J Math Model Algor (2013) 12:233–251
DOI 10.1007/s10852-012-9208-2

A New Hybrid Evolutionary Multiobjective Algorithm Guided by Descent Directions

Roman Denysiuk · Lino Costa · Isabel Espírito Santo

Received: 29 July 2012 / Accepted: 14 August 2012 / Published online: 1 September 2012

Abstract Hybridization of local search based algorithms with evolutionary algorithms is still an under-explored research area in multiobjective optimization. In this paper, we propose a new multiobjective algorithm based on a local search method. The main idea is to generate new non-dominated solutions by adding a linear combination of descent directions of the objective functions to a parent solution. Additionally, a strategy based on subpopulations is implemented to avoid the direct computation of descent directions for the entire population. The evaluation of the proposed algorithm is performed on a set of benchmark test problems allowing a comparison with the most representative state-of-the-art multiobjective algorithms. The results show that the proposed approach is highly competitive in terms of the quality of non-dominated solutions and robustness.

Keywords Multiobjective optimization · Evolutionary algorithms · Pattern search · Performance assessment

1 Introduction

Many real-world optimization problems involve the simultaneous optimization of several conflicting objectives. These problems are called multiobjective optimization

R. Denysiuk (✉)
Algoritmi R&D Center, University of Minho, Minho, Portugal
e-mail: roman.denysiuk@algoritmi.uminho.pt

L. Costa · I. Espírito Santo
Department of Production and Systems Engineering, University of Minho, Minho, Portugal

L. Costa
e-mail: lac@dps.uminho.pt

I. Espírito Santo
e-mail: iapinho@dps.uminho.pt

problems (MOPs). Without loss of generality, a multiobjective optimization problem with m objectives and n decision variables can be formulated mathematically as follows:

$$\begin{aligned} &\text{minimize: } \mathbf{F}(\mathbf{x}) = (f_1(x), f_2(x), \ldots, f_M(x))^{\mathrm{T}} \\ &\text{subject to: } \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the decision vector defined in the decision space $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$, $\mathbf{l}$ and $\mathbf{u}$ are the lower and upper bounds of the decision variables, respectively, and $\mathbf{F}(\mathbf{x})$ is the objective vector defined in the objective space $\mathbb{F}^M$. Since the objective space is partially ordered, solutions are compared on the base of Pareto dominance. For a multiobjective minimization problem, a solution $\mathbf{a}$ is said to dominate a solution $\mathbf{b}$, $\mathbf{a} \prec \mathbf{b}$, iff $\forall m \in \{1, \ldots, M\} : f_m(\mathbf{a}) \leq f_i(\mathbf{b})$ and $\exists j \in \{1, \ldots, M\} : f_j(\mathbf{a}) < f_j(\mathbf{b})$. Opposing single-objective optimization, the solution to multiobjective optimization problems is not a single solution, but a set of non-dominated solutions called the Pareto-optimal set. A solution $\mathbf{a}$ is called a Pareto-optimal, iff $\nexists \mathbf{b} \in \Omega : \mathbf{b} \prec \mathbf{a}$, or there is no feasible solution $\mathbf{b}$ such that $\mathbf{b}$ dominates $\mathbf{a}$. The main goal of the multiobjective optimization is to obtain the set of Pareto-optimal solutions.

Classical methods to solve multiobjective optimization problems mostly rely on scalarization of the multiple objectives and require repeated runs with different parameters sets to find several approximations to the Pareto-optimal solutions [14]. However, there are a few classical methods (stochastic and deterministic) which either attempt to find multiple Pareto-optimal solutions in a single simulation run, or attempt to solve multiple scalarized problems such that a good diversity among resulting solutions is maintained [15].

Evolutionary algorithms have been successfully applied to solve a large number of multiobjective optimization problems [2]. Compared with conventional optimization methods for solving MOPs, multiobjective evolutionary algorithms(MOEAs) generate multiple Pareto optimal solutions in a single run, since they can explore different regions of the solution space and maintain a set of diverse solutions. A number of MOEAs that achieved a good performance have been proposed [3, 12, 18, 19].

However, several drawbacks are related with these approaches, such as high computational cost or slow convergence at regions close to the Pareto-optimal front. The local search algorithms have shown good ability to circumvent such difficulties in the single-objective case. The study of local search based methods in the multiobjective optimization context is an essential step for further hybridization with evolutionary algorithms since this is still an under-explored research area. Nevertheless, some studies have been conveyed on combining the global and local search techniques in multiobjective optimization [9, 10].

In this paper we propose the Descent Directions based Multiobjective Algorithm (DDMOA) which is a hybrid evolutionary multiobjective algorithm. It is based on the local search principle, that during the iterations enables the given population to proceed towards the Pareto-optimal front. The main idea for generating new solutions relies on the approach proposed by Timmel [16]. The presence of gradients in this approach imposes significant limitations, regarding the differentiability of the problem being solved. Therefore, in order to overcome this drawback, descent directions for each objective are calculated to avoid the computation of derivatives.

The remainder of this paper is organized as follows. In Section 2, we introduce a classical method for multiobjective optimization whose main idea was extended and used in our algorithm. In Section 3, we give a detailed description of DDMOA. In Section 4, we present the methodology used to assess the performance of the proposed approach. In Section 5, we discuss the results of experimental study and show comparison with other MOEAs. In Section 6, we conclude and address some possible future work.

2 Timmel's Method

In this section, we describe a classical method for finding multiple Pareto-optimal solutions of a differentiable multiobjective optimization problem proposed by Timmel [16], whose main idea was extended and used in our hybrid algorithm. It is a population based approach, where for a given parent solution, a child solution is created in the following manner:

$$\mathbf{x}^{\text{child}} = \mathbf{x}^{\text{parent}} - t_k \sum_{m=1}^{M} u_m \nabla f_m(\mathbf{x}^{\text{parent}}) \tag{2}$$

where $\mathbf{x}^{\text{parent}}$ is the parent decision vector, $\mathbf{x}^{\text{child}}$ is the generated child decision vector, $\nabla f_m(\mathbf{x}^{\text{parent}})$ is the gradient of m-th objective, t_k is the step size at the k-th iteration and u_m is a uniformly distributed random number between zero and one ($u_m \sim \mathbb{U}(0, 1)$).

The above formulation ensures that not all objective functions can be worsened simultaneously. Thus, the child solution $\mathbf{x}^{\text{child}}$ is either non-dominated when compared to the parent solution $\mathbf{x}^{\text{parent}}$, or it dominates the parent solution.

Figure 1 shows the creation of child solutions $\{a, b, c, d\}$ (denoted by the lowercase letters) from the corresponding parents (denoted by the capital letters). The vectors of the gradients are presented for the parent solution C. Adding to the parent solution, the linear combination of the gradients generates the child c. For the other parent solutions children are generated in the same way. Not all children

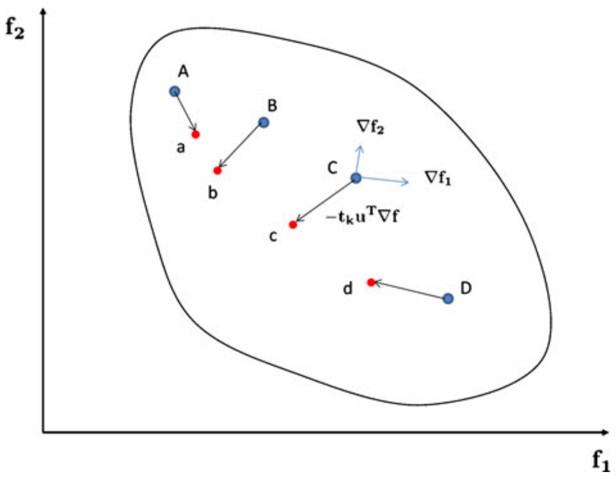


Fig. 1 Illustration of generating the child solutions $\{a, b, c, d\}$

dominate their parents. After the child population is created, it is combined with the parent population and only the non-dominated solutions are retained. Then, this set becomes the parent population, and this procedure is repeated for a pre-defined number of iterations.

The above approach is only applicable to differentiable optimization problems, and this is the main limitation of this method. In order to avoid the use of the gradient of the objectives, we calculate descent directions for each objective. Thus, we propose the following formulation to generate a child solution from a given parent:

$$\mathbf{x}^{\text{child}} = \mathbf{x}^{\text{parent}} + \sigma_k \sum_{m=1}^{M} u_m \mathbf{s}_m, \tag{3}$$

where σ_k is the step size in the k-th iteration, u_m is a uniformly distributed random number between 0 and 1 ($u_m \sim \mathbb{U}(0, 1)$), and $\mathbf{s}_m$ is the descent direction for m-th objective.

3 Descent Directions Based Multiobjective Algorithm

In this section, we present the Descent Directions based Multiobjective Algorithm (DDMOA). It is a hybrid evolutionary multiobjective algorithm which borrows the idea of generating new child solutions from the above described classical method and comprises general features of evolutionary algorithms. Its pseudo-code is presented in Algorithm 1.

Algorithm 1 DDMOA

```
Require: $\mu > 0$, $\delta_0 > 0$, $\delta_{\text{tol}} > 0$, $\alpha > 0$, $\sigma_0 > 0$
Ensure: $P_{\text{ND}}$                ▷ approximation to the Pareto-optimal set
 1: $P^0 \leftarrow$ initialPopulation($\mu$)
 2: $k \leftarrow 0$
 3: $\sigma^k \leftarrow \sigma_o$
 4: $P^k \leftarrow$ nondominationSorting($P^0$)
 5: repeat
 6:     $(S^k, A^k) \leftarrow$ descentDirections($P^k$)
 7:     $R^k \leftarrow$ tournamentSelection($P^k$)
 8:     $O^k \leftarrow$ generateOffspring($R^k, S^k, \sigma^k$)
 9:     $P^{k+1} \leftarrow$ nondominationSorting($P^k \cup O^k \cup A^k$)
10:     $P^{k+1} \leftarrow$ diversityPreserving($P^{k+1}$)
11:     $\sigma^{k+1} \leftarrow$ updateStepSize($k$)
12:     $k \leftarrow k + 1$
13: until the stopping criterion is met
14: $P_{\text{ND}} \leftarrow P^k$
```

DDMOA starts by randomly generating an initial population P^0 of μ individuals (**initialPopulation** procedure). All solutions in the population are tuples of the form (x, δ), where δ is step size for coordinate search. Next, the population is evaluated and all dominated solutions are removed from the population (**nondominationSorting** procedure). The iterative process is started by computing descent directions for each objective for all solutions in the population (**descentDirections** procedure), which returns a matrix of descent directions S^k, and a temporary archive A^k, with all non-dominated solutions found during coordinate search. Then, in order to select a pool of parent solutions, R^k, a binary tournament selection based on crowding distances is performed (**tournamentSelection** procedure). After the pool of parent solutions is selected, a set of child solutions O^k is created (**generateOffspring** procedure), using Eq. 3. Then, the multi-set that includes the set of parent solutions, the generated offspring and the temporary archive is sorted (**nondominationSorting** procedure), returning only non-dominated solutions and forming a new population. If the number of solutions in the new population is greater than the pre-defined population size, solutions which reside in less crowded regions are retained for the next iteration (**diversityPreserving** procedure). After that, the step size σ^k used to generate offspring is updated. If the stopping criterion is met, the iterative process terminates, and the algorithm returns the set of non-dominated solutions as an approximation to the Pareto-optimal front for a given problem, otherwise the algorithm proceeds to the next iteration. The following two stopping conditions are used: (i) the maximum number of objective function evaluations is reached, and (ii) $\delta \leq \delta_{\text{tol}}$ for all solutions in the population.

In the following subsections the components of DDMOA are discussed in more detail.

3.1 Initial Population

The initial population P^0 can be generated in multiple ways. It can be either generated randomly such that all the variables are inside the search space or can be uniformly sampled. We choose to create the initial population using Latin hypercube (LH) sampling [13] since it gives a good overall random distribution of the population in the variable space. Let the size of the population be N and the number of variables be n. Let the lower and upper bounds of variable i be l_i and u_i, respectively. To generate a LH sample, the variable range is divided into N equal segments of size $\frac{u_i - l_i}{N}$ each, and a real random number is generated in each segment. Then a random permutation of integers from 1 to N is generated and the individual with index i is assigned with a value located at the $\pi(i)$–th position in the permutation. This process is repeated for all the variables. This ensures that the resultant population spans the entire decision space, it is sufficiently random and is free from any biases.

3.2 Descent Directions

The distinctive feature of our algorithm is that it needs to compute m descent directions for each solution in the current population to allow offspring generation. Therefore, each iteration is started by **descentDirections** procedure, which allows to compute descent directions for each objective for all solutions in the population. We present the pseudo-code for this procedure in Algorithm 2.

The population is sorted for the first objective in ascending order and partitioned into α equal parts. Thus, α subpopulations are defined in order to promote different reference points for the computation of descent directions. It follows that a descent

Algorithm 2 descentDirections

```
Require: $P$
Ensure: $(S,A)$
 1: $A \leftarrow \{\}$
 2: for $m = 1 \ldots M$ do
 3:     sort population $P$ in ascending order according to $f_m$
 4:     partition sorted $P$ into $\alpha$ subpopulations:
 5:     $P = \{p_1, p_2, .., p_\alpha\}$
 6:     for $i = 1 \ldots \alpha$ do
 7:         identify the leader individual $\mathbf{x}_{\text{leader}} \in p_i$
 8:         $(\mathbf{s}_{\text{leader}}, A) \leftarrow$ coordinateSearch$(P, A, \mathbf{x}_{\text{leader}}, m)$
 9:         for $j = 1 \ldots |p_i|$ do
10:             $\mathbf{s}_{j,m} \leftarrow \mathbf{x}_{\text{leader}} - \mathbf{x}_j + \mathbf{s}_{\text{leader}}$
11:         end for
12:     end for
13: end for
```

direction, $\mathbf{s}_{\text{leader}}$, is computed using coordinate search method, in each subpopulation p_i for the leader $\mathbf{x}_{\text{leader}}$ (the solution with the lower objective function value among the other solutions in the subpopulation) and $\delta > \delta_{\text{tol}}$. For the rest of the solutions in the given subpopulation descent directions are calculated as:

$$\mathbf{s}_{j,m} = \mathbf{x}_{\text{leader}} - \mathbf{x}_j + \mathbf{s}_{\text{leader}}, \tag{4}$$

where $\mathbf{s}_{j,m}$ is a descent direction for j-th solution of the i-th subpopulation for the m-th objective, $\mathbf{x}_j$ is the decision vector of the j-th solution in the subpopulation, $\mathbf{x}_{\text{leader}}$ is the leader solution in the subpopulation, and $\mathbf{s}_{\text{leader}}$ is the descent direction for the leader. This procedure is repeated for the other objectives. At the end, M descent directions are associated with each solution in the population. Finding such descent directions avoids the direct calculation of descent directions for the entire population using coordinate search, which is computationally expensive, allowing to reduce significantly the number of function evaluations.

3.3 Parent Selection

In order to select a pool R^k with μ parent solutions, a binary tournament selection is performed (**tournamentSelection** procedure), based on crowding distances measured in the objective space, as proposed in [3]. Therefore, solutions with the higher crowding distances have more probability of creating offspring. However, as DDMOA uses only non-dominated solutions, the situation where the number of solutions in the population is very small or even equal to one may occur. So, if the number of solutions in the population is less or equal to the number of objective functions parents are randomly selected from given solutions, otherwise the usual tournament selection is performed. This selection process that combines crowding distances with a stochastic selection operator promotes a good spread of solutions in the objective space as well as the exploration of new promising areas of the decision space.

We present the pseudo-code for this procedure in Algorithm 3.

Algorithm 3 tournamentSelection

```
Require: $P$
Ensure: $R$
 1: $R \leftarrow \{\}$
 2: while $|R| < \mu$ do
 3:     if $|P| \leq M$ then
 4:         randomly pick $a \in P$
 5:         $R \leftarrow R \cup a$
 6:     else
 7:         randomly pick $a \in P, b \in P \wedge a \neq b$
 8:         if $a <_c b$ then                ▷ crowding distance of b is greater than a
 9:             $R \leftarrow R \cup b$
10:         else
11:             $R \leftarrow R \cup a$
12:         end if
13:     end if
14: end while
```

3.4 Offspring Generation

After a pool of parent solutions is created and M descent directions are associated with each solution in the population, a child population is generated (**generateOffspring** procedure). In order to guarantee that each new solution $\mathbf{x} = (x_1, \ldots, x_N)^T$ belongs to Ω (see Eq. 1), projection is applied to each component of the decision vector.

We present pseudo-code for this procedure in Algorithm 4.

Algorithm 4 generateOffspring

```
Require: $R, S, \sigma_k$
Ensure: $O$
1: for $i = 1 \ldots |R|$ do
2:     $O(i) \leftarrow R(i) + \sigma_k \sum_{m=1}^{M} u_m \mathbf{s}_m$
3:     $O(i) \leftarrow \min\{\max\{O(i), \mathbf{l}\}, \mathbf{u}\}$
4: end for
5: evaluate child population
```

3.5 Non-Dominated Sorting

The **nondominationSorting** procedure removes all dominated solutions from a given set. The following pseudo-code (Algorithm 5) is used for this purpose:

3.6 Diversity Preserving

If the size of the new population is greater than a pre-defined population size, $P^{k+1} > \mu$, solutions which have the minimum distance to the other solutions are iteratively

Algorithm 5 nondominationSorting

```
Require: $P$
Ensure: $P\prime$
 1: $P\prime \leftarrow \{\}$
 2: for $i = 1 \ldots |P|$ do
 3:     for $j = 1 \ldots |P|$ and $j \neq i$ do
 4:         if $P(j) \prec P(i)$ then
 5:             break
 6:         end if
 7:     end for
 8:     if $j = |P|$ then
 9:         $P\prime \leftarrow P\prime \cup P(i)$
10:     end if
11: end for
```

removed from the population P^{k+1} until $|P^{k+1}| = \mu$. If there are several individuals with minimum distance the tie is broken by considering the second smallest distances, and so on.

The pseudo-code for this procedure is presented in Algorithm 6.

Algorithm 6 diversityPreserving

```
Require: $P$
Ensure: $P$
1: while $|P| > \mu$ do
2:     find $a \in P$        ▷ solution with the lower distance to a neighboring solution
3:     $P \leftarrow P \backslash a$
4: end while
```

3.7 Step Size Adaptation

There is no common rule to update the step size σ^k in Eq. 3, but it must be done carefully to ensure convergence to the Pareto-optimal front. Thus, after the new population is formed, the step size used to generate offspring is updated (**updateStepSize** procedure). We use the following approach for updating:

$$\sigma^k = \max\left(\delta_{\text{tol}}, \frac{\sigma_0}{k}\right) \tag{5}$$

Using this method the step size σ decreases during the iterations from the initial value σ_0 at the first iteration and never becomes less then δ_{tol}.

4 Performance Assessment

The performance comparison of different multiobjective optimization algorithms is more difficult than in the case of single objective optimization. In the multiobjective case, the goal is to find a good approximation to the true Pareto-optimal set, and to obtain a well distributed subset of the whole Pareto-optimal frontier. Many ways of measuring the performance of multiobjective algorithms have been proposed in the literature. In our study, we compare the quality of the non-dominated sets obtained by the algorithms using two quality indicators: the unary additive epsilon indicator ($I^1_{\epsilon+}$) [21] to assess the convergence, and the hypervolume indicator (I^-_H) [20] to assess both the convergence and the diversity. Since we are dealing with stochastic algorithms and we want to provide the results with confidence, we compare the results on the individual benchmark functions using a standard two-sided Wilcoxon rank sum test. In this work we consider a confidence level of 95 % in the statistical tests, which means that the difference are unlikely to have occurred by chance with a probability of 95 %. For a detailed description of the use of non-parametric tests to analyze the performance of evolutionary algorithms we refer to [7]. Finally, we present the results in the form of performance profiles [5] providing good visualization and easiness of making inferences about the performance of the algorithm. A brief description of the performance profiles and the quality indicators are provided in this section.

4.1 Performance Profiles

The performance profiles were proposed to compare the performance of deterministic algorithms over a set of distinct optimization problems [5]. These performance profiles can be extended to the context of stochastic algorithms with some adaptations [1].

Let $\mathcal{P}$ and $\mathcal{S}$ be the set of problems and the set of solvers in comparison, respectively, and let $m_{p,s}$ be the performance metric required to solve problem $p \in \mathcal{P}$ by solver $s \in \mathcal{S}$. The comparison is based on performance ratios defined by

$$r_{p,s} = \frac{m_{p,s}}{\min\{m_{p,s} : s \in \mathcal{S}\}}$$

and the overall assessment of the performance of a particular solver s is given by

$$\rho_s(\tau) = \frac{1}{\text{total number of problems}}\{\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}\}.$$

For $\tau = 1$, $\rho_s(\tau)$ gives the probability that the solver s will win over the others in the set. Thus, for $\tau = 1$, the uppermost curve shows the algorithm with the highest percentage of problems with the best metric value. However, for large values of τ, $\rho_s(\tau)$ measures the solver robustness. Overall, the higher the ρ_s values, the better the solver is. Also, for solver s that performs the best on a problem p, $r_{p,s} = 1$. If $r_{p,s} = 2$, it means that the m-fold improvement by solver s on problem p is twice the best value found by another solver on the same problem p.

In this paper, we use quality indicators [11] to measure the quality of non-dominated sets Ω obtained by the algorithms which can be defined as a function $I : \Omega \to \mathbb{R}$. This mapping to the set of real numbers allows to apply statistical tools such as performance profiles. In this study we consider the ϵ-indicator ($I^1_{\epsilon+}$) and hypervolume indicator (I^-_H), which are positive and superior to zero, and a smaller value of the quality indicator corresponds to a better performance of the algorithm. Thus, we define the metric:

$$m_{p,s} = I_{\text{stats}}$$

where I_{stats} represents a statistic computed for a given quality indicator value obtained in several runs (e.g., minimum, median,...). A description of the epsilon and the hypervolume indicators follows.

Table 1 Median values of the epsilon indicator after 30 runs

	DDMOA	NSGA–II	IBEA	MOEA/D
Two-objective test problems				
ZDT1	$\mathbf{0.0063}^{\text{II,III,IV}}$	0.013^{IV}	$0.0082^{\text{II,IV}}$	0.0997
ZDT2	$\mathbf{0.0074}^{\text{II,III,IV}}$	$0.0158^{\text{III,IV}}$	0.029^{IV}	0.4135
ZDT3	$\mathbf{0.0066}^{\text{III,IV}}$	$0.0075^{\text{III,IV}}$	0.0216^{IV}	0.1788
ZDT4	$\mathbf{0.0006}^{\text{II,III,IV}}$	$0.0029^{\text{III,IV}}$	0.0087^{IV}	0.0675
ZDT6	$\mathbf{0.0002}^{\text{II,III,IV}}$	0.0166^{IV}	$0.0151^{\text{II,IV}}$	0.0413
DTLZ1	$\mathbf{0.0174}^{\text{II,III,IV}}$	0.1422	0.1206^{II}	0.1175
DTLZ2	$\mathbf{0.0088}^{\text{II,III}}$	0.013^{III}	0.0138	$0.0092^{\text{II,III}}$
DTLZ3	$\mathbf{0.0183}^{\text{II,III,IV}}$	0.1592	0.1731	0.1935
DTLZ4	$\mathbf{0.0091}^{\text{II,III,IV}}$	0.0133^{III}	0.0149	0.0123^{III}
DTLZ5	$\mathbf{0.0091}^{\text{II,III}}$	0.0125^{III}	0.0138	$0.0094^{\text{II,III}}$
DTLZ6	$0.0009^{\text{II,III}}$	0.5668	0.4662^{II}	$\mathbf{0.0007}^{\text{I,II,III}}$
DTLZ7	$\mathbf{0.0051}^{\text{II,III,IV}}$	$0.0093^{\text{III,IV}}$	0.0254^{IV}	0.2967
WFG1	0.0179^{III}	$\mathbf{0.0136}^{\text{I,III,IV}}$	0.0242	0.0171^{III}
WFG2	0.2514	0.193^{I}	0.1931^{I}	$\mathbf{0.0196}^{\text{I,II,III}}$
WFG3	0.0704	0.0148^{I}	$\mathbf{0.0083}^{\text{I,II,IV}}$	$0.0128^{\text{I,II}}$
WFG4	0.0532^{IV}	$0.016^{\text{I,IV}}$	$\mathbf{0.012}^{\text{I,II,IV}}$	0.0805
WFG5	0.085	0.0187^{I}	$0.0169^{\text{I,II}}$	$\mathbf{0.0167}^{\text{I,II,III}}$
WFG6	0.0349	0.0409	0.0465	$\mathbf{0.0111}^{\text{I,II,III}}$
WFG7	0.0242	0.0176^{I}	$0.0121^{\text{I,II}}$	$\mathbf{0.0098}^{\text{I,II,III}}$
WFG8	0.1471^{III}	0.1838^{III}	0.2246	$\mathbf{0.1264}^{\text{II,III}}$
WFG9	0.0331	0.0144^{I}	$\mathbf{0.0112}^{\text{I,II,IV}}$	$0.0128^{\text{I,II}}$
Three-objective test problems				
DTLZ1	$\mathbf{0.0023}^{\text{II,III,IV}}$	0.0824	$0.0179^{\text{II,IV}}$	0.1309
DTLZ2	$0.0811^{\text{II,IV}}$	0.1149^{IV}	$\mathbf{0.0775}^{\text{I,II,IV}}$	0.5403
DTLZ3	$\mathbf{0.0063}^{\text{II,III,IV}}$	0.1217^{IV}	$0.0487^{\text{II,IV}}$	0.3549
DTLZ4	$0.1282^{\text{III,IV}}$	$\mathbf{0.1175}^{\text{I,III,IV}}$	0.632^{IV}	0.6643
DTLZ5	$\mathbf{0.0111}^{\text{II,III,IV}}$	$0.0154^{\text{III,IV}}$	0.0407^{IV}	0.7908
DTLZ6	$\mathbf{0.0004}^{\text{II,III,IV}}$	0.5041	0.1919^{II}	$0.0688^{\text{II,III}}$
DTLZ7	$\mathbf{0.0489}^{\text{II,III,IV}}$	$0.0802^{\text{III,IV}}$	0.0904^{IV}	0.6863
WFG1	$0.0763^{\text{II,IV}}$	0.0902^{IV}	$\mathbf{0.0521}^{\text{I,II,IV}}$	0.3201
WFG2	0.1511^{IV}	$\mathbf{0.0967}^{\text{I,IV}}$	0.1949	0.1947
WFG3	$0.0751^{\text{III,IV}}$	$\mathbf{0.0541}^{\text{I,III,IV}}$	0.1109^{IV}	0.4534
WFG4	$0.1097^{\text{II,IV}}$	0.1282^{IV}	$\mathbf{0.0794}^{\text{I,II,IV}}$	0.7283
WFG5	0.1638^{IV}	$0.1171^{\text{I,IV}}$	$\mathbf{0.0752}^{\text{I,II,IV}}$	0.6741
WFG6	$0.0977^{\text{II,IV}}$	0.1485^{IV}	$\mathbf{0.0887}^{\text{I,II,IV}}$	0.854
WFG7	0.151^{IV}	$0.1069^{\text{I,IV}}$	$\mathbf{0.0697}^{\text{I,II,IV}}$	0.587
WFG8	$\mathbf{0.1809}^{\text{II,IV}}$	0.2328^{IV}	$0.1832^{\text{II,IV}}$	0.6951
WFG9	$0.1085^{\text{II,IV}}$	0.1267^{IV}	$\mathbf{0.0782}^{\text{I,II,IV}}$	0.7224

4.2 Epsilon Indicator

The unary additive epsilon indicator $I^1_{\epsilon+}$ is based on additive ϵ-dominance and defined with respect to a reference set R as:

$$I^1_{\epsilon+}(A) = I_{\epsilon+}(A, R) = \inf_{\epsilon \in \mathbb{R}} \{\forall \mathbf{a}' \in R \; \exists \mathbf{a} \in A : \; \mathbf{a} \preceq_{\epsilon} \mathbf{a}'\}. \tag{6}$$

Table 2 The median values of the hypervolume indicator after 30 runs

	DDMOA	NSGA–II	IBEA	MOEA/D
Two-objective test problems				
ZDT1	**0.2265**$^{\text{II,III,IV}}$	0.2315$^{\text{IV}}$	0.2287$^{\text{II,IV}}$	0.3285
ZDT2	**0.5576**$^{\text{II,III,IV}}$	0.5657$^{\text{IV}}$	0.5639$^{\text{IV}}$	0.7364
ZDT3	**0.2758**$^{\text{II,III,IV}}$	0.2787$^{\text{III,IV}}$	0.2801$^{\text{IV}}$	0.4499
ZDT4	**0.0041**$^{\text{II,III,IV}}$	0.0048$^{\text{III,IV}}$	0.0089$^{\text{IV}}$	0.0526
ZDT6	**0.2854**$^{\text{II,III,IV}}$	0.2951$^{\text{IV}}$	0.2941$^{\text{II,IV}}$	0.3125
DTLZ1	**0.0007**$^{\text{II,III,IV}}$	0.041	0.0289$^{\text{II}}$	0.0438
DTLZ2	**0.7776**$^{\text{II,III,IV}}$	0.7791$^{\text{IV}}$	0.7785$^{\text{II,IV}}$	0.7799
DTLZ3	**0.0006**$^{\text{II,III,IV}}$	0.0394	0.0345$^{\text{II,IV}}$	0.0684
DTLZ4	**0.7797**$^{\text{II,III,IV}}$	0.7812$^{\text{IV}}$	0.7806	0.7858
DTLZ5	**0.779**$^{\text{II,III,IV}}$	0.7801$^{\text{IV}}$	0.7797$^{\text{II,IV}}$	0.7814
DTLZ6	**0.0083**$^{\text{II,III,IV}}$	0.5818	0.4125$^{\text{II}}$	0.0083$^{\text{II,III}}$
DTLZ7	**0.3398**$^{\text{II,III,IV}}$	0.3442$^{\text{IV}}$	0.3428$^{\text{II,IV}}$	0.5286
WFG1	0.3476	**0.3382**$^{\text{I,III,IV}}$	0.34$^{\text{I}}$	0.3395$^{\text{I,III}}$
WFG2	0.4645	0.3746$^{\text{I,III}}$	0.3766$^{\text{I}}$	**0.3283**$^{\text{I,III}}$
WFG3	0.4828	0.4221$^{\text{I}}$	**0.4184**$^{\text{I,II,IV}}$	0.4202$^{\text{I,II}}$
WFG4	0.7228	0.6892$^{\text{I,IV}}$	**0.6873**$^{\text{I,II,IV}}$	0.7064$^{\text{I}}$
WFG5	0.645	0.6291$^{\text{I,IV}}$	**0.6287**$^{\text{I,II,IV}}$	0.6301$^{\text{I}}$
WFG6	0.7477	0.732$^{\text{I}}$	0.7313$^{\text{I}}$	**0.7014**$^{\text{I,II,III}}$
WFG7	0.6954	0.6775$^{\text{I}}$	**0.6753**$^{\text{I,II,IV}}$	0.6761$^{\text{I,II}}$
WFG8	**0.5968**$^{\text{II,III}}$	0.633	0.6415	0.5971$^{\text{II,III}}$
WFG9	0.6779	0.6343$^{\text{I,IV}}$	**0.6321**$^{\text{I,II,IV}}$	0.6362$^{\text{I}}$
Three-objective test problems				
DTLZ1	**0.08**$e-6^{\text{II,III,IV}}$	0.0026$^{\text{IV}}$	0.05$e-3^{\text{II,IV}}$	0.2962
DTLZ2	0.4592$^{\text{II,IV}}$	0.4813$^{\text{IV}}$	**0.4429**$^{\text{I,II,IV}}$	0.893
DTLZ3	**0.08**$e-5^{\text{II,III,IV}}$	0.0043$^{\text{IV}}$	0.0002$^{\text{II,IV}}$	0.5863
DTLZ4	**0.4666**$^{\text{II,III,IV}}$	0.4721$^{\text{III,IV}}$	0.6256$^{\text{IV}}$	0.8852
DTLZ5	**0.8808**$^{\text{II,III,IV}}$	0.8818$^{\text{IV}}$	0.8817$^{\text{II,IV}}$	0.9807
DTLZ6	**0.0047**$^{\text{II,III,IV}}$	0.3865	0.0438$^{\text{II,IV}}$	0.1807$^{\text{II}}$
DTLZ7	**0.3763**$^{\text{II,III,IV}}$	0.4017$^{\text{IV}}$	0.3876$^{\text{IV}}$	0.9499
WFG1	0.1143$^{\text{IV}}$	**0.0778**$^{\text{I,III,IV}}$	0.0816$^{\text{I,IV}}$	0.4749
WFG2	0.2022$^{\text{IV}}$	**0.0725**$^{\text{I,IV}}$	0.2223$^{\text{IV}}$	0.3724
WFG3	0.416$^{\text{IV}}$	0.3514$^{\text{I,IV}}$	**0.349**$^{\text{I,II,IV}}$	0.7235
WFG4	0.5309$^{\text{IV}}$	0.5104$^{\text{I,IV}}$	**0.4587**$^{\text{I,II,IV}}$	0.8674
WFG5	0.492$^{\text{IV}}$	0.4602$^{\text{I,IV}}$	**0.4263**$^{\text{I,II,IV}}$	0.8553
WFG6	0.5735$^{\text{II,IV}}$	0.6106$^{\text{IV}}$	**0.5553**$^{\text{I,II,IV}}$	0.9292
WFG7	0.4372$^{\text{IV}}$	0.3732$^{\text{I,IV}}$	**0.3376**$^{\text{I,II,IV}}$	0.814
WFG8	0.5204$^{\text{II,IV}}$	0.5579$^{\text{IV}}$	**0.5075**$^{\text{I,II,IV}}$	0.8991
WFG9	0.5372$^{\text{IV}}$	0.5214$^{\text{I,IV}}$	**0.4717**$^{\text{I,II,IV}}$	0.8794

The ϵ-indicator gives the minimum factor ϵ such that any objective vector in R is ϵ-dominated by at least one objective vector in A. Smaller values of $I^1_{\epsilon+}$ are preferable.

To define the reference set R, all non-dominated sets obtained by the solvers over all performed runs are combined. Then, all dominated solutions are removed, and the remaining points, which are non-dominated by any of the approximation sets are taken as a reference set.

4.3 Hypervolume Indicator

The hypervolume measure or S-metric was introduced by Zitzler and Thiele [20]. It can be defined as the Lebesgue measure Λ of the union of hypercuboids in the objective space,

$$S_{\mathbf{a}_{\text{ref}}}(A) = \Lambda\left(\bigcup_{\mathbf{a}\in A}\{f_1(\mathbf{a}'),\ldots,f_M(\mathbf{a}') : \mathbf{a} \prec \mathbf{a}' \prec \mathbf{a}_{\text{ref}}\}\right),$$

where A denotes the set of non-dominated solutions and $\mathbf{a}_{\text{ref}}$ is the appropriately chosen reference point. We consider the hypervolume difference to a reference set R, and this is referred as the hypervolume indicator I^-_H [11]. Given an approximation set A, the indicator value is defined as,

$$I^-_H(A) = S_{\mathbf{a}_{\text{ref}}}(A_{\text{ref}}) - S_{\mathbf{a}_{\text{ref}}}(A)$$

where smaller values correspond to higher quality in contrast to the original hypervolume $S_{\mathbf{a}_{\text{ref}}}(A)$. The reference point is determined by the point with largest values of the objective functions among all solvers over the all performed runs for

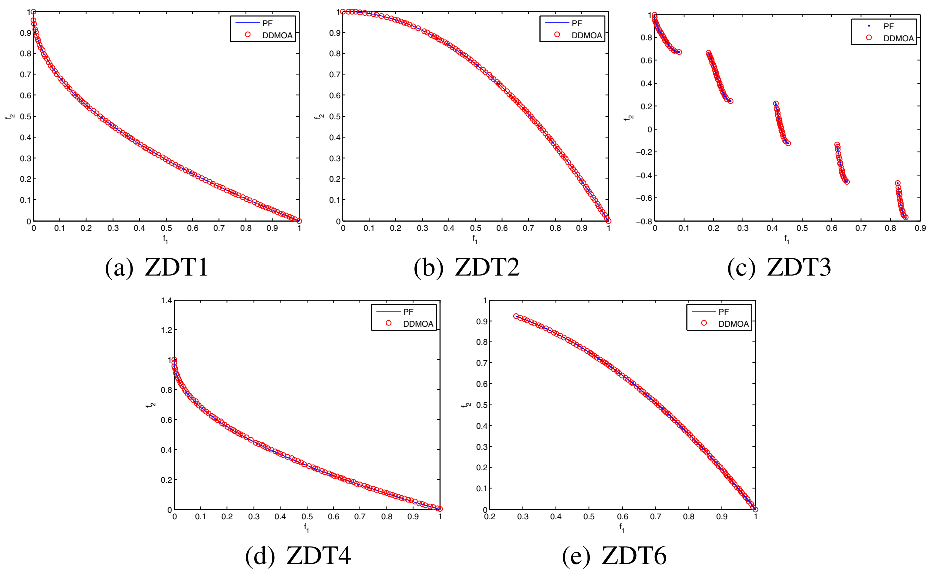


Fig. 2 Plots of the non-dominated solutions with the best hypervolume value found by DDMOA in 30 runs in the objective space on ZDT test problems

each problem. Before calculating the hypervolume, the data is normalized. Thus, the hypervolume indicator [11] can be defined as:

$$I_H^-(A) = 1 - S_{\mathbf{a}_{\text{ref}}}(A).$$

5 Experimental Results

In order to assess the performance of the proposed approach, we compare the obtained results with those obtained by NSGA–II [3], IBEA [18] and MOEA/D [12]. We have used the implementation of these algorithms provided by jMetal framework [6]. The choice of the algorithms is not occasional, because this algorithms are representative well-established state-of-the-art evolutionary algorithms and represent three different trends in evolutionary multiobjective optimization. Specifically, NSGA–II is dominance based, IBEA is indicator based and MOEA/D is recent powerful decomposition based.

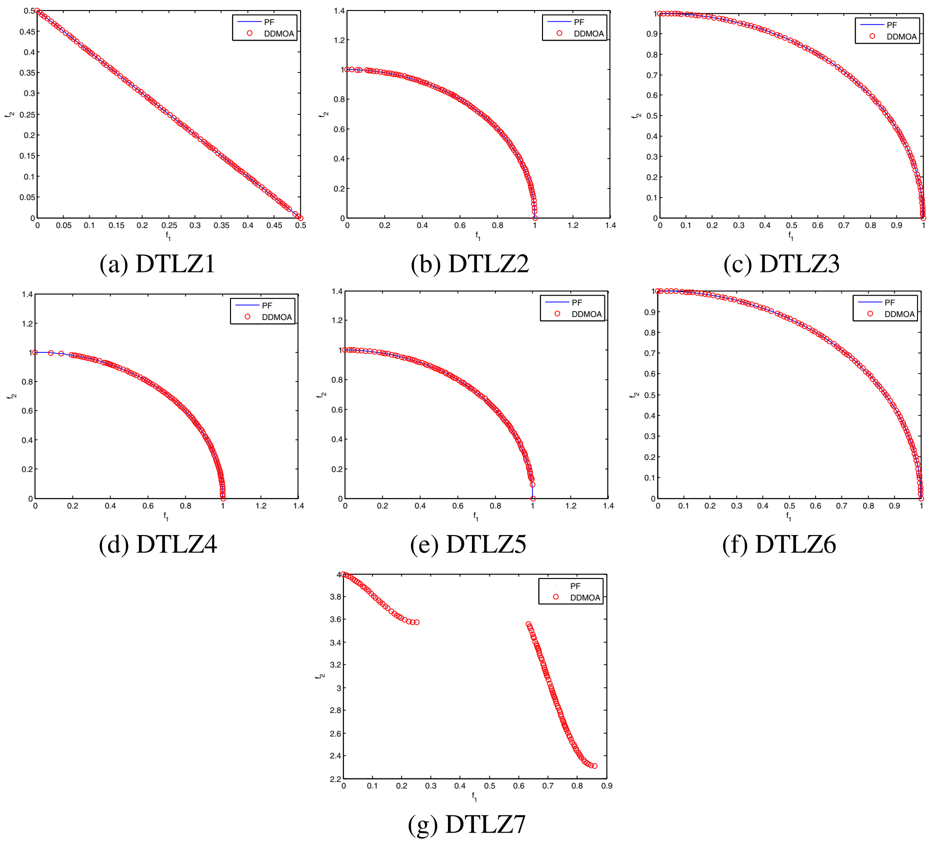
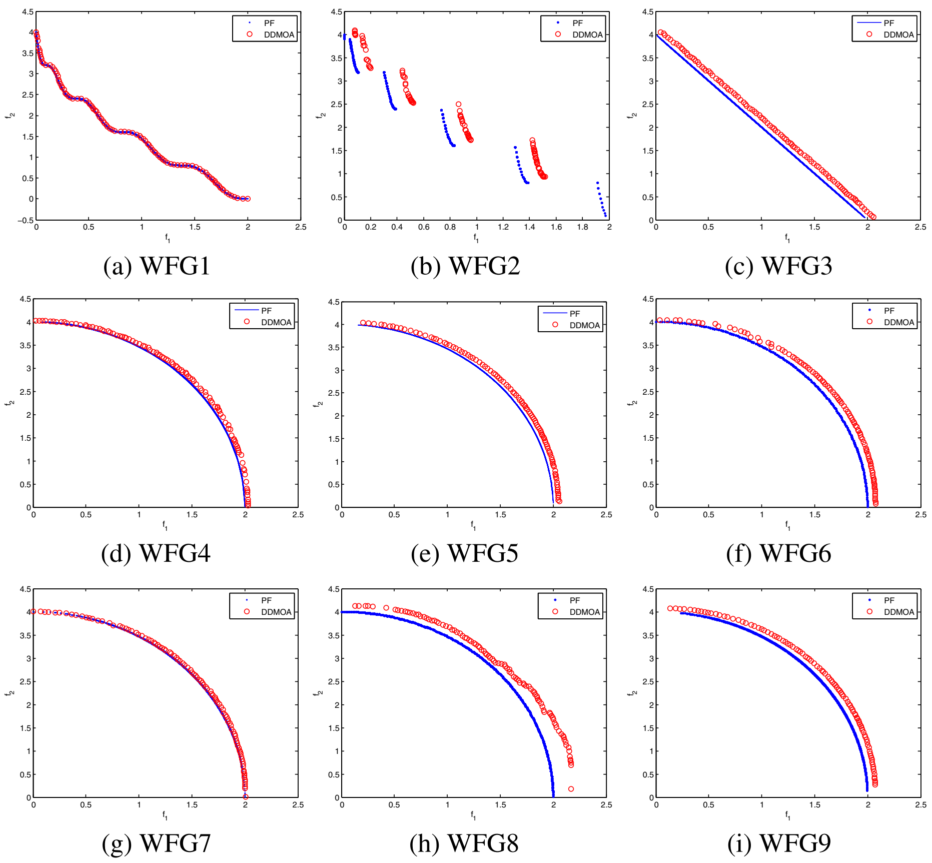


Fig. 3 Plots of the non-dominated solutions with the best hypervolume value found by DDMOA in 30 runs in the objective space on the two-objective DTLZ test problems

We adopted the two-objective ZDT [17], DTLZ [4] and WFG [8] test problems, as well as the three-objective DTLZ [4] and WFG [8] test problems. We used 30 decision variables for ZDT1, ZDT2, ZDT3, ZDT6, and ZDT4 was tested using 10 decision variables. For the two-objective and three-objective DTLZ test problems, 30 decision variables were adopted. The two-objective and three-objective WFG test problems were tested with 10 decision variables ($k = 4$ position related parameters and $l = 6$ distance related parameters).

For all algorithms, we set the population size equal to 100 and the maximum number of objective function evaluations is set to 15,000 and 20,000 for the two-objective and three-objective test problems, respectively. All other parameters for NSGA–II, IBEA and MOEA/D are the default in jMetal framework [6]. The default values for the parameters used in DDMOA are the following: initial step size and tolerance for local search $\delta_0 = 1$, $\delta_{\text{tol}} = 10^{-3}$, the number of subpopulations $\alpha = 5$, the initial step size $\sigma_0 = 5$. These parameters were obtained as a result of the performed sensitivity analysis. Since we compare stochastic algorithms, for each problem 30 independent runs of each algorithm were performed.



(a) WFG1 (b) WFG2 (c) WFG3
(d) WFG4 (e) WFG5 (f) WFG6
(g) WFG7 (h) WFG8 (i) WFG9

Fig. 4 Plots of the non-dominated solutions with the best hypervolume value found by DDMOA in 30 runs in the objective space on the two-objective WFG test problems

Table 1 presents the statistical tests and the median values of the epsilon indicator. The superscripts I, II, III and IV indicate whether the respective algorithm performs significantly better than DDMOA, NSGA–II, IBEA and MOEA/D, respectively. The best value in each row is marked bold (the lower the better). DDMOA outperforms the other algorithms on ZDT and the majority of the two and three-objective DTLZ test problems. For the two-objective WFG test problems, the best performance is achieved by MOEA/D that provides the best median values of the epsilon indicator in 5 out of 9 test problems. For the three-objective WFG test problems, IBEA has the best performance, providing the best median values of the epsilon indicator in 6 out of 9 test problems. DDMOA outerperforms on the three-objective WFG test instances.

Table 2 shows the statistical tests and the median values of the hypervolume indicator. The superscripts I, II, III and IV indicate whether the respective algorithm performs significantly better than DDMOA, NSGA–II, IBEA and MOEA/D, respectively. The best value in each row is marked bold (the lower the better). DDMOA outperforms the other algorithms on the two-objective ZDT and DTLZ test problems, providing significantly better values of the hypervolume indicator than all other considered algorithms. IBEA performs significantly better than the

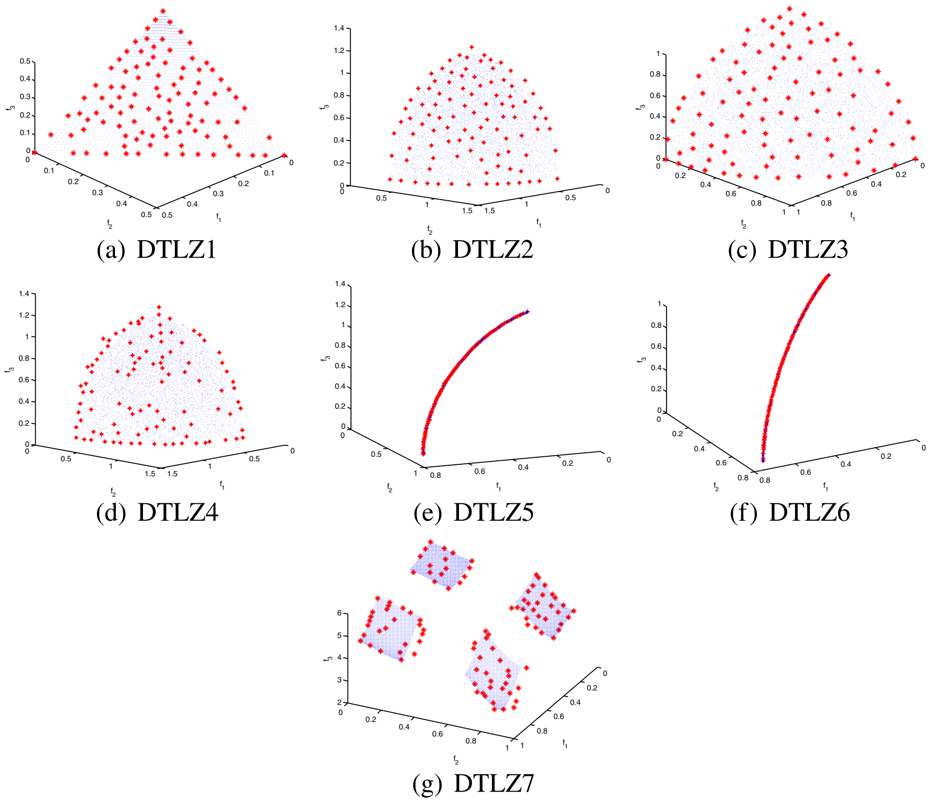


Fig. 5 Plots of the non-dominated solutions with the best hypervolume value found by DDMOA in 30 runs in the objective space on the three-objective DTLZ test problems

other algorithms on the 5 two-objective WFG test problems. For the three-objective WFG3-9 test problems, IBEA gives significantly better results in terms of the hypervolume indicator than the other algorithms. The dominance of IBEA on WFG test problems in terms of the hypervolume can be explained because it is the only one of the considered algorithms which attempts to maximize the cumulative hypervolume covered by non-dominated solutions.

Figures 2, 3, 4, 5 and 6 show the non-dominated solutions with the best hypervolume value found by DDMOA in 30 runs ploted in the objective space. Attending Figs. 2, 3 and 5, we can see that DDMOA reaches the Pareto-optimal fronts for the ZDT two-objective problem as well as to three objective DTLZ problem, proving well-distributed solutions in the objective space. From Fig. 4, we see that DDMOA performs poorly on the majority of the two-objective WFG test problems. The adequate Pareto-optimal fronts are achieved by DDMOA for the problems WFG1, 4 and 7. Although DDMOA performs poorly on WFG8 test problem, from Table 2 we can see that DDMOA provides the best median value of the hypervolume indicator, being significantly better than NSGA-II and IBEA on the same problem. Analyzing Figure 6, we can observe that DDMOA is able to find an adequate approximation to

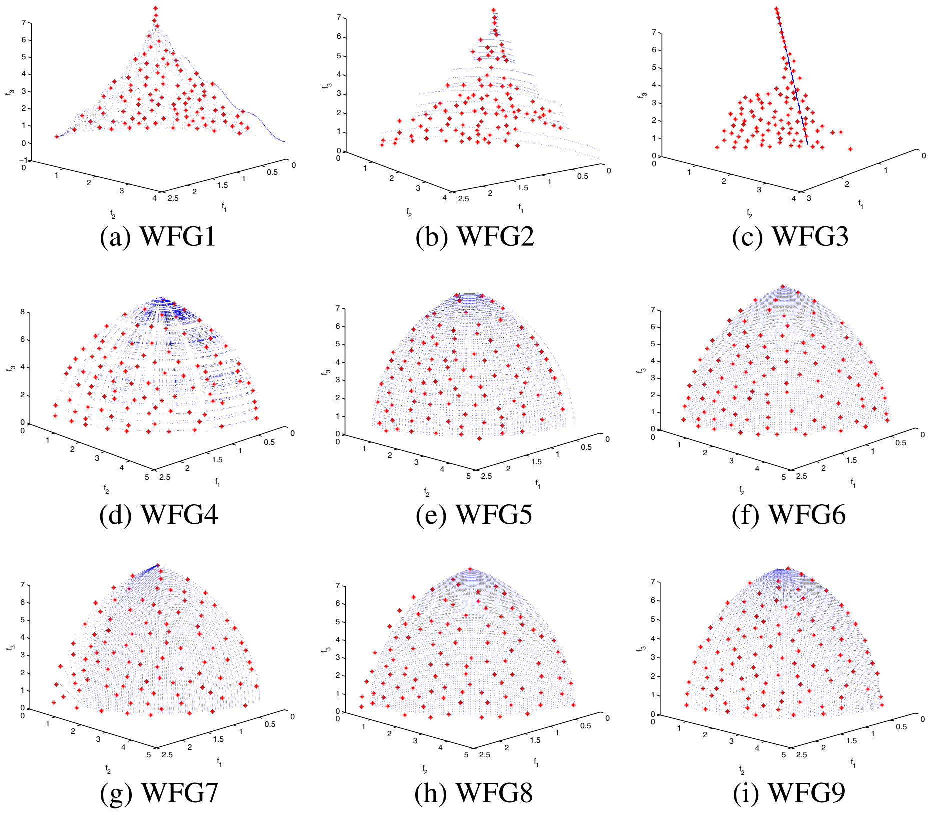
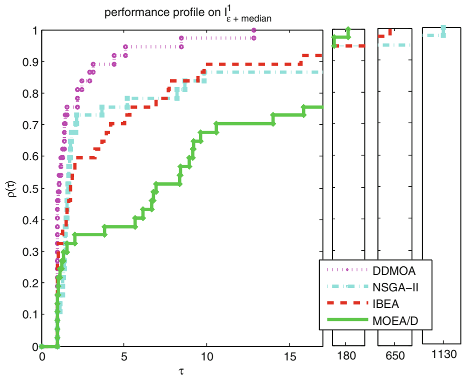
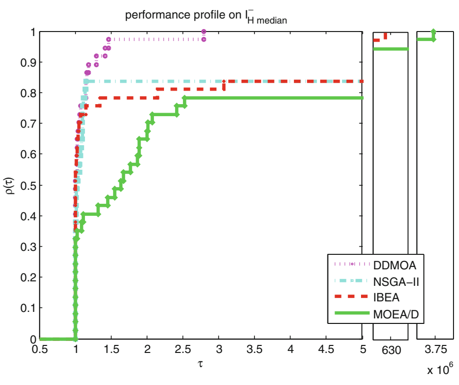


Fig. 6 Plots of the non-dominated solutions with the best hypervolume value found by DDMOA in 30 runs in the objective space on the three-objective WFG test problems



(a) Performance profile on $I^1_{\epsilon+\text{median}}$.



(b) Performance profile on $I^-_{H\ \text{median}}$.

Fig. 7 Performance profiles on the median values of the quality indicators

the Pareto-optimal fronts for all the three-objective WFG test problems except for WFG3 test problem.

Figure 7 presents performance profiles on the median values of both quality indicators, which allow to get insights about the overall performance of the algorithms on all tested problems. Figure 7a shows that the most accurate algorithm is DDMOA, providing the best median value of the epsilon indicator in 46 % of the tested problems. In terms of the robustness, the best algorithm is DDMOA. Figure 7b shows that the most accurate algorithm is DDMOA, providing the best median value of the hypervolume indicator in 49 % of the tested problems. In terms of robustness, the best algorithm is again DDMOA.

6 Conclusions

Most of existing multiobjective techniques rely on evolutionary algorithms or classical methods based on decomposition. However, combination of mathematical programming methods and evolutionary algorithms has been proven to be very successful in single-objective optimization. This work attempts to provide such combination to the mutiobjective case.

This paper propose a new hybrid evolutionary multiobjective algorithm (DDMOA). The algorithm combines a classical method for multiobjective optimization, local search and general features of evolutionary algorithms. Child solutions are generated adding to a parent solution a linear combination of descent directions of the objective functions. To find descent directions of the objectives, pattern search method is employed combined with a strategy based on subpopulations. The proposed strategy efficiently guide the population toward the Pareto-optimal front.

We have compared DDMOA with some outstanding state-of-the-art evolutionary multiobjective algorithms on a set of benchmark test problems. Our analysis has shown that DDMOA outperforms other algorithms on a large number of the tested problems, providing always competitive results. Moreover, all results are provided with statistical confidence.

As future work, we will convey further studies of the methods to update the step size used to generate offspring as well as during the local search procedure to find descent directions. We will also try to improve the convergence of the algorithm through the application of the techniques which allow to use the solutions from other non-dominated fronts to generate offspring.

Acknowledgements The authors would like to thank FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) that supported in part this work.

References

1. Costa, L., Espírito Santo, I., Oliveira, P.: Stochastic algorithms assessment using performance profiles. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 933–940 (2011)
2. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
3. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: Nsga-ii. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
4. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. Technical Report 112, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)
5. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)
6. Durillo, J.J., Nebro, A.J.: jmetal: a java framework for multi-objective optimization. Adv. Eng. Softw. **42**(10), 760–771 (2011)
7. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. J. Heuristics **15**(6), 617–644 (2009)
8. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006)
9. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans. Syst. Man Cybern., Part C Appl. Rev. **28**(3), 392–403 (1998)
10. Knowles, J., Corne, D.: M-paes: a memetic algorithm for multiobjective optimization. In: Proceedings of the Congress on Evolutionary Computation, pp. 325–332 (2000)
11. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2006)
12. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. IEEE Trans. Evol. Comput. **13**(2), 229–242 (2009)
13. Loh, W.L.: On latin hypercube sampling. Ann. Stat. **33**(6), 2058–2080 (1996)
14. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Dordrecht (1999)
15. Shukla, P.K., Deb, K.: On finding multiple pareto-optimal solutions using classical and evolutionary generating methods. Eur. J. Oper. Res. **181**(3), 1630–1652 (2007)
16. Timmel, G.: Ein stochastisches suchverrahren zur bestimmung der optimalen kompromilsungen bei statischen polzkriteriellen optimierungsaufgaben. Wiss. Z. Tech. Hochsch. Ilmenau **26**(5), 159–174 (1980)
17. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000)
18. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, pp. 832–842 (2004)
19. Zitzler, E., Laumanns, M., Thiele, L.: Spea2: improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)

20. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - a comparative case study. In: Proceedings of the 5th International Conference on Parallel Problem Solving From Nature, pp. 292–304 (1998)
21. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)