Memetic Comp. (2011) 3:271–285
DOI 10.1007/s12293-011-0072-9

REGULAR RESEARCH PAPER

DMEA: a direction-based multiobjective evolutionary algorithm

Lam T. Bui · Jing Liu · Axel Bender · Michael Barlow · Slawomir Wesolkowski · Hussein A. Abbass

Received: 10 August 2010 / Accepted: 17 October 2011 / Published online: 3 November 2011

Abstract A novel direction-based multi-objective evolutionary algorithm (DMEA) is proposed, in which a population evolves over time along some directions of improvement. We distinguish two types of direction: (1) the convergence direction between a non-dominated solution (stored in an archive) and a dominated solution from the current population; and, (2) the spread direction between two non-dominated solutions in the archive. At each generation, these directions are used to perturb the current parental population from which offspring are produced. The combined population of offspring and archived solutions forms the basis for the creation of both the next-generation archive and parental pools. The rule governing the formation of the next-generation parental pool is as follows: the first half is populated by non-dominated solutions whose spread is aided by a niching criterion applied in the decision space. The second half is filled with both non-dominated and dominated solutions from the sorted remainder of the combined population. The selection of non-dominated solutions for the next-generation archive is also assisted by a mechanism, in which neighborhoods of rays in objective space serve as niches. These rays originate from the current estimate of the Pareto optimal front's (POF's) ideal point and emit randomly into the hyperquadrant that contains the current POF estimate. Experiments on two well-known benchmark sets, namely ZDT and DTLZ have been carried out to investigate the performance and the behavior of the DMEA. We validated its performance by comparing it with four well-known existing algorithms. With respect to convergence and spread performance, DMEA turns out to be very competitive.

L. T. Bui
Department of Software Engineering,
Faculty of Information Technology Organization,
The Le Quy Don Technical University, Hanoi, Vietnam
e-mail: lam.bui07@gmail.com

J. Liu (✉) · M. Barlow · H. A. Abbass
School of Engineering and Information Technology,
The University of New South Wales at the Australian Defence Force Academy, Canberra 2600, Australia
e-mail: jing.liu@adfa.edu.au

M. Barlow
e-mail: m.barlow@adfa.edu.au

H. A. Abbass
e-mail: h.abbass@adfa.edu.au

A. Bender
Defence Science and Technology Organisation,
Edinburgh SA 5111, Adelaide, Australia
e-mail: axel.bender@dsto.defence.gov.au

S. Wesolkowski
DRDC Centre for Operational Research and Analysis, Ottawa, Canada
e-mail: s.wesolkowski@ieee.org

Keywords Multi-objective optimization problems · Evolutionary algorithms · Direction information

1 Introduction

Evolutionary algorithms (EAs) are well accepted tools for solving multi-objective optimization problems (MOPs) [4,8,11,29] and a rich variety of real applications ([10,15,21], among other). Since the first EA for MOPs was introduced in 1985 [24], there has been a plethora of proposals for supplementing multi-objective EAs (MOEAs) with various advanced techniques such as elitism, use of local information, and decomposition. A thorough survey of MOEAs has been provided in [7].

Among those techniques, the use of elitism is the most popular one. Usually, a particular archive is (either implicitly or explicitly) maintained over time. The non-dominated solutions are stored in this archive and later used to mix with the newly derived offspring to form the next generation of solutions. In our opinion, this archive is underutilized; it is an information-rich data set and can contribute to the evolutionary process in more ways than just being the pool of parents for future offspring.

We propose that given a good archive maintenance scheme, some directions of improvement can be derived from archival information and used to guide the evolutionary process. Finding effective and efficient ways of steering the optimization process is most desirable in the design of an optimization algorithm. The steepest gradient descent method [27] is the best-known example of a simple search scheme that derives direction from information present in the local neighborhood of iterated solutions. In general thought, defining a good direction is not a trivial task, especially when the optimization process involves non-linear or black-box functions. Good direction-guided algorithms need to strike a delicate balance between exploitation and exploration. On the one hand, an algorithm is required to move quickly toward a potential optimum. On the other hand, it should not get trapped too easily in a basin surrounding a local optimum. This could prematurely end its search for the domain in which the global optimum can be found.

In evolutionary computation (EC), direction-guided search has proved to be quite promising. Differential evolution (DE) utilizes the direction between two parents to guide the generation of offspring [22]. It has been very effective in solving continuous optimization problems, both single- and multi-objective [1,2]. Lara et al. [19] analyzed the main issues when introducing gradient information into a MOEA from both, the local search and the global search point of view. Recently, we also demonstrated the benefit of direction-guided search in a framework for distributed MOEAs [5].

In this paper, we describe a new algorithm—the direction-based multi-objective evolutionary algorithm (DMEA). In DMEA, an external archive of non-dominated solutions is maintained over time. DMEA is based on the effective use and refinement of information contained in the archive. The archive not only contribute solutions to the next generation, but also supports the derivation of directions for offspring production. Offspring are generated in a process in which randomly-selected parents are perturbed along identified directions of improvement. Two types of direction are used: (1) a convergence direction between an archived non-dominated solution and a dominated solution from the parental pool; and, (2) a spread direction between two non-dominated solutions in the archive. The pool of offspring is combined with the archive in order to generate the next-generation archival and parental pools. By being updated with non-dominated as well as dominated solutions, the next-generation parental pool allows for both exploitation and exploration to occur throughout the evolutionary process.

Niching—a mechanism of enticing a population to breed uniformly and thereby spawning diversity—also plays an important role in DMEA design [25,26]. Ideally, non-dominated solutions of the MOP are homogeneously distributed on the Pareto optimal front (POF) in objective space. In order to facilitate this even spread, we use a bundle of rays that emits uniformly from the POF's estimated ideal point into the (hyper) quadrant that contains the POF estimate. During the archival update, solutions' proximity to the ray ensemble is used as a selection criterion that supplements non-dominance. In our algorithm we set the number of rays equal to archive size. To generate the next-generation parent population, the non-dominated solutions are kept some distance apart from each other according to niching information computed in the decision space.

To validate the newly proposed DMEA, we carried experiments on 12 problems from two well-known benchmark sets. The results strongly suggest that DMEA performs well in both convergence and solution spread. The results indicate that DMEA is very competitive.

The paper is organized in six sections. Section 2 introduces common notations in MOP. A brief summary of MOEAs is given in Sect. 3 and followed by the description of DMEA in Sect. 4. The experimental results on 12 benchmark problems is presented in Sect. 5 to examine the effectiveness and efficiency of DMEA. Conclusion and future work are given in Sect. 6.

2 Common concepts

Real-life problems are typically characterized by multiple competing objectives (or criteria). Their solutions therefore describe alternatives, each of which represents a different compromise between the conflicting objectives. A subset of these alternatives contains all *Pareto optimal* solutions. In the case of multi-objective minimization problems, a "solution to a MOP is Pareto optimal if there exists no other feasible solution which would decrease some criterion without causing a simultaneous increase in at least one other criterion" [7]. The set of solutions that satisfies the Pareto optimality definition is called the *Pareto optimal set* (POS). Its projection in objective space is known as the *Pareto optimal front*. The *ideal point* of the POF is the vector whose components contain the result of minimizing each objective individually.

Mathematically, in a k-objective unconstrained (bound constrained) minimization problem, a vector function $\mathbf{f}(\mathbf{x})$ of k objectives is defined as:

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \quad (1)$$

in which $\mathbf{x}$ is a vector of decision variables in n-dimensional $\mathbb{R}^n$. In EC, $\mathbf{x}$ represents an individual in the population to be evolved. The value $f_j(\mathbf{x})$, then, describes the performance of individual $\mathbf{x}$ as evaluated against the jth objective in the MOP.

An individual $\mathbf{x}_1$ is said to *dominate* $\mathbf{x}_2$ if $\mathbf{x}_1$ is not worse than $\mathbf{x}_2$ on all k objectives and is better than $\mathbf{x}_2$ on at least one objective. If $\mathbf{x}_1$ does not dominate $\mathbf{x}_2$ and $\mathbf{x}_2$ also does not dominate $\mathbf{x}_1$, then $\mathbf{x}_1$ and $\mathbf{x}_2$ are said to be *non-dominated* with respect to each other. If we use the symbol "$\preceq$" to denote that $\mathbf{x}_1 \preceq \mathbf{x}_2$ means $\mathbf{x}_1$ dominates $\mathbf{x}_2$, and the symbol "$\ntriangleright$" between two scalars a and b to indicate that $a \ntriangleright b$ means a is not worse than b, then *dominance* can be formally defined as [11]:

Definition 1 (*Dominance*) $\mathbf{x}_1 \preceq \mathbf{x}_2$ if the following conditions are held:

1. $f_j(\mathbf{x}_1) \ntriangleright f_j(\mathbf{x}_2) \forall j \in \{1, 2, \ldots, k\}$; and,
2. $\exists j \in \{1, 2, \ldots, k\} : f_j(\mathbf{x}_1) \lhd f_j(\mathbf{x}_2)$.

In general, if an individual is not dominated by any other individual in the population, it is called a non-dominated solution. All non-dominated solutions in a population form the non-dominated set as formally described in the following definition:

Definition 2 (*Non-dominated set*) A set S is said to be the non-dominated set of a population P if the following conditions are met:

1. $S \subseteq P$; and,
2. $\forall \mathbf{s} \in S \nexists \mathbf{x} \in P : \mathbf{x} \preceq \mathbf{s}$.

If P represents the entire search space, then S is referred to as the *global Pareto optimal set*. If P represents only a subspace, then S is called the *local Pareto optimal set*. While there can be multiple local POSs, there exists only one global one.

3 Related work

Elitist is a mechanism to preserve the best individuals, once found, during the optimization process. The concept of elitist was established at an early stage of EC (see, for example [14]); and to date, it has been widely used in EAs. Elitist can be realized either by placing one or more of the best parents directly into the next generation of individuals, or by replacing only those parents that are dominated by their offspring [28].

Elitist MOEAs usually (but not necessarily) employ an external set called the archive to store the non-dominated solutions after each generation. In general, when using an archive, there are two important aspects to consider [7]:

– *Interaction between archive and main population* during the optimization process the archive can be combined with the current population to form the population for the next generation as in [35]. However, the archive is more than just a gene pool. It also contains information about the best performance of the algorithm so far. Exploiting this rich archival information should enhance the optimization process and is the main motivation for the research reported in this paper.
– *Updating the archive* the method by which the archive is built also plays an important role. In some approaches the neighborhood relationship between individuals is used; e.g. in the form of geographical grid [17], crowded dominance [13], and clustering [35]. Others entail controlling the size of the archive through truncation when the number of non-dominated individuals exceeds a predefined threshold. In this paper we will pursue a different approach to maintaining the archive. Details will be given in the next section.

How archive and main population interact and how the archive is being updated differ from one MOEA to another. The general elitist principle is to preserve each generation's best individuals. This helps algorithms to get closer to the POF. A proof of convergence for MOEAs using elitist can be found in [23]. Algorithms such as Pareto archived evolution strategy (PAES) [17], strength Pareto EA 2 (SPEA2) [35], Pareto frontier DE (PDE) [2], NSGA-II [13], and multi-objective particle swarm optimization (MOPSO) [9] are typical examples of elitist MOEAs.

Memetic algorithms, which are population based meta-heuristics which combine local search components within an evolutionary framework, have also been applied to multi-objective optimization recently, and several multi-objective memetic algorithms (MOMAs) have therefore been designed [6,16,20,30]. In DMEA, since we perturb the solution according the information from archive before performing evolutionary operators, DMEA can be viewed as using a shallow-depth local search strategy. However, unlike most existing MOMAs, our local search is integrated into the evolutionary process.

4 Direction-based multi-objective evolutionary algorithm

4.1 Overview

To paraphrase the previous section, elitist is a very useful mechanism to enhance MOEAs. We will therefore adopt this mechanism in our methodology. We will especially address both issues mentioned above: *interaction between archive and main population* and *archive update*.

The novel algorithm we propose is an elitist MOEA; i.e. throughout the evolution of MOP solutions, an external archive is being maintained. This archive not only stores elitist solutions but also contributes directional information to guide the evolutionary process. Knowing how solutions have improved from one iteration to the next is useful information in any iterative optimization approach. We propose to make use of this information during the MOEA reproduction phase. At every generation, the archive is exploited to determine directions of improvement. The main population is then perturbed along those directions in order to produce offspring. Subsequently, the offspring are merged with the current archive to form a temporary population, called *combined population*, from which the next generation's archive and parental pool are derived. Based on this principle, we call our algorithm *direction-based multi-objective evolutionary algorithm* (*DMEA*).

The second unique feature of DMEA entails the deterministic control of some aspects of the selection of non-dominated solutions for archive and main population. Augmenting MOEA with deterministic steps is not uncommon; NSGA-II is a well-known example for this. However, unlike NSGA-II in which solutions are sorted into different layers, in DMEA we place solutions into two categories only: non-dominated and dominated solutions. The archive is updated by using niching in objective space, while up to half of the next-generation main population is filled by applying niching criteria in decision variable space.

The details of our methodology are described in the following subsections.

4.2 Directional information

We propose two types of directional information to perturb the parental population prior to offspring production: convergence and spread.

Convergence direction (CD). In general defined as the direction from a solution to a better one, CD in MOP is a normalized vector that points from dominated to non-dominated solutions. If non-dominated solutions are maintained globally, CD corresponds to the global direction of convergence. In unconstrained MOP, a dominated solution guided by this direction is more likely to find a better area in the decision space than an unguided solution. An example is given in Fig. 1 where *A* is a dominated solution and *BCDEFG* are non-dominated solutions. All the directions from *A* to *BCDEFG* are considered convergence directions.

Given a *dominated* parent Par_d with decision space coordinates $Par_d(i),\ i \in \{1, 2, \ldots, n\}$, a perturbation rate $0 < p < 1$, and a normalized CD d_1, then, prior to offspring production in DMEA, we perturb Par_d to form a vector S_1 with coordinates

$$S_1(i) \equiv Par_d(i) + RND_i(p)\sigma_1 d_1(i). \tag{2}$$

Here σ_1 is a scalar model parameter, which, in the case study of Sect. 5, is uniformly sampled from (0, 2) at random. The random choice of σ_1 ensures that the step length varies to avoid the bias inherent in a fixed step length. $RND_i(p)$ equals 1 if $U(0,1) < p$ and 0 if $U(0,1) \geq p$, where $U(0,1)$ is random real number uniformly sampled from (0, 1). Note that because of this random sampling, not necessarily all components of Par_d will receive contributions from d_1. Actually, the higher the decision space's dimension, the less likely it is that Par_d is perturbed along the exact CD. This follows the practice of how directional information is used in DE.

Spread direction (SD). Generally defined as the direction between two equivalent solutions, SD in MOP is a normalized vector that points from one non-dominated solution to another. If solutions are perturbed along the SD, a better spread within the population should be obtained. In Fig. 1, we show an example of spread directions between non-dominated solutions *BCDEFG*.

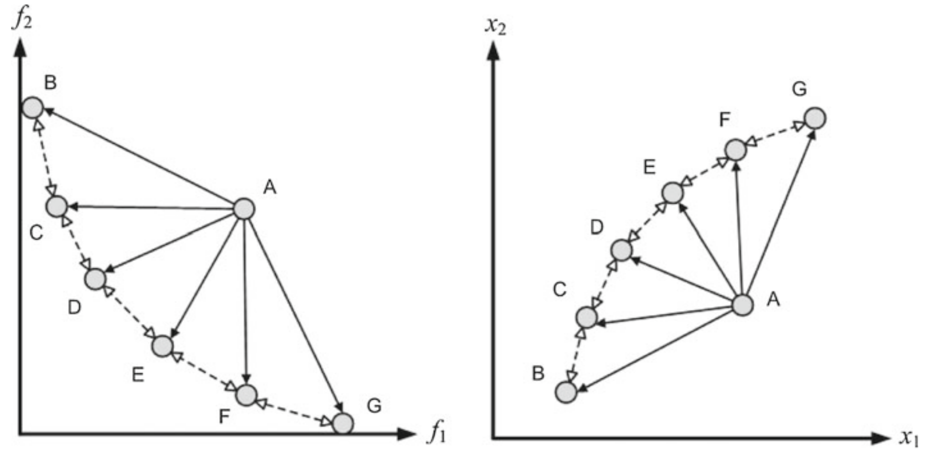


Fig. 1 Illustration of convergence (*black arrows*) and spread (*hollow arrows*) directions in objective space (*left*) and decision variable space (*right*)

Given a *non-dominated* parent Par_n and an SD d_2, the components of its spread perturbation S_2 are defined as:

$$S_2(i) \equiv Par_n(i) + RND_i(p)\sigma_2 d_2(i). \tag{3}$$

As with S_1, S_2 is not perturbed along the exact SD but only along some of its components.

There are many ways of defining directions for solution sets. In DMEA, we settle for a simple but computationally efficient method, when we perturb a parent:

$$d_1 \equiv \frac{A_1 - Par_d}{|A_1 - Par_d|} \tag{4}$$

$$d_2 \equiv \frac{A_2 - A_3}{|A_2 - A_3|} \tag{5}$$

where A_1, A_2 and A_3 are three solutions randomly selected from the archive. In combination with offspring production, niching and evolutionary selection, this crude method of calculating directions of improvement proves to be very effective as the case study in Sect. 5 will illustrate.

4.3 Niching information

A characteristic of solution quality in MOP is the even spread of non-dominated solutions across the POF [11]. In order to entice solutions to distribute homogeneously, DMEA makes use of a bundle of rays that emits randomly from the estimated ideal point into the part of objective space that contains the POF estimate (see Fig. 2). The number of rays equals the number of non-dominated solutions wanted by the user, i.e. it is equal to the size of the archive used to report the POF estimate. Rays emit into a "hyperquadrant" of objective space, i.e. the subspace that is bounded by the k hyperplanes $f_i = f_{i,\min}$, $i \in \{1, 2, \dots, k\}$ and described by $f_i \geq f_{i,\min} \forall i \in \{1, 2, \dots, k\}$ where $f_{i,\min} \approx \min_{\text{all} A_1, A_2, \dots} f_i$ with $A_1, A_2, \dots$ being the solutions stored in the current archive. By construction, this hyperquadrant contains the estimated POF.

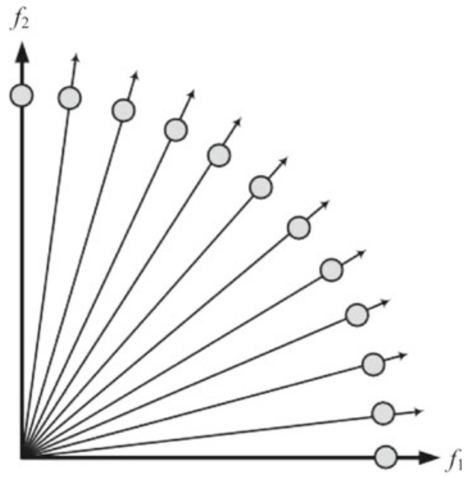


Fig. 2 Illustration of a ray bundle used for ray-assisted niching in a 2-dim MOP. The origin of the bundle is collocated with the estimated ideal point. The ray bundle is bounded by the two lines $f_1 = f_{1,\min}$ and $f_2 = f_{2,\min}$ and it emits randomly into the *top right* quadrant which contains the POF estimate

Once the archive fills with non-dominated solutions, the bundle's origin is located to the estimated ideal point. During the archival update, the rays are used as reference lines to select particular non-dominated solutions from the combined population. One by one, the rays are scanned and the non-dominated solution closest to a given ray is archived (and, for the purpose of archival update, removed virtually from the combined population). Because the number of rays equals archive size, DMEA automatically guarantees the limit of the archive to be its minimum size. There is no need for explicit truncation of solutions as seen in many other approaches such as NSGA-II and SPEA2.

Further, the ray system can be easily managed via maintaining a starting point and a set of points that are evenly spaced in objective space (a ray is equal to a pair of the starting point and a point in the set). For the sake of simplicity, this set of points will be located on a hyper-sphere with radius of 1 unit. The simplest way to generate a set of points that are evenly located in objective space, is to consider this process as an optimization problem where we need to find n points in the space (located on the unit sphere) that are kept distant from each other. The criteria for this problem is to maximize the minimal distance between points (d^*). Our finding is that for 2-objective problems, the obtained distance d^* is about $1/n$, while for 3-objective ones it is $\approx 10/n$.

Niching is also applied to the main population. From the second generation onward, the population is divided into two equal parts: one part for convergence, and one part for diversity. The first part is filled by non-dominated solutions up to a maximum of $n/2$ solutions from the combined population, where n is the population size. This filling task is based on niching information in the decision space.

Assume the number of non-dominated solutions in the combined population is $m \leq M \leq 2n$ where M is the size of the combined population. Then there are two possibilities:

- $m < n/2$: all m solutions are copied to the main population.
- $m \geq n/2$: each non-dominated solution is assigned a niching value calculated as the average Euclidean distance from all other non-dominated solutions. The solutions are then sorted according to these values and the first $[n/2]$ solutions are copied to the main population.

The second part of the main population (size $n - \min\{m, n/2\}$) is sampled from the remainder of the combined population. In the version of DMEA described in this paper, we sort the remaining solutions x_i based on the normalized weighted-sum objective value $F(x_i)$. For the sake of simplicity, the same weight is used for all objectives. We pick the $n - \min\{m, n/2\}$ solutions with the highest scores F and copy them into the main population. This *the weighted-sum scheme* allows dominated and (according to the niching value) low ranked non-dominated solutions to enter the next generation. Its purpose is to strike a balance between exploitation and exploration.

4.4 General structure of algorithm

The step-wise structure of the proposed algorithm is as follows:

- **Step 1.** Initialize the main population P with size n.
- **Step 2.** Evaluate the population P.
- **Step 3.** Copy non-dominated solutions to the archive A. (A's size is set equal to that of the main population P. Note that, initially, the archive might not be filled entirely.)
- **Step 4.** Generate an interim mixed population M of the same size n as P
  - Loop {
    - Select a random parent Par without replacement from P.
    - If Par is dominated, $j = 1$. Else $j = 2$.
    - Calculate S_j as of Eqs. 2 to 5.
    - Add S_j to M.
  - } Until (the mixed population is full).
- **Step 5.** Perform the polynomial mutation operator [11] on the mixed population M with a small rate.
- **Step 6.** Evaluate the mixed population M.
- **Step 7.** Combine the interim mixed population M with the current archive A to form a combined population C (i.e. $M + A \rightarrow C$).
- **Step 8.** Identify the estimated ideal point of the non-dominated solutions in C and determine a list of n rays R (starting from the ideal point and emitting uniformly into the hyperquadrant that contains the non-dominated solutions of C see Sect. 4.3).
- **Step 9:** Create new members of the archive A by copying non-dominated solutions from the combined population C
  - Make a copy C' of the combined population C.
  - Loop{
    - Select (without replacement) a ray $R(i)$.
    - In C', find the non-dominated solution whose distance to $R(i)$ is minimum.
    - Select (without replacement) this solution and copy it to the archive.
  - } Until (all n rays are scanned)
- **Step 10:** Determine the new population P for the next generation.
  - Empty P.
  - Determine the number m of non-dominated solutions in C.
    - If $m < n/2$, select (without replacement) all non-dominated solutions from C and copy to P.
    - Else,
      - · Determine niching value (average Euclidean distance to other non-dominated solutions in decision space) for all non-dominated solutions in C.
      - · Sort non-dominated solutions in C according to niching values.
      - · Copy (without replacement) the $n/2$ solutions with highest niching value to P.
  - Apply *weighted-sum scheme* to copy $\max\{n-m, n/2\}$ solutions to P, see end of Sect. 4.3.
- **Step 11**: Go to Step 4 if stopping criterion is not satisfied.

Except for σ_1 in (2), DMEA does not introduce any non-standard control parameters. As in other MOEA, population size, mutation rate and perturbation rate (which corresponds to crossover rate in conventional genetic algorithms) need to be specified as global parameters.

To further show the effect of the two directions we designed, we give an example on a benchmark problem, DTLZ3. In this example, the results of three versions of the above algorithm are compared. The first version is the same with the above algorithm, the second version is the same with the first version except the convergence direction does not used, and the third version is the same with the first version except the spread direction does not used. The parameters are the same with those used in the next section. The obtained POFs are shown in Fig. 3. As can be seen, the POF obtained by the first version is much better than those obtained by the two other versions. Especially, the results of the second version did not converge at all.

4.5 Computational complexity

The main computational cost comes from the task of filling the archive and the main population for the next generation. Filling the archive requires finding the closest solution to each ray from the combined population C (size $\overline{C}$). Since the number of rays is equal to the population size, the complexity of this task will be $O(\overline{C}n)$. Since $\overline{C} \leq 2n$, the complexity is $O(n^2)$. Filling solutions for the main population for the next generation requires sorting the non-dominated solutions in the combined population according to niching information in the decision space, and the sorting of the remaining solutions (after taking m solutions for the first part of the population).

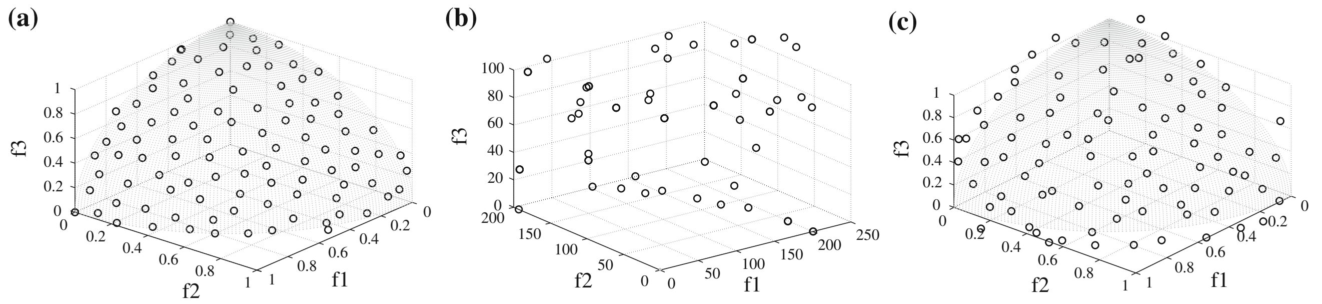


Fig. 3 The obtained POFs for DTLZ3 of the algorithm with **a** both the spread and convergence direction, **b** without the convergence direction, and **c** without the spread direction

So the complexity of this task should be on the order of $O(nlogn) + O(nlogn)$ or simply $O(nlogn)$. Therefore, the overall complexity of the algorithm is $O(n^2)$.

5 Experiments

5.1 Testing problems

This paper considers a set of 12 continuous benchmark problems that come from two well-known benchmark sets, namely ZDT [33] and DTLZ [12]. For these problems, the number of variables are between 7 and 30 while the number of objectives are 2 or 3. The reason for us to select these benchmarks is that each benchmark illustrates a different class of problems such as convexity/non-convexity, uniformity/non-uniformity, single-modality/multi-modality, linearity/non-linearity, interdependency, and continuity/discontinuity. The parameters for these problems are reported in Table 1.

Table 1 List of test problems and parameters used for experiments

Problems	Number of variables	Number of objectives	Decision space
ZDT1	30	2	$[0, 1]^{30}$
ZDT2	30	2	$[0, 1]^{30}$
ZDT3	30	2	$[0, 1]^{30}$
ZDT4	10	2	$[0, 1] \times [-5, 5]^9$
ZDT6	10	2	$[0, 1]^{10}$
DTLZ1	7	3	$[0, 1]^7$
DTLZ2	12	3	$[0, 1]^{12}$
DTLZ3	12	3	$[0, 1]^{12}$
DTLZ4	12	3	$[0, 1]^{12}$
DTLZ5	12	3	$[0, 1]^{12}$
DTLZ6	12	3	$[0, 1]^{12}$
DTLZ7	22	3	$[0, 1]^{22}$

5.2 Performance measurement methods

Performance metrics are usually used to compare algorithms in order to form an understanding of which algorithm is better and in what aspects. However, it is hard to define a concise definition of algorithmic performance. In general, when doing comparisons, a number of criteria are employed [34]. We will look at four popular criteria. The first measure is the generation distance, GD, which is the average distance from a set of solutions, denoted *P*, found by evolution to the global POS [31]. The first-norm equation is defined as

$$\mathrm{GD} = \frac{\sum_{i=1}^{n} d_i}{n} \tag{6}$$

where d_i is the Euclidean distance (in objective space) from solution i to the nearest solution in the POS, and n is the size of P. This measure is considered for convergence aspect of performance. Therefore, it could happen that the set of solutions is very close to the POS, but it does not cover the entire the POS.

The second measure will be the inverse generational distance (IGD). This measure takes into account both convergence and spread to all parts of the POS. The first-norm equation for IGD is as follows

$$\mathrm{IGD} = \frac{\sum_{i=1}^{\overline{N}} \overline{d_i}}{\overline{N}} \tag{7}$$

where $\overline{d_i}$ is the Euclidean distance (in objective space) from solution i in the POS to the nearest solution in P, and $\overline{N}$ is the size of the POS. In order to get a good value for IGD (ideally zero), P needs to cover all parts of the POS. However, this method only focuses on the solution that is closest to the solution in the POS indicating that a solution in P might not take part in this calculation. As the example see Fig. 4 where two algorithms found two sets of non-dominated solutions (black). Both sets of solutions will give the same IGD values. However, the first set, A, should not be considered as good as the second one, B. Therefore, we should use both GD and IGD to assess the performance of MOEAs.

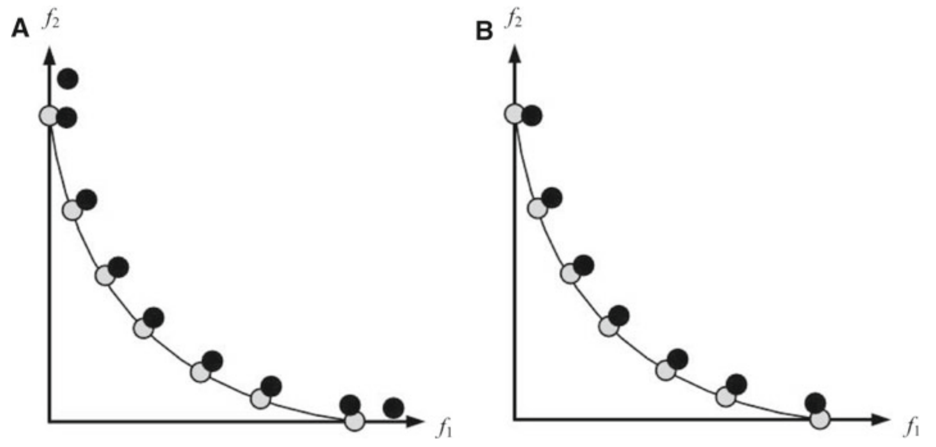


Fig. 4 A demonstration of the case where IGD cannot differentiate

The third measure is hypervolume indicator (HYP) [32], which is also named as S Metric. Being different from GD and IGD, HYP is a unary measure. Both GD and IGD use the POS as a reference, which is not practical for real-world applications. Thus, HYP attracts increasing attentions recently. HYP is a measure of the hypervolume in objective space that is dominated by a set of non-dominated points. In the following experiments, before computing HYP, the values of all objectives are normalized to the range of [1, 2], and 2.0 is taken as the reference point, which is the same with the method in PISA [3].

Since we design a spread direction in DMEA, we use M_2^* [34] as the fourth measure to evaluate the distribution of the obtained solutions, which is defined as follows,

$$M_2^* = \frac{1}{n-1} \sum_{p \in P} |\{q \in P \,|\, \|p - q\|^* > \delta^*\}| \tag{8}$$

This measure gives a value within the interval $[0, n]$ that reflects the number of δ^*-niches in P. Obviously, the higher the value is, the better the distribution for an appropriate neighbourhood parameter. For examples, when it equals to n, it means that for each objective vector there is no other objective vector within δ^*-distance to it. The value of δ^* is set to 0.05 in the following experiments.

5.3 Experimental setup

The experiments for DMEA were carried out with the following parameters: the population size was set to 100, the number of generations was fixed at 1,000, the mutation rate was kept at the same small rate of 0.01, and the perturbation rate was a relatively small 0.4.

We also selected four well-known existing algorithms for comparison purposes. For these algorithms, the population size was also set to 100 and the number of function evaluations is set to 100,000, which is the same as that of DMEA. We used the best settings we found for each algorithm as follows:

- Non-dominated sorting Genetic Algorithm version 2 (**NSGA-II**) [13]: the crossover rate, mutation rate and controlling parameters (η_m and η_c) were set to 0.9, 0.01, 20, and 15, respectively.
- Non-dominated sorting DE (**NSGA2DE**) [18]: the crossover rate was set to 0.7 while the step length uses random values uniformly sampled from (0, 1).
- Strength Pareto Evolutionary Algorithm version 2 (**SPEA2**) [35]: the crossover rate, mutation rate and controlling parameters (η_m and η_c) were set to 0.9, 0.01, 20, and 15, respectively.
- Multiobjective particle swarm optimization (**MOPSO**) [9]: the mutation rate is 0.5 and the number of divisions for the adaptive grid is set to 30.

For both DMEA and the four existing algorithms, we repeated the experiments 30 times with different random seeds for testing each problem.

5.4 Experimental results and comparison with existing algorithms

The full experimental results of DMEA are given in Table 2. The average and standard deviation (StdDev) out of 30 runs on each problem were reported. As can be seen, DMEA performed consistently over 30 different runs and obtained small standard deviations for all test problems.

The comparison on the average GD, IGD, HYP, and M_2^* out of 30 runs between DMEA and four existing algorithms were given in Tables 3, 4, 5, and 6, respectively. For each problem, we ranked the algorithms based on their performance, where a rank of 1 represents the best algorithm out of the five tested and a rank of 5 represents the algorithm

Table 2 The experimental results of DMEA

Problems	GD		IGD		HYP		M_2^*	
	Ave	SD	Ave	SD	Ave	SD	Ave	SD
ZDT1	0.0003	0.0000	0.0051	0.0003	0.7747	0.0003	99.9643	0.0086
ZDT2	0.0003	0.0000	0.0042	0.0001	0.6471	0.0000	99.9616	0.0061
ZDT3	0.0004	0.0000	0.0108	0.0010	0.6725	0.0004	99.0498	0.0915
ZDT4	0.0005	0.0003	0.0049	0.0002	0.9896	0.0000	99.9805	0.0099
ZDT6	0.0003	0.0000	0.0035	0.0000	0.4042	0.0001	99.9596	0.0000
DTLZ1	0.0025	0.0012	0.0218	0.0008	0.9994	0.0000	99.9811	0.0236
DTLZ2	0.0052	0.0003	0.0527	0.0008	0.4724	0.0025	99.9582	0.0248
DTLZ3	0.2248	0.5730	0.0872	0.1731	0.9999	0.0000	99.9623	0.0374
DTLZ4	0.0056	0.0009	0.0525	0.0010	0.6340	0.0016	99.9859	0.0131
DTLZ5	0.0005	0.0003	0.0096	0.0008	0.0926	0.0005	99.1805	0.1006
DTLZ6	0.0000	0.0000	0.0095	0.0007	0.7598	0.0001	99.2269	0.1379
DTLZ7	0.0118	0.0030	0.1506	0.0184	0.5041	0.0118	99.5145	0.1223

Table 3 The rank, mean, and Wilcoxon test on GD

Problems	NSGA-II	NSGA2DE	SPEA2	MOPSO	DMEA
ZDT1	2 (0.0006)(1)	4 (0.0037)(1)	2 (0.0006)(1)	5 (0.0493)(1)	1 (0.0003)
ZDT2	2 (0.0004)(1)	5 (0.1239)(1)	2 (0.0004)(1)	4 (0.0477)(1)	1 (0.0003)
ZDT3	2 (0.0005)(1)	4 (0.0045)(1)	2 (0.0005)(1)	5 (0.0523)(1)	1 (0.0004)
ZDT4	2 (0.0009)(1)	4 (1.0993)(1)	3 (0.0020)(1)	5 (9.9910)(1)	1 (0.0005)
ZDT6	4 (0.0009)(1)	1 (0.0003)(0)	5 (0.0033)(1)	1 (0.0003)(0)	1 (0.0003)
DTLZ1	2 (0.0912)(1)	5 (0.8997)(1)	3 (0.1726)(1)	4 (0.7585)(1)	1 (0.0025)
DTLZ2	4 (0.0057)(1)	1 (0.0052)(0)	3 (0.0056)(1)	5 (0.0074)(1)	1 (0.0052)
DTLZ3	2 (0.2881)(0)	5 (13.5661)(1)	3 (2.0648)(1)	4 (7.0195)(1)	1 (0.2248)
DTLZ4	3 (0.0055)(0)	2 (0.0051)(1)	1 (0.0045)(1)	5 (0.0059)(0)	4 (0.0056)
DTLZ5	3 (0.0001)(1)	1 (0.0000)(1)	3 (0.0001)(1)	1 (0.0000)(1)	5 (0.0005)
DTLZ6	3 (0.0941)(1)	1 (0.0000)(1)	5 (0.2813)(1)	4 (0.1038)(1)	1 (0.0000)
DTLZ7	1 (0.0111)(0)	4 (0.0154)(1)	2 (0.0118)(1)	5 (0.0555)(1)	2 (0.0118)
Average rank	2.5	3.08	2.83	4.00	**1.67**

Format of the data for DMEA: rank (mean); format of the data for the four other algorithms: rank (mean) (Wilcoxon-test result, where 1 indicates the difference is significant important while 0 not). Bold value is statistically significant

with the worst performance. We then calculated the average rank obtained by each algorithm in each metric. To further check whether the difference between the results of DMEA and those of other algorithms is statistically important, we also use the Wilcoxon-test to compare the difference between DMEA and each of the four other algorithms, where 1 indicates the difference is significant important while 0 not.

As can be seen, in terms of GD and HYP, DMEA has the best performance with the lowest average rank, while in terms of IGD and M_2^*, the performance of DMEA is slightly worse than SPEA2 and better than the three other algorithms. Moreover, the Wilcoxon-test results show that most results are significant different.

5.5 Behaviour of the algorithm over time

There is no doubt that when solving the problems, it is important to obtain the best solutions. However, in the design stage, it is also essential to understand the behavior of the algorithms with respect to important criteria. One of these criteria is the performance of DMEA during the evolutionary process. To study this process, we recorded the values of GD and IGD over all generations of one independent run. They were all visualized in Figs. 5, 6, and 7.

It is clear that at the early stage, DMEA's convergence is not fast. However, at the later stage, DMEA managed to get over multiple local POF and finally once it found a way to

Table 4 The rank, mean, and Wilcoxon test on IGD

Problems	NSGA-II	NSGA2DE	SPEA2	MOPSO	DMEA
ZDT1	2 (0.0047)(1)	4 (0.0101)(1)	1 (0.0038)(1)	5 (0.0612)(0)	3 (0.0051)
ZDT2	3 (0.0048)(1)	5 (0.1340)(1)	1 (0.0038)(1)	4 (0.0783)(1)	2 (0.0042)
ZDT3	2 (0.0074)(1)	4 (0.0124)(1)	1 (0.0066)(1)	5 (0.0966)(1)	3 (0.0108)
ZDT4	2 (0.0048)(1)	4 (1.0112)(1)	1 (0.0046)(1)	5 (9.5138)(1)	3 (0.0049)
ZDT6	3 (0.0043)(1)	5 (0.0055)(1)	4 (0.0047)(1)	1 (0.0032)(1)	2 (0.0035)
DTLZ1	2 (0.0981)(1)	5 (0.7455)(1)	3 (0.1586)(0)	4 (0.6369)(1)	1 (0.0218)
DTLZ2	4 (0.0680)(1)	3 (0.0667)(1)	1 (0.0525)(0)	5 (0.0718)(1)	2 (0.0527)
DTLZ3	2 (0.3237)(1)	5 (13.5691)(1)	3 (2.0065)(1)	4 (6.9878)(1)	1 (0.0872)
DTLZ4	3 (0.0984)(1)	2 (0.0693)(1)	5 (0.2881)(1)	4 (0.1661)(1)	1 (0.0525)
DTLZ5	2 (0.0052)(1)	4 (0.0074)(1)	1 (0.0041)(1)	3 (0.0063)(1)	5 (0.0096)
DTLZ6	3 (0.0884)(1)	1 (0.0076)(1)	5 (0.2444)(1)	4 (0.1059)(1)	2 (0.0095)
DTLZ7	2 (0.0730)(1)	3 (0.0973)(1)	1 (0.0592)(1)	5 (0.2357)(0)	4 (0.1506)
Average rank	2.50	3.75	**2.25**	4.08	2.42

Format of the data for DMEA: rank (mean); format of the data for the four other algorithms: rank (mean) (Wilcoxon-test result, where 1 indicates the difference is significant important while 0 not). Bold value is statistically significant

Table 5 The rank, mean, and Wilcoxon test on HYP

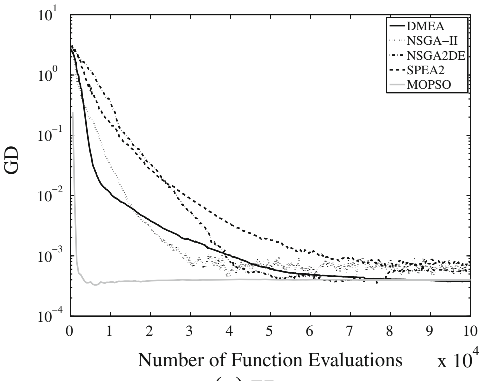
Problems	NSGA-II	NSGA2DE	SPEA2	MOPSO	DMEA
ZDT1	2 (0.7752)(1)	4 (0.7699)(1)	1 (0.7759)(1)	5 (0.7244)(1)	3 (0.7747)
ZDT2	3 (0.6468)(1)	5 (0.5576)(1)	1 (0.6473)(1)	4 (0.6055)(0)	2 (0.6471)
ZDT3	2 (0.6747)(1)	4 (0.6693)(1)	1 (0.6748)(1)	5 (0.6184)(1)	3 (0.6725)
ZDT4	1 (0.9896)(0)	4 (0.9490)(1)	3 (0.9895)(0)	5 (0.6749)(1)	1 (0.9896)
ZDT6	3 (0.4015)(1)	4 (0.4008)(1)	5 (0.3990)(1)	1 (0.4047)(1)	2 (0.4042)
DTLZ1	2 (0.9965)(1)	5 (0.9487)(1)	3 (0.9958)(1)	4 (0.9589)(1)	1 (0.9994)
DTLZ2	5 (0.4491)(1)	4 (0.4527)(1)	1 (0.4784)(1)	2 (0.4755)(1)	3 (0.4724)
DTLZ3	1 (0.9999)(1)	5 (0.9847)(1)	3 (0.9997)(1)	4 (0.9864)(1)	1 (0.9999)
DTLZ4	4 (0.5925)(1)	3 (0.5938)(1)	5 (0.5717)(0)	2 (0.6090)(0)	1 (0.6340)
DTLZ5	2 (0.0957)(1)	4 (0.0942)(1)	1 (0.0961)(1)	3 (0.0955)(1)	5 (0.0926)
DTLZ6	3 (0.7306)(1)	1 (0.7599)(1)	5 (0.6777)(1)	4 (0.7165)(1)	2 (0.7598)
DTLZ7	2 (0.5392)(1)	3 (0.5332)(1)	1 (0.5495)(1)	5 (0.4964)(0)	4 (0.5041)
Average rank	2.50	3.83	2.50	3.67	**2.33**

Format of the data for DMEA: rank (mean); format of the data for the four other algorithms: rank (mean) (Wilcoxon-test result, where 1 indicates the difference is significant important while 0 not). Bold value is statistically significant

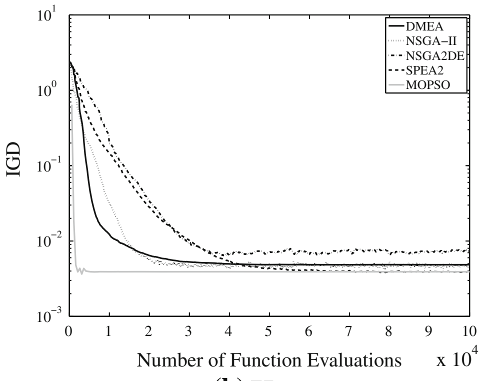
Table 6 The rank, mean, and Wilcoxon test on M_2^*

Problems	NSGA-II	NSGA2DE	SPEA2	MOPSO	DMEA
ZDT1	3 (99.8000)(1)	4 (99.0539)(1)	1 (100.0000)(1)	5 (98.1952)(1)	2 (99.9643)
ZDT2	3 (99.7987)(1)	4 (99.0539)(1)	1 (100.0000)(1)	5 (99.0312)(1)	2 (99.9616)
ZDT3	2 (99.8438)(1)	3 (99.2741)(1)	1 (100.0000)(1)	5 (97.0369)(1)	4 (99.0498)
ZDT4	3 (99.7731)(1)	4 (99.2741)(1)	1 (100.0000)(1)	5 (98.3986)(1)	2 (99.9805)
ZDT6	3 (99.6046)(1)	4 (99.1118)(1)	1 (100.0000)(1)	5 (97.7633)(1)	2 (99.9596)
DTLZ1	3 (99.8835)(1)	4 (99.7859)(1)	1 (100.0000)(1)	5 (99.5000)(0)	2 (99.9811)
DTLZ2	3 (99.9576)(0)	4 (99.6855)(1)	1 (100.0000)(1)	5 (98.4333)(0)	2 (99.9582)
DTLZ3	3 (99.9522)(1)	5 (99.6754)(1)	1 (100.0000)(1)	4 (99.8929)(0)	2 (99.9623)
DTLZ4	4 (96.6384)(1)	2 (99.6504)(1)	5 (93.3332)(1)	3 (97.0000)(1)	1 (99.9859)
DTLZ5	2 (99.7508)(1)	4 (99.1300)(1)	1 (100.0000)(1)	5 (98.4825)(0)	3 (99.1805)
DTLZ6	2 (99.8020)(1)	4 (99.1030)(1)	1 (100.0000)(1)	5 (98.4631)(0)	3 (99.2269)
DTLZ7	2 (99.9805)(1)	4 (99.3246)(1)	1 (100.0000)(1)	5 (97.8589)(0)	3 (99.5145)
Average rank	2.75	3.83	**1.33**	4.75	2.33

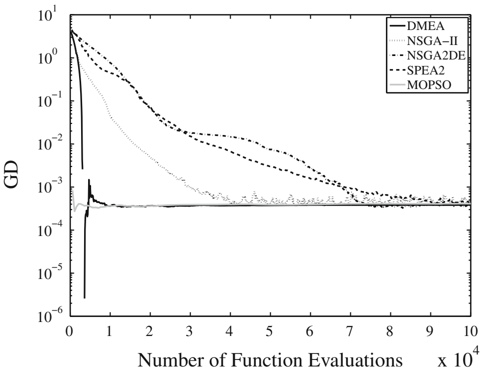
Format of the data for DMEA: rank (mean); format of the data for the four other algorithms: rank (mean) (Wilcoxon-test result, where 1 indicates the difference is significant important while 0 not). Bold value is statistically significant



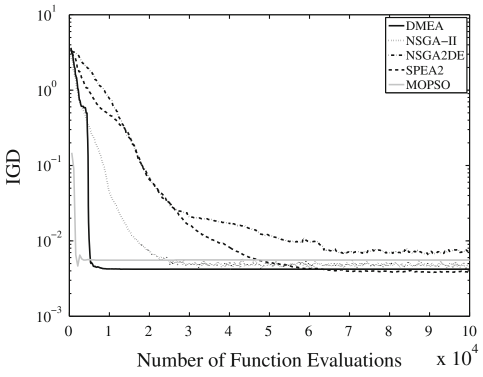
(a) ZDT1



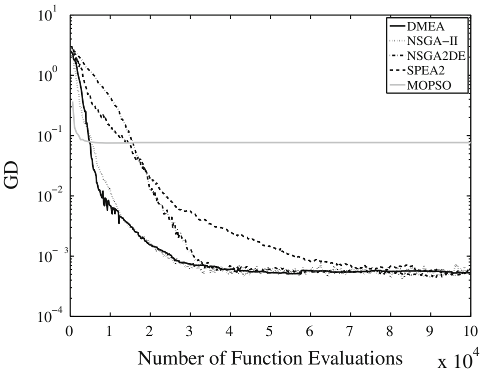
(b) ZDT1



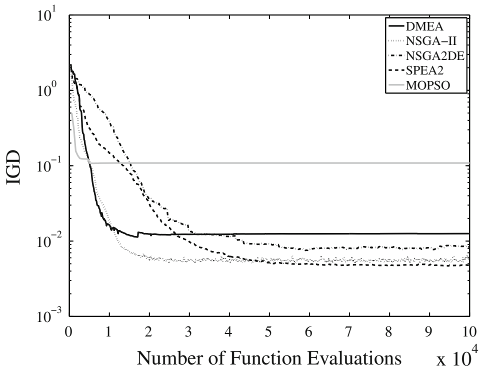
(c) ZDT2



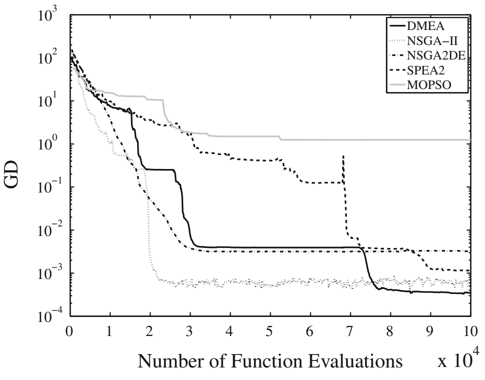
(d) ZDT2



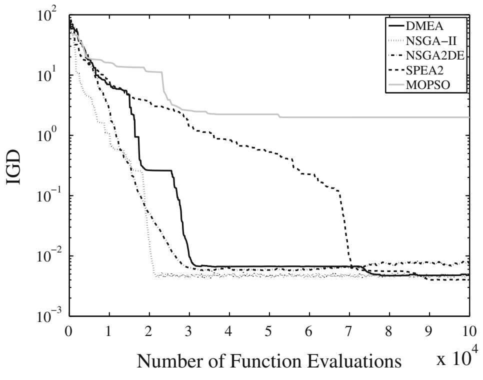
(e) ZDT3



(f) ZDT3

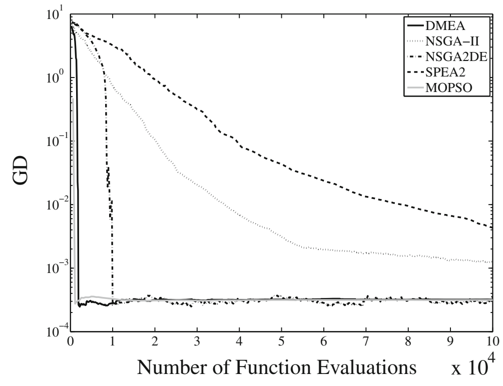


(g) ZDT4

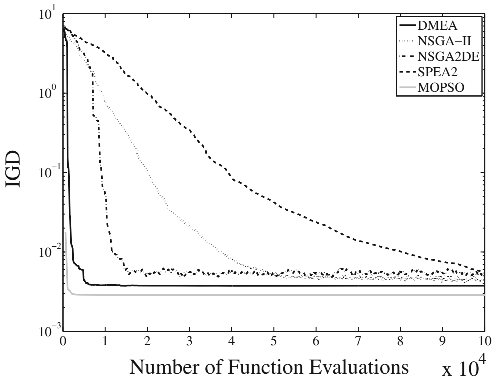


(h) ZDT4

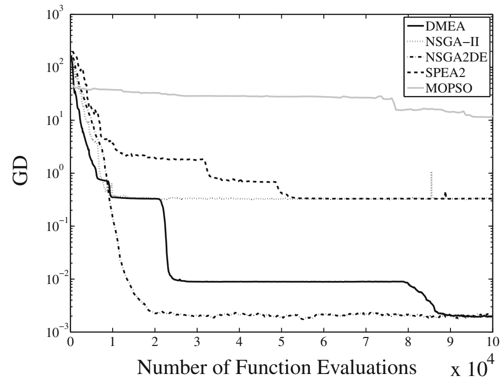
Fig. 5 Visualization of GD (*left*) and IGD (*right*) over time for all studied approaches (ZDT1–ZDT5)



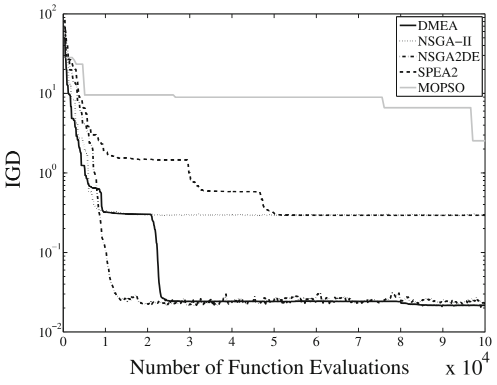
(a) ZDT6



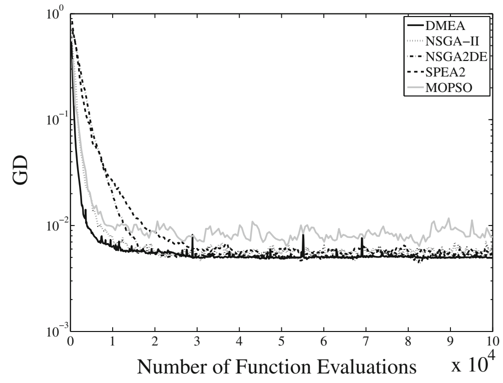
(b) ZDT6



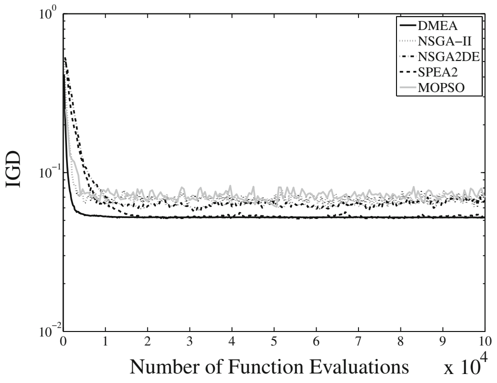
(c) DTLZ1



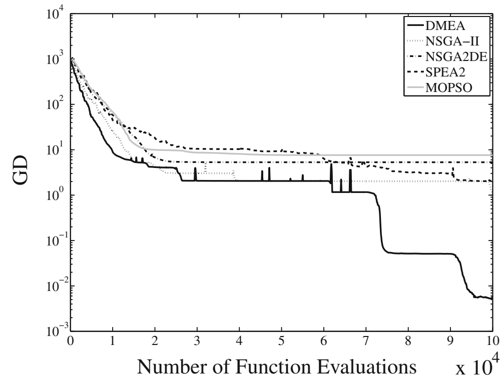
(d) DTLZ1



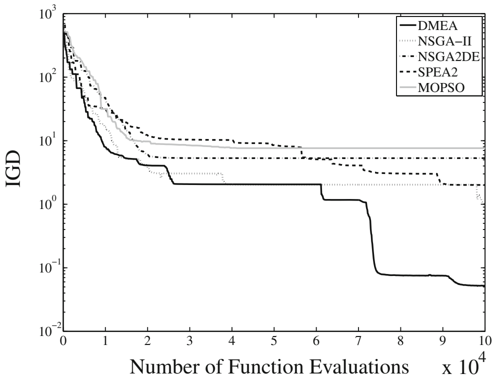
(e) DTLZ2



(f) DTLZ2

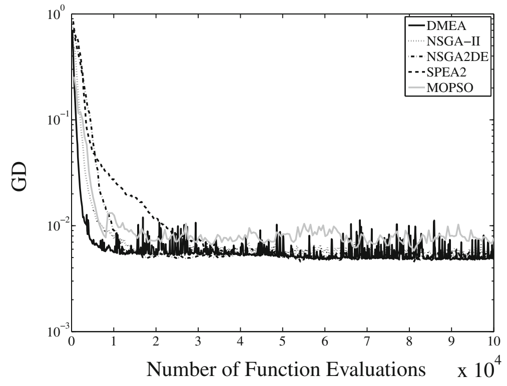


(g) DTLZ3

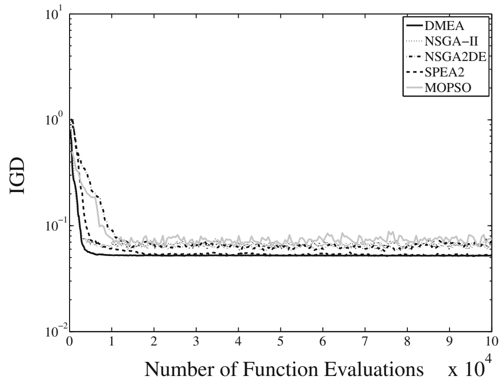


(h) DTLZ3

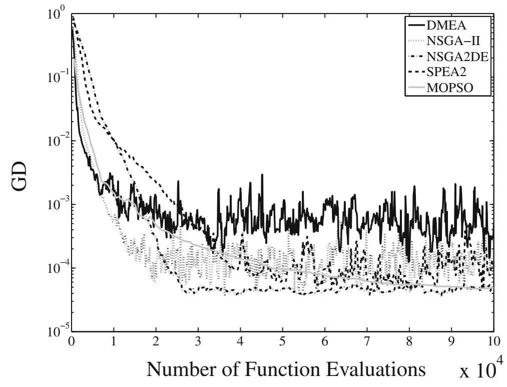
Fig. 6 Visualization of GD (*left*) and IGD (*right*) over time for all studied approaches (ZDT6 and DTLZ1–DTLZ3)



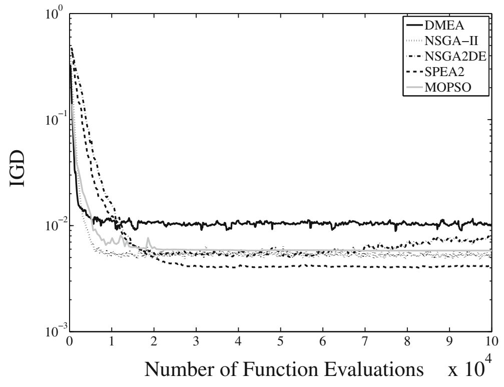
(a) DTLZ4



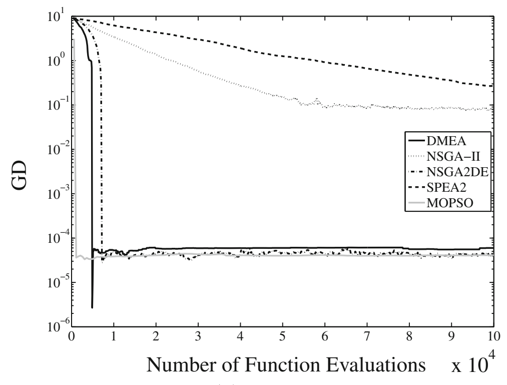
(b) DTLZ4



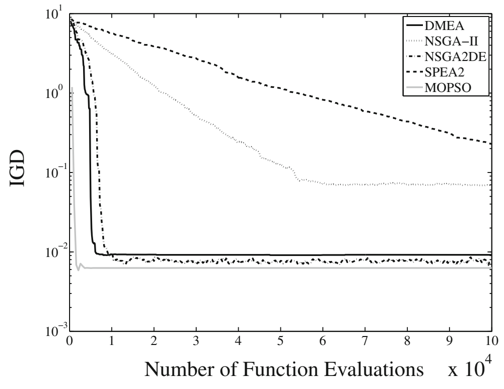
(c) DTLZ5



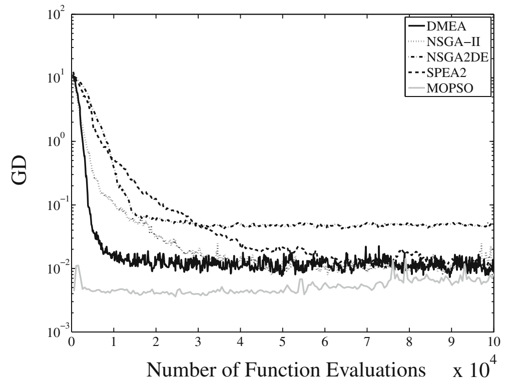
(d) DTLZ5



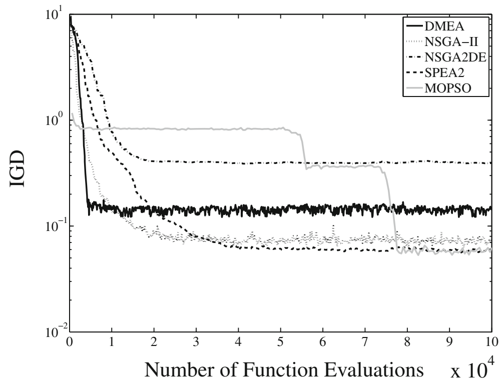
(e) DTLZ6



(f) DTLZ6



(g) DTLZ7



(h) DTLZ7

Fig. 7 Visualization of GD (*left*) and IGD (*right*) over time for all studied approaches (DTLZ4–DTLZ7)

Table 7 The average values (plus the standard error) of GD obtained by DMEA using R1 and R2

Problems	GD		IGD	
	R1	R2	R1	R2
ZDT1	**0.0003 ± 0.0000**	0.0005 ± 0.0000	0.0051 ± 0.0003	**0.0040 ± 0.0000**
ZDT2	**0.0003 ± 0.0000**	0.0004 ± 0.0000	0.0042 ± 0.0001	**0.0041 ± 0.0000**
ZDT3	**0.0004 ± 0.0000**	0.0005 ± 0.0000	**0.0108 ± 0.0010**	0.0567 ± 0.0002
ZDT4	**0.0005 ± 0.0003**	0.0008 ± 0.0009	0.0049 ± 0.0002	**0.0041 ± 0.0006**
ZDT6	0.0003 ± 0.0000	**0.0002 ± 0.0000**	**0.0035 ± 0.0000**	0.0039 ± 0.0000
DTLZ1	0.0025 ± 0.0012	**0.0015 ± 0.0008**	**0.0218 ± 0.0008**	0.0320 ± 0.0013
DTLZ2	**0.0052 ± 0.0003**	0.0054 ± 0.0003	**0.0527 ± 0.0008**	0.0725 ± 0.0017
DTLZ3	0.2248 ± 0.5730	**0.0357 ± 0.1393**	**0.0872 ± 0.9971**	0.1002 ± 0.1367
DTLZ4	0.0056 ± 0.0009	**0.0054 ± 0.0003**	**0.0525 ± 0.0010**	0.0721 ± 0.0020
DTLZ5	**0.0005 ± 0.0003**	0.0174 ± 0.0043	**0.0096 ± 0.0008**	0.0236 ± 0.0030
DTLZ6	**0.0000 ± 0.0000**	**0.0000 ± 0.0000**	**0.0095 ± 0.0007**	0.0139 ± 0.0017
DTLZ7	0.0118 ± 0.0030	**0.0106 ± 0.0020**	**0.1506 ± 0.0184**	0.4562 ± 0.0953

Bold values are statistically significant

the global POF, it quickly passed all other approaches (here, in the case of ZDT4, DTLZ1, and DTLZ3).

5.6 Division of the objective space

In the section on methodology, in order to obtain well-spaced set of non-dominated solutions, we proposed a method to explicitly divide the (bounded) objective space evenly by using a list of non-parallel rays starting from a single position (the ideal point). Each ray was used to track a solution. However, it is possible to use a system of parallel rays. To demonstrate this, we generate a number of rays placed evenly within the objective space and in parallel to the straight line between the origin and its opposite corner point of the unit hyper-cube. For the sake of simplicity, we call the former method R1, while the second (parallel rays) R2. We report the values of GD and IGD for both R1 and R2 in Table 7. These results show that both R1 and R2 were quite competitive.

6 Conclusions

In this paper, we introduced a novel algorithm for employing directions of improvement during the optimization process using a MOEA which we call *direction-based multi-objective evolutionary algorithm* (*DMEA*). The unique properties of DMEA lie with the ways to use the directional, as well as niching information. With DMEA, a population of solutions is evolved over time under the guidance of directions of improvement. At each generation, two types of directions are generated: (1) the convergence direction between a non-dominated solution (stored in an archive) and a dominated solution from the current population, and (2) the spread direction between two non-dominated solutions in the archive. These directions are then used to perturb the current population to get a temporary population of offspring. The combination of this offspring population and the current archive (the combined population) is used to generate the next archive. In addition, the rule for forming the population for the next generation is as follows: the first 50% of the population is filled by the non-dominated solutions and the last 50% is filled by the remaining solutions of the combined population (both dominated and non-dominated solutions). Finally, the selection of non-dominated solutions to fill the archive and the next population is assisted by a new technique of explicit niching in the objective space by using a system of straight lines or rays starting from the current estimation of the ideal point and dividing the space evenly. Each ray is in charge of locating a non-dominated solution.

Experiments on 12 well-known benchmark problems have been carried out to investigate the performance and behavior of the newly proposed algorithm. We also compared its performance with four other well-known algorithms. DMEA showed to be competitive in comparison with these algorithms with respect to both solution convergence and spread. Several analyses on the behaviors of components of the algorithm were thoroughly investigated. Moreover, DMEA pioneers the explicit use of direction information as a shallow local search mechanism within EMOAs. For future work, the use of direction information will be analyzed to characterize its behavior and evolutionary dynamics.

Acknowledgments The authors gratefully acknowledge the financial support from the Australian Research Council via the ARC Discovery Grant No. DP0667123. The first author also gratefully acknowledges financial support from Vietnam's National Foundation for Science and Technology Development (NAFOSTED) Grant No. 107.04-2011.09

References

1. Abbass HA (2006) An economical cognitive approach for bi-objective optimization using bliss points, visualization, and interaction. Soft Comput 10(8):687–698
2. Abbass HA, Sarker R, Newton C (2001) PDE: a Pareto frontier differential evolution approach for multiobjective optimization problems. In: Proceedings of the congress on evolutionary computation, IEEE Service Center, Seoul, Korea, vol 2, pp 971–978
3. Bleuler S, Laumanns M, Thiele L, Zitzler E (2003) Pisa: a platform and programming language independent interface for search algorithms. In: Evolutionary multi-criterion optimization. Lecture notes in computer science, vol 2632. Springer, Berlin, pp 494–508
4. Bui LT, Alam S (2008) Multi-objective optimization in computational intelligence: theory and practice. Information Science Reference Series: IGI Global, Hershey, USA
5. Bui LT, Abbass HA, Essam D (2009) Local models—an approach to distributed multi-objective optimization. Comput Optim Appl 42(1):105–139
6. Caponio A, Neri F (2009) Integrating cross-dominance adaptation in multi-objective memetic algorithms. In: Goh C-K, Ong Y-S, Tan KC (eds) Multi-objective memetic algorithms. Studies in computational intelligence, vol 171. Springer, Berlin, pp 325–351
7. Coello CAC (2006) Evolutionary multi-objective optimization: a historical view of the field. IEEE Comput Intell Mag 1(1):28–36
8. Coello CAC, Veldhuizen DAV, Lamont GB (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer, New York
9. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279
10. Coello CAC, Lamont GB, Veldhuizen DAV (2007) Evolutionary algorithms for solving multi-objective problems. Springer, New York
11. Deb K (2001) Multiobjective optimization using evolutionary algorithms. Wiley, New York
12. Deb K, Thiele L, Laumanns M, Zitzler E (2001) Scalable test problems for evolutionary multi-objective optimization. TIK-Report no. 112. Tech. rep., Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich
13. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
14. DeJong KA (1975) An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arbor
15. Díaz-Madroñero M, Peidro D, Mula J, Ferriols FJ (2009) Fuzzy multiobjective mathematical programming approaches for operational transport planning in an automobile supply chain. J Quant Methods Econ Bus Adm 9:44–68
16. Goh CK, Teoh EJ, Tan KC (2008) Hybrid multiobjective evolutionary design for artificial neural networks. IEEE Trans Neural Netw 19(9):1531–1548
17. Knowles J, Corne D (2000) Approximating the nondominated front using the Pareto archived evolution strategy. Evol Comput 8(2):149–172
18. Kwan CM, Chang CS (2008) Timetable synchronization of mass rapid transit system using multiobjective evolutionary approach. IEEE Trans Syst Man Cybern Part C 38(5):636–648
19. Lara A, Coello CAC, Schuetze O (2010) Using gradient information for multi-objective problems in the evolutionary context. In: GECCO'10, pp 2011–2014
20. Ong YS, Tan KC, Goh C-K (eds) (2009) Multi-objective memetic algorithms. Studies in computational intelligence, vol 171. Springer, Berlin
21. Osman IH, Kelly JP (eds) (1996) Meta-heuristics: theory and applications. Kluwer, New York
22. Price K, Storn R, Lampinen J (2005) Differential evolution—a practical approach to global optimization. Springer, Berlin
23. Rudolph G, Agapie A (2000) Convergence properties of some multi-objective evolutionary algorithms. In: Proceedings of the congress on evolutionary computation. IEEE Press, New York, pp 1010–1016
24. Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Genetic algorithms and their applications: proceedings of the first international conference on genettic algorithms, Hillsdale, New Jersey, pp 93–100
25. Shir OM, Bäck T (2009) Niching with derandomized evolution strategies in artificial and real-world landscapes. Nat Comput 8(1):171–196
26. Shir OM, Preuss M, Naujoks B, Emmerich M (2009) Enhancing decision space diversity in evolutionary multiobjective algorithms. In: Matthias E et al (eds) Proceedings of the 5th international conference on evolutionary multi-criterion optimization, France, pp 95–109
27. Snyman JA (2005) Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms. Springer, Berlin
28. Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report tr-95-012. Tech. rep., ICSI
29. Tan KC, Khor EF, Lee TH (2005) Multiobjective evolutionary algorithms and applications. Springer, Berlin
30. Tan KC, Chiam SC, Mamun AA, Goh CK (2009) Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. Eur J Oper Res 197(2):701–713
31. Veldhuizen DAV (1999) Multiobjective evolutionary algorithms: Classifications, analyses, and new innovation. PhD thesis, Department of Electrical Engineering and Computer Engineering, Air Force Institute of Technology, Ohio, USA
32. Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: methods and applications. PhD thesis, Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
33. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195
34. Zitzler E, Thiele L, Deb K (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(1):173–195
35. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou KC, Tsahalis DT, Periaux J, Papailiou KD, Fogarty T (eds) Evolutionary methods for design optimization and control with applications to industrial problems. International Center for Numerical Methods in Engineering (Cmine), pp 95–100