)	33.25E+08 (6.28E+06)	33.37E+08 (1.14E+07)	3.22E+09 (8.39E+06)

Table 5 Results for 2-obj instances of MOSCP

Instance	Quality	MOMSLS	JMOGLS	EMOGLS	MOEA/D
2scp41A	<i>R</i>	180.77 (0.28)	179.19 (0.09)	179.16 (0.03)	179.35 (0.18)
	<i>HV</i>	38.18E+05 (5.12E+03)	3840.54E+03 (8.39E+02)	3840.15E+03 (2.31E+02)	38.37E+05 (3.23E+03)
2scp61A	<i>R</i>	549.09 (0.78)	537.80 (0.22)	537.88 (0.35)	538.68 (0.34)
	<i>HV</i>	66.61E+06 (2.49E+04)	67034.46E+03 (3.08E+04)	67034.62E+03 (3.05E+04)	66.96E+06 (3.36E+04)
2scp81A	<i>R</i>	1077.91 (1.35)	1050.83 (0.66)	1050.70 (0.36)	1052.75 (0.69)
	<i>HV</i>	16.71E+07 (4.79E+04)	168.21E+06 (5.13E+04)	168.19E+06 (3.46E+04)	168.09E+06 (4.19E+04)

Table 6 Results for 3-obj instances of MOSCP

Instance	Quality	MOMSLS	JMOGLS	EMOGLS	MOEA/D
3scp41A	<i>R</i>	184.14 (0.27)	180.24 (0.09)	180.24 (0.14)	180.86 (0.12)
	<i>HV</i>	14.61E+09 (7.94E+06)	14738.61E+06 (2.26E+06)	14738.41E+06 (3.17E+06)	147.15E+08 (3.06E+06)
3scp61A	<i>R</i>	707.72 (1.38)	665.06 (0.63)	665.88 (0.77)	673.27 (1.23)
	<i>HV</i>	8.31E+11 (7.85E+08)	855.48E+09 (2.97E+08)	855.28E+09 (3.96E+08)	85.04E+10 (5.06E+08)
3scp81A	<i>R</i>	1362.97 (1.58)	1271.18 (2.49)	1272.83 (1.99)	1298.7 (1.59)
	<i>HV</i>	3.67E+12 (3.69E+09)	38.30E+11 (3.40E+09)	38.29E+11 (2.34E+09)	3.77E+12 (2.38E+09)

Table 7 Influence of number of weight vectors for instance 2scp81A

Number of Weight vector	Quality	JMOGLS	EMOGLS	MOEA/D
101 weight vectors	<i>R</i>	1050.69 (0.37)	1051.77 (0.53)	1054.60 (1.58)
	<i>HV</i>	186.79E+06 (3.95E+04)	186.58E+06 (4.19E+04)	186.39E+06 (7.15E+04)
201 weight vectors	<i>R</i>	1050.92 (0.38)	1051.19 (0.48)	1052.97 (0.32)
	<i>HV</i>	1867.82E+05 (4.03E+04)	1867.41E+05 (4.08E+04)	186.63E+06 (2.38E+04)
301 weight vectors	<i>R</i>	1050.83 (0.66)	1050.70 (0.36)	1052.75 (0.69)
	<i>HV</i>	168.21E+06 (5.13E+04)	168.19E+06 (3.46E+04)	168.09E+06 (4.19E+04)

the Pareto archive ensures better diversity and allows us to avoid a premature convergence in JMOGLS and EMOGLS.

On the other hand, since EMOGLS and JMOGLS perform very similarly, we may conclude that selecting weight vectors either at random or from a set of evenly distributed weight vectors does not substantially influence algorithm performance.

Note, however, that the number of weight vectors should be large enough to ensure that EMOGLS works comparably to JMOGLS. To illustrate it, in Table 7 we present the results for instance 2scp81A with 101, 201, and 301 weight vectors obtained with JMOGLS, EMOGLS, and MOEA/D with a constant total number of iterations. Note that, in the case of JMOGLS, the number of weight vectors influences only the number of initial solutions. These results show that, with 101 and 201 weight vectors, results of EMOGLS are worse than those of JMOGLS and, at the same time, significantly better than results of MOEA/D. However, for 301 weight vectors, results of JMOGLS and EMOGLS are similar but at the same time significantly better than results of MOEA/D. Also note that the performance of MOEA/D improves with the growing number of weight vectors, probably due to an increasing population size from which parents are selected. The performance of MOEA/D improves, however, more slowly than the performance of EMOGLS.

7 Conclusions

In the paper, we experimentally compared the performance of JMOGLS and MOEA/D algorithms. We used three different multiobjective combinatorial problems, i.e., the multiobjective symmetric traveling salesperson problem, traveling salesperson problem with profits, and multiobjective set covering problem. In this comparison, we focused on identifying the main design elements which influence their performance. To our best knowledge, this is the first such systematic comparison of these algorithms. Such analysis provides deeper insight into the sources of variation between different methods.

Our results indicate that the main factor influencing the performance of algorithms is the selection of parents. The choice of parents with tournament selection used in JMOGLS and EMOGLS performed better on all three problems used in the experiment.

We have also proposed a slight modification of JMOGLS in which parents are selected from the Pareto archive without the use of an additional population of solutions.

We have obtained similar results for three different multiobjective combinatorial problems. Without a doubt, further computational studies on other combinatorial and continuous problems, including problems with higher numbers of objectives, would be beneficial to assess whether the same pattern holds in other cases.

Recently, Zhang et al. proposed (Li et al. 2015) certain extended versions of MOEA/D in which the selection of solutions for the current population is performed differently, explicitly taking into account both the quality and diversity of solutions associated with particular weight vectors (sub-problems). Thus, an interesting direction for further research would be to compare JMOGLS/EMOGLS with this new version of MOEA/D.

Acknowledgements We also would like to thank Dr. Manuel Lopez-Ibez (University of Manchester, UK) for his helpful comments on this work. This study was funded by the Polish National Science Center, Grant No. UMO-2013/11/B/ST6/01075.

Compliance with ethical standards

Conflict of interest The authors declare no conflict of interest.

Human and animals rights This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Abualigah LM, Khader AT, Hanandeh ES (2018) A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis. *Eng Appl Artif Intell* 73:111–125. <https://doi.org/10.1016/j.engappai.2018.05.003>
- Carvalho Rd, Saldanha RR, Gomes B, Lisboa AC, Martins A (2012) A multi-objective evolutionary algorithm based on decomposition for optimal design of Yagi-Uda antennas. *IEEE Trans Magn* 48(2):803–806
- Croes GA (1958) A method for solving traveling-salesman problems. *Oper Res* 6(6):791–812
- Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. Wiley, Hoboken
- Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evolut Comput* 6(2):182–197
- Ding D, Wang G (2013) Modified multiobjective evolutionary algorithm based on decomposition for antenna design. *IEEE Trans Antennas Propag* 61(10):5301–5307
- Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. *Trans Sci* 39(2):188–205
- Freisleben B, Merz P (1996) New genetic local search operators for the traveling salesman problem. In: Voigt HM, Ebeling W, Rechenberg I, Schwefel HP (eds) *Parallel problem solving from nature-PPSN IV*. Springer, Berlin, Heidelberg, pp 890–899
- Gong M, Ma L, Zhang Q, Jiao L (2012) Community detection in networks by using multiobjective evolutionary algorithm with decomposition. *Phys A: Stat Mech Appl* 391(15):4050–4060
- Hansen MP, Jaskiewicz A (1998) Evaluating the quality of approximations to the non-dominated set. Department of Mathematical Modelling, Technical University of Denmark, IMM
- Ishibuchi H, Murata T (1998) A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Trans Syst Man Cybernet Part C: Appl Rev* 28(3):392–403
- Ishibuchi H, Hitotsuyanagi Y, Tsukamoto N, Nojima Y (2009) Use of biased neighborhood structures in multiobjective memetic algorithms. *Soft Comput* 13(8):795–810. <https://doi.org/10.1007/s00500-008-0352-6>
- Ishibuchi H, Nakashima Y, Nojima Y (2011) Performance evaluation of evolutionary multiobjective optimization algorithms for multiobjective fuzzy genetics-based machine learning. *Soft Comput* 15(12):2415–2434. <https://doi.org/10.1007/s00500-010-0669-9>
- Ishibuchi H, Akedo N, Nojima Y (2015) Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Trans Evol Comput* 19(2):264–283
- Jaskiewicz A (2002a) Genetic local search for multi-objective combinatorial optimization. *Eur J Oper Res* 137(1):50–71
- Jaskiewicz A (2002b) On the performance of multiple-objective genetic local search on the 0/1 knapsack problem—a comparative experiment. *IEEE Trans Evol Comput* 6(4):402–412. <https://doi.org/10.1109/TEVC.2002.802873>
- Jaskiewicz A (2003) Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem. *IEEE Trans Evol Comput* 7(2):133–143. <https://doi.org/10.1109/TEVC.2003.810759>
- Jaskiewicz A (2004) A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm. *Ann Oper Res* 131(1–4):135–158
- Jaskiewicz A, Kominek P (2003) Genetic local search with distance preserving recombination operator for a vehicle routing problem. *Eur J Oper Res* 151(2):352–364. [https://doi.org/10.1016/S0377-2217\(02\)00830-5](https://doi.org/10.1016/S0377-2217(02)00830-5). 18th EURO Summer-Winter-Institute on Meta-Heuristics in Combinatorial Optimization (ESWI XVIII), Switzerland
- Jozefowicz N, Glover F, Laguna M (2008) Multi-objective metaheuristics for the traveling salesman problem with profits. *J Math Model Algorithms* 7(2):177–195
- Kafafy A, Bounekkar A, Bonnevey S (2012) Hybrid metaheuristics based on moea/d for 0/1 multiobjective knapsack problems: a comparative study. In: 2012 IEEE congress on evolutionary computation, pp 1–8. <https://doi.org/10.1109/CEC.2012.6253015>
- Ke L, Zhang Q, Battiti R (2013) MOEA/D-ACO: a multiobjective evolutionary algorithm using decomposition and antcolony. *IEEE Trans Cybern* 43(6):1845–1859
- Knowles J, Corne D (1999) The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: *Proceedings of the 1999 Congress on IEEE Evolutionary Computation*. CEC 99, vol 1
- Kolen A, Pesch E (1994) Genetic local search in combinatorial optimization. *Discrete Appl Mathe* 48(3):273–284. [https://doi.org/10.1016/0166-218X\(92\)00180-T](https://doi.org/10.1016/0166-218X(92)00180-T)
- Konstantinidis A, Yang K (2011) Multi-objective energy-efficient dense deployment in wireless sensor networks using a hybrid problem-specific MOEA/D. *Appl Soft Comput* 11(6):4117–4134. <https://doi.org/10.1016/j.asoc.2011.02.031>. <http://www.sciencedirect.com/science/article/pii/S1568494611000950>
- Li K, Fialho A, Kwong S, Zhang Q (2014) Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput* 18(1):114–130
- Li K, Kwong S, Zhang Q, Deb K (2015) Interrelationship-based selection for decomposition multiobjective optimization. *IEEE Trans Cybern* 45(10):2076–2088
- Liu HL, Gu F, Zhang Q (2014) Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Trans Evol Comput* 18(3):450–455
- Lust T, Jaskiewicz A (2010) Speed-up techniques for solving large-scale Bi-Objective TSP. *Comput Oper Res* 37(3):521–533
- Lust T, Teghem J (2010) Two-phase pareto local search for the biobjective traveling salesman problem. *J Heuristics* 16(3):475–510. <https://doi.org/10.1007/s10732-009-9103-9>
- Lust T, Tuytens D (2014) Variable and large neighborhood search to solve the multiobjective set covering problem. *J Heuristics* 20(2):165–188. <https://doi.org/10.1007/s10732-013-9236-8>
- Mei Y, Tang K, Yao X (2011) Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. *IEEE Trans Evol Comput* 15(2):151–165
- Murata T, Ishibuchi H, Gen M (2001) Specification of genetic search directions in cellular multi-objective genetic algorithms. In: Zitzler E, Thiele L, Deb K, Coello Coello CA, Corne D (eds) *Evolutionary multi-criterion optimization*. Springer, Berlin, pp 82–95
- Neri F, Cotta C, Moscato P (2012) *Handbook of memetic algorithms*, vol 379. Multiobjective memetic algorithms. Springer, Berlin
- Sengupta S, Das S, Nasir M, Panigrahi BK (2013) Multi-objective node deployment in WSNs. In: *search of an optimal trade-off among coverage, lifetime, energy consumption, and connectivity*. *Eng Appl Artif Intell* 26(1):405–416
- Sengupta S, Das S, Nasir M, Vasilakos AV, Pedrycz W (2012) An evolutionary multiobjective sleep-scheduling scheme for differentiated coverage in wireless sensor networks. *IEEE Trans Syst Man Cybern Part C: Appl Rev* 42(6):1093–1102
- Sindhya K, Ruuska S, Haanpää T, Miettinen K (2011) A new hybrid mutation operator for multiobjective optimization with differential evolution. *Soft Comput* 15(10):2041–2055. <https://doi.org/10.1007/s00500-011-0704-5>
- Steuer RE (1985) *Multiple criteria optimization: theory, computation and application*. Wiley, New York
- Trivedi A, Srinivasan D, Pal K, Saha C, Reindl T (2015) Enhanced multiobjective evolutionary algorithm based on decomposition for solving the unit commitment problem. *IEEE Trans Ind Inform* 11(6):1346–1357

- Yu G (2013) Industrial applications of combinatorial optimization, 16th edn. Springer, Berlin
- Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput* 11(6):712–731
- Zhang Q, Liu W, Li H (2009) The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In: 2009 IEEE congress on evolutionary computation, pp 203–208 . <https://doi.org/10.1109/CEC.2009.4982949>
- Zhang Q, Liu W, Tsang E, Virginas B (2010) Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Trans Evol Comput* 14(3):456–474
- Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG (2003) Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans Evol Comput* 7(2):117–132. <https://doi.org/10.1109/TEVC.2003.810758>
- Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans Evol Comput* 3(4):257–271

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.