Nat Comput (2017) 16:661–675
DOI 10.1007/s11047-016-9585-y

Multi-objective differential evolution based on normalization and improved mutation strategy

Noor H. Awad[1] · **Mostafa Z. Ali**[2] · **Rehab M. Duwairi**[2]

Published online: 1 November 2016

Abstract Developing efficient algorithms for solving multi-objective optimization problems is a challenging and essential task in many applications. This task involves two or more conflicting objectives that need to be simultaneously optimized. Many real-world problems fall into this category. We introduce an improved version of multi-objective differential evolution (DE) algorithm, namely MO*n*DE that uses a new mutation strategy and a normalization method to select non-dominated solutions. The new mutation strategy "DE/rand-to-*n*best" uses the best normalized individual in terms of all the objectives to guide the search towards the true pareto optimal solutions. As a result, the probability of producing superior solutions is increased and a faster convergence is achieved. Summation of normalized objective values method is used instead of non-domination sorting to overcome the high computational complexity and overhead problems of sorting non-dominated solutions. The performance of our approach is tested on a set of benchmark problems that consist of two to five objectives. Different combinations of multi-objective evolutionary programming and multi-objective differential evolution algorithms have been used for comparisons. The results affirm the efficiency and robustness of the proposed approach among other well-known algorithms from the literature.

✉ Noor H. Awad
noor0029@ntu.edu.sg

[1] Electrical and Electronic Engineering, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore

[2] Computer Information Systems Department, School of Computer Information Systems, Jordan University of Science and Technology, Irbid 22110, Jordan

Keywords Multi-objective optimization problems · Differential evolution · Summation of normalized objective values method and multi-objective evolutionary programming

JEL Classification 90C27 · 74P99 · 13P25 · 65K10 · 80M50

1 Introduction

Optimization is defined as the selection of the best feasible solution from a set of available alternatives based on solution objectives and some constraints (Deb 2001). Optimization problems are categorized into two types based on the number of objectives: single objective and multi-objective optimization. Single objective optimization is concerned with solving a single objective function and with finding the best solution. This type of optimization provides a decision making tool that gives insights into the nature of the problem, although it cannot handle multiple objectives. The second type is multi-objective optimization, which is concerned with solving multiple conflicting objectives simultaneously (Ehrgott 2005), (Miettinen 1999). Many real-world problems are considered as multi-objective problems because of their nature where two or more conflicting objectives need to be simultaneously optimized (Fogel 1999; Goldberg 1989; Ahn 2006; Sivanandam and Deepa 2008; Rechenberg 1965; Knowles and Corne 1999). In economics, most of the problems involve a process of optimizing multiple conflicting objectives (e.g. consumer's demands for various goods). In finance, the well-known portfolio problem provides a challenge to minimize the risk and maximize the return simultaneously. In the medical field, micro data classifications especially for cancer datasets need to reduce

the number of misclassifications in both testing and learning datasets and extract the appropriate features that have many conflicts (Hamdi-Cherif and Kara-Mohammed 2011). Other applications like electronic chip design formulates the design tradeoffs such as processing time, power consumption and architecture cost as multi-objective problem (Erbas et al. 2006).

Most of the evolutionary algorithms are used for single objective problems (SOPs). When dealing with such problems, one criterion is used to compare all the solutions because there is only one objective. Mathematically, the solutions of a single objective problem are composed of a set of ordered solutions. The selection of the best solution for this type of optimization is given as the minimum or maximum solution, which is related to the nature of the problem. Existing optimization techniques use an evolutionary process that consists of several steps: reproduction, mutation, recombination and selection to guide the search towards optimal solutions. If a problem has multiple objectives that are not conflicting with each other then they can be unified into a single objective and hence form a single objective problem (Michalewicz 1994). Otherwise, if the objectives are conflicting then the problem is called a multi-objective problem (MOP).

Multi-Objective Evolutionary Algorithms (MOEAs) were introduced in 1985 which involve decision making criteria with evolutionary algorithms to find solutions called pareto-optimal solutions for multiple conflicting objectives (Schaffer 1985). Such problems pose a challenge for researchers in providing efficient algorithms that are capable of helping decision makers. The main goal when solving such type of problems is to avoid the problem of premature convergence and stagnation scenario to produce good pareto optimal solutions. Pointing at the developed techniques, we can refer to: Pareto Archived Evolutionary Strategy (PAES) (Knowles and Corne 2000), Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler et al. 2001) and Non-dominated Sorting Genetic Algorithm (NSGA) (Deb et al. 2002).

DE has been successfully extended to solve multi-objective optimization problems. In Babu et al. (2003), a differential evolution with penalty function and a weighing factor method is used to find pareto-optimum sets for an engineering application of a cantilever design problem. In Iorio and Li (2004), a DE approach has been proposed based on correlated self-adapting mutation step sizes to solve rotated multi-objective problems. In Kukkonen and Lampinen (2004), a generalized DE technique that uses a crowdedness mechanism for maintaining good non-dominated solutions is proposed for constrained multi-objective problems. A new paradigm of self-adaptive differential evolution for multi-objective optimization has been introduced in Huang et al. (2009). Furthermore, the hybrid DE approaches have been introduced in, which other evolutionary algorithms or kinds of local search have been merged with DE. In Santana-Quintero et al. (2010) the DEMORS technique, which is a hybrid multi-objective optimization algorithm using differential evolution and a local search based on a rough set theory, has been introduced to solve constrained problems. Another hybridization that combines a self-adaptive DE with a local search method based on sequential quadratic programming can be found in Zamuda et al. (2009).

In this paper, we introduce a differential evolution algorithm with an improved mutation strategy and summation of normalized objectives method. The proposed algorithm uses the differential evolution algorithm, because it is a simple and powerful technique to solve diverse type of problems with stochastic direct search and because of its simplicity. The proposed algorithm is the first to use one of the greedy strategies of differential evolution algorithm in multi-objective optimization. The conflicting objectives state a challenge to choose one of the objectives in a reasonable time. A modified mutation strategy has been proposed, which is able to benefit from all the best knowledge of the best objective-wise solutions. The best objective-wise solution is the solution, which has the best value for one of the objectives. The new mutation strategy, namely "DE/rand-to-*n*best", uses the best normalized individual in terms of all the objectives to guide the search in each optimization step. An external archive is used to store the non-dominated solutions in each generation. The summation of normalized objective value has been used for non-domination sorting to select the well-distributed solutions. This method aims at developing a new sorting method capable of solving the issues that other algorithms have such as computational overhead. The results prove the effectiveness of selecting non-dominated solutions that approximate the true pareto front. The following sections delineate the proposed technique in detail and analyze its structure. Section 2 introduces a review of various multi-objective differential evolution algorithms. Section 3 gives some preliminaries for multi-objective optimization and differential evolution algorithm. Section 4 presents the proposed algorithm in detail. Section 5 presents experimental results. Section 6 draws conclusions of this work.

2 Related work

Differential evolution was proposed by Storn and Price (1997) as a metaheuristic evolutionary algorithm that was designed to solve optimization problems over continuous spaces. Different types of mutation and crossover strategies were proposed for DE (Price et al. 2005; Brest et al. 2006; Das et al. 2009). The individuals are represented as

chromosomes and each decision parameter is encoded by a real value. The initial population space is generated and then evaluated based on an objective function. After that, the selection process, which usually uses three random parents to generate a new child takes place. The difference vector between two parents is computed and added to the third one. For single objective optimization, if the value of the objective function using the newly obtained solution is better than its parent, the child replaces its parent. While in the context of multi-objective optimization, the domination concepts are used to compare both individuals. DE is considered as an effective global optimizer and a robust technique for producing the optimal for many optimization problems and real world applications (Joshi and Sanderson 1999; Zhang et al. 2008).

The classical DE performance is highly dependent on the choice of the mutation strategy and the associated control parameter values for the scaling factor (F), crossover rate (CR) and population size (NP). Any inappropriate use of those control parameters or mutation strategy may lead to premature convergence (Zhang et al. 2009; Gamperle et al. 2002). The early work introduced by Storn and Price (1997) suggested a reasonable value of NP should be between $5D$ and $10D$ where D is the problem dimensions and 0.5, 0.1 were the initial values of F and CR, respectively. Recent works suggested different values for NP, F and CR based on results from experimental studies (Swagatam et al. 2016). These values are fine-tuned according to the tested benchmark problems to help avoiding premature convergence. From such studies, Swagatam et al. concluded that if the population reaches a stagnation state, either F or NP can be increased or the value of CR can be decreased.

Many multi-objective differential evolution algorithms have been successfully introduced to solve multi-objective problems. In Abbass (2002) a self-adaptive pareto DE (SPDA) algorithm has been introduced. In each iteration, a new population space is generated using basic mutation and crossover operations except that a self-adaptive normal distribution is used. In Babu et al. (2003), a new differential evolution technique for MOPs was proposed that uses a penalty function method, and a weighting factor method to find the Pareto optimum set for an engineering application of cantilever design problem. In Iorio and Li (2004), a new DE approach has been proposed based on correlated self-adapting mutation step sizes to solve rotated multi-objective problems. Another approach based on pareto evaluation is applied to optimize the cycle time and cost objectives for an enterprise planning problem (Xue 2003). In Madavan (2002), the elitism, ranking and non-dominated sorting incorporated in NSGA-II Deb et al. (2002) have also been used in differential evolution method. Kukkonen and Lampinen (2004) proposed a generalized DE technique that used a crowdedness mechanism for maintaining the good non-dominated solutions for constrained multi-objective problems. Many other approaches can be found in Mezura-Montes et al. (2008) and Parsopoulos et al. (2004). A new paradigm of self-adaptive differential evolution for multi-objective optimization has been introduced (Huang et al. 2007). The SaDE algorithm (Brest et al. 2006) was extended to MOSaDE by introducing the domination concept when comparing trial and target vectors. A modified version of the aforementioned technique has been introduced in which their proposed DE technique learns its suitable strategy and associated parameters for each objective separately (Huang et al. 2009).

Hybrid DE approaches have been introduced in which other evolutionary algorithms or local searches were merged with DE. In Santana-Quintero et al. (2010) the DEMORS technique, as a multi-objective optimization algorithm that hybridizes differential evolution and local search based on a rough set theory has been introduced for solving constrained multi-objective optimization problems. In their approach there are two main steps, the first step is to use a multi-objective DE technique to generate an initial solution of the pareto front. The second one is a local search technique of a rough set theory to find out more solutions and guide the search towards better solutions. Another hybridization that combines a self-adaptive DE with a local search method based on sequential quadratic programming can be found in Zamuda et al. (2009). An enhanced version of DE algorithm has been merged with simulated annealing algorithm (Chen et al. 2014). In this algorithm, a new acceptance function based on a probability computation is used to utilize simulated annealing for better guiding the search towards better regions. Another technique that aims at reducing the complexity of multi-objective differential evolution by computing the domination ranks and crowding distance is presented in Drozdik (2014). A memetic search that used probabilistic solution principles in the differential evolution algorithm is introduced in Kumar et al. (2014). An advanced teaching–learning technique has been merged with a modified differential evolution algorithm for solving the reactive power dispatch problem as can be found in Ghasemi et al. (2014). The fuzzy clustering problem has been solved using a multi-objective differential evolution algorithm (Das 2014).

The summation of normalized objective value (SNOV) method was used in this context by the multi-objective evolutionary algorithms. In Patel et al. (2011), an improved selection scheme using the SNOV method is used with genetic algorithm. In this algorithm, the SNOV method is used as an improved ranking scheme to select the parents for mutation in genetic algorithm. The summation of

normalized objective value is also used in the differential evolution algorithm. In Qu and Suganthan (2010), the authors replace the non-domination sorting by the use of SNOV and a diversified selection. The solutions are sorted according to a normalized rank that represents the summation of all the individual objectives ranks.

Recently, a modified differential evolution algorithm for multi-objective reactive power (VAR) is introduced (Singh and Srivastava 2014). The aim of this work is to formulate VAR as a multi-objective problem and to use a modified differential evolution algorithm to address the problem of minimizing the real power losses and voltage deviation, simultaneously. The algorithm used a time varying chaotic mutation and crossover to avoid time and effort in tuning DE parameters. A modified differential evolution algorithm for multi-objective optimization is also introduced (Ali et al. 2012). This algorithm uses the concept of random localization in mutation and also a new selection mechanism for generating pareto optimal front. A multi-objective differential evolution algorithm is also proposed in feature selection of classification tasks (Xue et al. 2014). The aim of this work is to minimize the classification error and the number of features at the same time. A differential evolution algorithm for solving multi-label feature selection is also introduced in Zhang et al. (2015). The differential evolution algorithm is also used to solve tunable multi-objective engineering problems (Adeyemo and Olofintoye 2014). In this algorithm, the multi-objective differential evolution and pareto selection methods are used to introduce a new selection scheme. An enhanced differential evolution algorithm is hybridized with simulated annealing for solving multi-objective optimization problems (Chen et al. 2014). Another application that uses a two-phase differential evolution algorithm for solving time–cost trade-offs in resource constrained construction projects is also introduced in Cheng and Tran (2014). More recently, a modified differential evolution algorithm that uses a new diversity maintenance strategy is introduced (Chen et al. 2015). In this algorithm, a new cluster degree measure is used to determine a better spread of non-dominated solutions. An enhanced version of differential evolution algorithm that uses an archive-base mutation is also proposed to provide new direction information toward true pareto front by utilizing useful inferior solutions (Fan and Yan 2015).

3 Scientific background

3.1 Basic concepts of multi-objective optimization

Multi-objective optimization problems consist of several incommensurable and conflicting objectives that need to be optimized simultaneously. They are usually expressed as follows:

$$\begin{aligned} &\text{Min/Max} \quad f(x) := \{f_1(x), f_2(x), \ldots, f_k(x)\}, \\ &\text{Subject to :} \quad g_a(x) \le 0 \quad a = 1, 2, \ldots, n \\ &\qquad\qquad\quad h_b(x) = 0 \quad b = 1, 2, \ldots, p \end{aligned} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)^T$ is the decision variables vector such that $f_i : \Re^n \to \Re$, $i = 1, \ldots, m$ is the objective function, k is the number of objective functions, $g_a, h_b : \Re^n \to \Re$, $a = 1, \ldots, n$, $b = 1, \ldots, p$ are the constraint functions of the problem.

The domination concept is used to relate the association between the two solutions x, y according to the following definitions:

Definition 1 We say x dominates y (denoted by $x \prec_{1\ldots k} y$) if both conditions below are met:

1. x is not worse than y in all objectives $i = 1, \ldots, k$ and
2. x is strictly better than y in at least one objective.

$$\begin{aligned} x \prec_{1\ldots k} y \ \textit{iff} \ \{ &f_i(x) \le f_i(y), \ \forall i \in 1, \ldots, k \ \wedge \ \exists j \in 1, \ldots, k \\ &\Rightarrow \ f_j(x) < f_j(y)\} \end{aligned} \tag{2}$$

Definition 2 The solution $\bar{x}$ is said to be a non-dominated or Pareto-optimal solution if it is not possible to find another $\hat{x} \in \chi$ such that $\hat{x} \prec x$ where χ is the set of non-dominated solutions.

Definition 3 The set of pareto-optimal solutions is called Pareto Front ρF^* which is defined by:

$$\rho F^* = \{f_1(x) \cdots f_m(x) \in \Re^m | x \in \rho^*\} \tag{3}$$

where ρ^* is the pareto optimal set which corresponds to decision variable vectors for the pareto-optimal solutions that can be expressed as:

$$\rho^* = \{x \in \chi | x \, is \, Pareto - optimal\} \tag{4}$$

3.2 Classical differential evolution

The differential evolution algorithm is one of the most promising population-based evolutionary algorithms because of its simplicity with powerful stochastic direct search technique. The algorithm consists of several steps as shown in the following sub-sections.

3.2.1 Initialization step

The standard DE consists of a population of NP individuals. Each individual is represented as a vector of D-dimensional parameters as follows:

$$X_{i,G} = \left\{x_{i,G}^1, \ldots, x_{i,G}^D\right\}, \quad i = 1, \ldots, NP \tag{5}$$

where G is the generation number and NP is the number of individuals in the population space. The DE algorithm aims at evolving the population space towards the global optima distributing random individuals within the search space and preserving the lower and upper bounds of parameters for the problem being solved represented by $X_{\min} = \{x^1_{\min}, \ldots, x^D_{\min}\}$ and $X_{\max} = \{x^1_{\max}, \ldots, x^D_{\max}\}$, receptively. Hence, the initialization of the population space at generation $G = 0$ is formulated in the following equation:

$$x^j_{i,0} = x^j_{\min} + rand(0,1) \cdot (x^j_{\max} - x^j_{\min}) \quad j = 1, 2, \ldots, D \tag{6}$$

where j is the index of parameter value in the i^{th} individual at generation $G = 0$, and $rand(0,1)$ is a uniformly distributed random generator in the range [0,1].

3.2.2 Mutation operation

Five mutation strategies were suggested for the original DE (Price et al. 2005) as described below:

$$\begin{aligned}
&DE/rand/1: \; v_i(t) = x_{r_1}(t) + F \cdot \left(x_{r_2}(t) - x_{r_3}(t)\right)\\
&DE/best/1: \; v_i(t) = x_{best}(t) + F \cdot \left(x_{r_1}(t) - x_{r_2}(t)\right)\\
&DE/current\text{-}to\text{-}best/1: \; v_i(t)\\
&\quad = x_i(t) + F \cdot \left(x_{best}(t) - x_i(t) + x_{r_1}(t) - x_{r_2}(t)\right)\\
&DE/best/2: \; v_i(t) = x_{best}(t) + F \cdot \left(x_{r_1}(t) - x_{r_2}(t) + x_{r_3}(t) - x_{r_4}(t)\right)\\
&DE/rand/2: \; v_i(t) = x_{r_1}(t) + F \cdot \left(x_{r_2}(t) - x_{r_3}(t) + x_{r_4}(t) - x_{r_5}(t)\right)
\end{aligned} \tag{7}$$

The $V_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ is the mutant vector, which is generated for each individual $X_{i,G}$ in the population space.$r^i_1, r^i_2, r^i_3, r^i_4, r^i_5$ are random integer numbers. These numbers represent the indices of chosen individuals within the range [1, NP]. These numbers and the super index i should be mutually exclusive integers. F is the mutation control parameter (scaling factor) and assumes positive values for controlling the scaling ratio of the difference vector. The best individual vector of lowest fitness value in the current population space at generation G is denoted as $X_{best,G}$.

After the initialization step, the mutation phase starts in which each individual $X_{i,G}$ in the population space is mutated according to one of the abovementioned strategies to generate a mutant vector $V_{i,G}$. The scaling factor F controls the speed and convergence of the search space in which the lower values exploit the search towards the local optima and the larger values explore the search towards the global optimum solution.

3.2.3 Crossover operation

After mutant vectors have been generated, the crossover phase is applied to generate new offspring or trial vector $U_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \ldots, u^D_{i,G}\}$. The original DE has defined a binomial crossover as follows (Storn and Price 1997):

$$u^j_{i,G} = \left\{ \begin{matrix} v^j_{i,G}, & if\ (with\ probability\ of\ CR)\ or\ (j = j_{rand}) \\ x^j_{i,G}, & otherwise \end{matrix} \right\} j = 1, 2, \ldots, D \tag{8}$$

CR is the crossover rate, which is a user-defined value within the range [0,1] to control the percentage of parameter values of mutant vectors that should be copied to form a new child solution.j_{rand} is a random index of a position in the mutant vector within the range 1…D.

3.2.4 Selection operation

To ensure that the new trial vectors are within upper and lower bounds, DE must check parameter values of its trial vectors and when they exceed the search range, their values will be reinitialized. After that, the fitness values of trial vectors are calculated by computing the objective function of the problem being solved. Then, a selection criterion is performed as in Eq. 9 to fill in the population space with new individuals for the next generation by comparing fitness values of target and trial vectors.

$$X_{i,G+1} = \left\{ \begin{matrix} U_{i,G}, & if\ f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, & otherwise \end{matrix} \right\} \tag{9}$$

The above DE steps including mutation, crossover and selection will be repeated until a termination criterion is met, which is usually after finishing a specified number of generations. When dealing with multi-objective optimization, the domination concepts are needed to compare the trial and target vectors as shown in the following subsection.

4 Proposed algorithm

A new multi-objective differential evolution is explained in this section. The proposed algorithm introduces a new mutation "DE/rand-to-nbest" and uses the summation of normalized objective value (SNOV) method for selecting the best individual and non-dominated solutions. The following subsections explain the MOnDE algorithm in detail.

4.1 DE/rand-to-nbest

The most widely used strategy in differential evolution is DE/rand/1 that is said to be the most successful mutation strategy (Price et al. 2005). However, greedy strategies such as DE/*best*/*k* and DE/*current-to-best* provide some advantages by benefiting from the information of the best solution in the search process (Gamperle et al. 2002). In this paper, we introduce an improved greedy mutation suitable for multi-objective optimization namely DE/*rand-to-nbest*. The key difference between this new mutation and the original DE/*rand-to-best* is in the way of selecting the best individual. In single optimization, the selection of the best individual during the search is carried out according to the lowest fitness value. However, in multi-objective optimization the conflicting objectives pose a challenge for selecting the best individual. Using the crowding distance to determine the best individual in terms of all objectives at each generation is very time-consuming and imposes a high risk on the diversity of solutions, especially when the number of objectives is more than two (Kukkonen and Deb 2006). The DE/*rand-to-nbest* is proposed to overcome the problem of choosing the best individual in multi-objective optimization based on the summation of normalized objective value (SNOV) method. The SNOV assigns one normalized value for an individual to represent all the different objectives as being from different ranges and without normalization, the obtained distribution might distort. It starts by finding the minimum and maximum value of each objective from the current DE solutions. Then new normalized objective values are assigned to the individuals after adding all the computed values to form one value for each individual. Finally, the individual with the lowest normalized value is returned and is used as the best individual to guide the evolution process by benefiting from the information of all the objective values.

The mutant vector in "DE/rand-to-*n*best'' is generated as shown in Fig. 1. The selection of the best individual is chosen based on SNOV. This process is called in each time the mutant vector is generated. Using this approach, we guarantee that the best M objectives will be used in each generation to guide the search using the best solutions from all objectives. To ensure that the new trial vector v_i is within bounds, lines 3–6 check the values and when they exceed boundaries, their values are reinitialized.

4.2 MOnDE algorithm

The mutation strategy and its control parameters are the dominant settings that reflect the DE performance. The literature studies suggested many ways to enhance the performance of the DE algorithm. The MOnDE algorithm aims at finding the most suitable way of incorporating all the different objectives of the problem being solved. The algorithm suggests a new mutation strategy to overcome the trade-off due to conflicting objectives and chooses the best solution that can guide the search toward pareto optimal solutions. Using the best solution in terms of one objective guides the search in only one direction for only one objective. This leads to an increase in the search speed for one objective without having any enhancement for others. To find a balanced way that aims at enhancing the search for all the objectives, the DE/rand-to-*n*best is proposed. In each generation, this new mutation strategy is called after selecting the best normalized solution. This selection is done by computing the normalized rank for each objective of each solution in the population space then summing all these ranks to have one rank. After that, the solutions are sorted to find the best solution that has the lowest value.

It is known that choosing the non-dominated solutions is the key step for every MOO optimization algorithm, and is considered a time-consuming step. After testing the ability of normalization in selecting the best solution in each generation, a new method based on normalization is also used to select the best non-dominated solutions. After the termination criterion is met and the external archive is filled with the non-dominated solutions in each generation, this method is called. It starts by finding a normalization rank for each solution in the external archive by summing all the normalized values for each objective. Then all the solutions are sorted and the best non-dominated solutions with least values are selected. The number of selected solutions is determined by a pre-defined value and the used summation process keeps the archive domination-free.

The pseudo-code of MO*n*DE algorithm is presented in Fig. 2. It starts by initializing a population of random individuals and evaluates the initial values of objectives for the problem being solved. In each generation, the Summation of Normalized Objective Value (SNOV) method is called and takes the current population as an input to select the best normalized individual. The SNOV assigns one normalized value for each individual to represent all the objectives. This is because the different objectives might be in very different value ranges. Without normalization, the obtained distribution might be distorted. After that the "DE/rand-to-*n*best" is used to generate the mutant vector for each individual using the best normalized individual that benefits from its fast convergence by incorporating information from the best solution for all the objectives in the evolutionary search. After evaluating the trial vector, it is compared with the corresponding target vector. If the new trial vector dominates the target vector, replacement occurs and the trial vector enters the external archive if it dominates some individuals in the archive or if it is non-dominated with all the archive solutions. The aforementioned steps are repeated until the

Algorithm: DE/current-to-*n*best

Input: Entire population *p*at generation *G*: $P_G = x_1^G, \ldots, x_{NP}^G$ **, target vector** x_i **,crossover rate** CR_i **, scaling factor** F_i **, dimension of problem *D*, lower and upper boundaries** $x^{up} = \{x_1^{up}, \ldots, x_D^{up}\}$ **and** $x^{lu} = \{x_1^{lu}, \ldots, x_D^{lu}\}$ **,and number of objectives** M

Output: Trial vector v_i

1. Find minimum and maximum objective value of j^{th} objective, $\{f_j^{\min}, f_j^{\max}\}$
2. Normalize objectives of each individual $i \in (1:NP)$ as follows:

$$fn_j^i = \frac{f_j^i - f_j^{\min}}{f_j^{\max} - f_j^{\min}}, \; j = 1, .., k$$

3. **For** $i = 1:NP$
4. Sum all normalized objectives vector fn^i
5. **EndFor**
6. Find the best individual x_{nbest} of lowest fn value
7. Generate mutant vector V_i as follows
8. **For** $j=1$ to D

$$v_i = x_{nbest} + F.(x_{r1} - x_{r2}) + F.(x_{r3} - x_{r4})$$

9. **If** ($v_i^j > x_j^{up}$)
10. $x_i^j = x_j^{up}$
11. **Else If** ($v_i^j < x_j^{up}$)
12. $v_i^j = x_j^{lu}$
13. **End if**
14. **End For**

Fig. 1 DE/rand-to-*n*best mutation based on summation of normalized objective value (SNOV)

maximum number of function evaluations is reached. Finally, the size of the external archive is adjusted to choose the best non-dominated solutions with the allowable size. To enhance the time complexity and quality of solutions that are considered some of the major shortcomings of other heuristics, which is crucial in the MOO area, we used the normalization sorting in which the solutions are sorted according to their normalized values that the SNOV method computed. Normalization sorting aims at discarding the worst individuals from being selected as the best individual to guide the evolutionary search and also use the best information from all the objectives to keep the search as diverse as possible.

5 Experimental results

5.1 Experimental setup

To test the performance of the proposed algorithm, the CEC 2009 benchmark suite for multi-objective optimization (Zhang et al. 2009a, b) is used. This benchmark suite consists of 13 unconstrained multi-objective instances including 7 functions of two-objectives, 3 functions of 3 objectives and 3 functions of five objectives (Zhang et al. 2009a, b). The algorithm was coded using Java 1.7, and was run on a PC with 2.4 GHz Core processor and 8 GB RAM on windows 7. The experiments are performed with a maximum number of 300,000 function evaluations (FEs) and the sizes of the external archive are 100, 150 and 800 for problems with 2, 3 and 5 objectives, respectively. After extensive experiments and tuning the parameters, the control parameters of the proposed algorithm have been chosen. A sensitivity analysis for *F*, *CR* and *NP* is presented in Table 1. Four values are used to test each parameter. The best settings Based on experimentations for scaling factor (*F*) is 0.5, crossover rate (*CR*) is 0.01 and the size of population space (*NP*) is 200 for all the problems instances. 30 independent runs have been executed for each test problem using the proposed algorithm.

5.2 Performance measure

In multi-objective optimization, a performance metric is needed to compare the performance of different algorithms from two points of view: (1) diversity and (2) convergence approximation set of obtained pareto-optimal solutions.

Algorithm: MO*n*DE

Input: Multi-Objective Problem with *M* objectives

Output: Final approximation set of non-dominated solutions

*1. **Initialization Step***

2. Initializing a population space of NP individuals uniformly distributed in the range $[X^{lu}, X^{up}]$ where $x^{lu} = \{x_1^{lu}, ..., x_D^{lu}\}$ and $x^{up} = \{x_1^{up}, ..., x_D^{up}\}$ as follows: $P_G = \{x_{1,G}, ..., x_{NP,G}\}$ where $x_{i,G} = X^{lu} + rand(0,1)(X^{up} - X^{lu})$, $i = 1, ..., NP$, G is generation number and D is problem dimension.

3. Evaluate the initial population $P_{G=0}$ by computing objectives values $\{f_1, ..., f_M\}$

4. Fill external archive A with the individuals of $P_{G=0}$

*5. **Optimization Step***

6. **While** $G < G_{max}$

7. **Mutate and recombine:**

8. **For** *i=1* to *NP*

9. Randomly choose $x_{r1}, x_{r2}, x_{r3}, x_{r4}$ such that $r_1, r_2, r_3, r_4 \in [1, 2, ..., NP]$, $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i$

10. Choose the best normalized individual X_{nbest} using (SNOV) method

11. Generate mutant vector using "DE/rand-to-nbest" as shown in Fig. 1

$$v_i = x_{nbest} + F.(x_{r1} - x_{r2}) + F.(x_{r3} - x_{r4})$$

12. **Apply Binomial Crossover:**

13. Generate a random position within problem dimension $j_{rand} = \text{randint}(1, D)$

$$u_{i,G}^{j} = \left\{ \begin{matrix} v_{i,G}^{j}, if\ (with\ probability\ of\ CR_i)\ or\ (j = j_{rand}) \\ x_{i,G}^{j}, otherwise \end{matrix} \right\} j = 1, 2, ..., D$$

14. Evaluate the offspring u_i

15. **Domination Selection:**

16. **If** x_i dominates u_i discard u_i

17. **Else If** u_i dominates x_i, replace x_i with u_i

18. **Else If** u_i and x_i are non-dominated, select the best normalized one using SNOV method shown in Figure 5

19. **End if**

20. u_i will be added to the external archive *A* if

(i) u_i dominates some individual of *A* and dominated solutions are removed from A

(ii) or u_i is non-dominated with *A* individuals

21. **End For**

22. Adjust the size of external archive *A* using the SNOV method that takes the best *M* normalized solutions where *M* is the allowable size.

Fig. 2 Pseudo-code of MOnDE algorithm

The inverted generalized distance (IGD) (Zhang et al. 2009a, b) performance metric is used as shown in Eq. (10).

$$IGD(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \quad (10)$$

where $d(v, A)$ is the minimum Euclidean distance between true pareto front P^* and a set of solutions obtained by an algorithm *A*. Smaller IGD values are better for measuring the diversity and convergence of the algorithm.

5.3 Simulations results

The numerical results of the proposed algorithm MO*n*DE are presented in Table 2. The statistical measure of the IGD performance metric including best, worst, mean and

Table 1 Average IGD value for a sensitivity analysis for *F*, *CR* and *NP* for MO*n*DE

F	0.1	0.3	0.5	0.7
F_1	5.4517E−01	1.6842E−01	**1.3180E−03**	3.7845E−02
F_3	6.8425E−01	2.4627E−02	**4.9150E−03**	5.7432E−02
F_5	7.7841E−01	6.7456E−02	**3.9987E−02**	8.4216E−02
F_8	8.7456E−01	9.8616E−02	6.0850E−03	**4.7861E−03**
F_{10}	4.7961E−01	2.6574E−02	**3.8330E−03**	1.2148E−02
F_{11}	9.5426E−01	5.5419E−01	**1.1969E−02**	3.8743E−02
F_{13}	9.7452E−01	7.5412E−02	**6.9751E−02**	8.6321E−02
CR	0.001	**0.01**	0.1	0.2
F_1	9.7425E−02	**1.3180E−03**	5.8743E−01	9.6541E−01
F_3	8.4126E−02	**4.9150E−03**	6.8742E−01	8.3145E−01
F_5	3.7643E−01	**3.9987E−02**	7.6435E−01	9.4692E−01
F_8	7.6428E−02	**6.0850E−03**	5.4631E−01	7.8513E−01
F_{10}	5.8476E−02	**3.8330E−03**	8.2148E−02	7.6189E−01
F_{11}	9.5419E−02	**1.1969E−02**	5.7319E−01	7.5842E−01
F_{13}	9.8426E−02	**6.9751E−02**	7.8452E−01	9.2541E−01
NP	50	100	**200**	300
F_1	9.5483E−01	4.2159E−01	**1.3180E−03**	3.8452E−02
F_3	7.5129E−01	4.6541E−01	**4.9150E−03**	6.8426E−02
F_5	8.9694E−01	5.5419E−01	**3.9987E−02**	7.8456E−02
F_8	7.7456E−01	5.5403E−01	**6.0850E−03**	4.3618E−02
F_{10}	5.7531E−01	2.8435E−02	3.8330E−03	**2.5631E−03**
F_{11}	8.7531E−01	5.4216E−01	**1.1969E−02**	6.8423E−02
F_{13}	9.4256E−01	5.1269E−01	**6.9751E−02**	9.5136E−02

The best entries are marked in boldface

Table 2 The IGD value statistical results of the final approximation set obtained for each test problem for MOnDE algorithm at 30*D*

	Best	Worst	Mean	StD
F_1	3.82E−04	0.002535	0.001318	5.88E−04
F_2	0.002479	0.009405	0.006373	0.001499
F_3	0.003874	0.005758	0.004915	5.34E−04
F_4	0.001595	0.002115	0.00188	1.37E−04
F_5	0.033868	0.044841	0.039987	0.002692
F_6	0.008389	0.010971	0.009618	6.21E−04
F_7	0.001083	0.004847	0.001875	6.87E−04
F_8	0.003702	0.006605	0.006085	6.81E−04
F_9	0.001294	0.007033	0.003161	0.001484
F_{10}	0.003203	0.004644	0.003833	2.83E−04
F_{11}	0.008047	0.016441	0.011969	0.002214
F_{12}	6.640672	10.30076	8.860979	0.850274
F_{13}	0.067742	0.071655	0.069751	9.82E−04

standard deviation over 30 independent runs for all the test functions, are shown in Table 2. Note that most of these results have small values for the mean measure on all the functions F_1–F_{13} except for F_{12}, however the mean value for this function, which equals to 8.03E+00, shows a superior result compared to other MOEAs as the following sub-sections show. Referring to this table, one can find that at 30*D,* the obtained mean values are of order 10^{-4} in 3 problems (F_1, F_2 and F_4), 10^{-3} in 7 other problems (F_3, F_6–F_{11}), 10^{-2} in 2 problems (F_5 and F_{13}) and greater than zero in one problem (F_{12}). Referring to the best and worst values of each problem over 30 independent runs as shown in Table 2, the results reveal the stability of MOnsDE algorithm when generating the optimal pareto front solutions. Moreover, the algorithm was able to converge to the true pareto front very smoothly.

The results in Table 2 reveal that MO*n*DE appears to provide high quality results when optimizing multi-objective problems that consist of 2–5 difficult conflicting objectives at the same time. Results assert the fact that the proposed method effectively solves the multi-objective benchmark problems that reflect complicated real-life problems. The IGD performance metric affirms that MO*n*DE was capable of finding a set of non-dominated solutions that approximate the true pareto front. The smaller mean values of IGD show high diversity and convergence of MO*n*DE algorithm that provide very competitive results to the best known results from other powerful MOEAs as the following sub-sections demonstrate.

5.4 Algorithm running time

The average running time for each multi-objective problem in the benchmark set for 30 independent runs per function is presented in Table 3. It is apparent from the table that the running time of the problems raises slightly as the number of objectives increases. The average running time for the problems of 2 objectives (F_1–F_7) does not have a big difference from those problems with 3 objectives (F_8–F_{10}). The average elapsed time for solving (F_1–F_7) is ranged between (0.06 and 0.086) min and (0.074 and 0.098) min for (F_8–F_{10}). The last three functions appear to require more time ranged from (0.15 to 4.41) min since they consist of 5 objectives that need to be optimized at the same time. However functions F_{11} and F_{12} are slightly different compared to other problems with 2 or 3 objectives but F_{13} took an average of 4 min because of its complexity and difficult nature with new extended rotated features that reflect the real world applications (Huband et al. 2006).

5.4.1 Comparison with other MODE algorithms

In this section, the performance of MO*n*DE algorithm is compared with other algorithms from the literature. Comparisons with other MOEAs algorithms are presented. The chosen MODE algorithms are summarized as follows:

Table 3 Average CPU time for each test problem over 30 independent runs for MO*n*DE algorithm

	CPU time in minutes
F_1	0.059714
F_2	0.071034
F_3	0.086264
F_4	0.069011
F_5	0.06485
F_6	0.066488
F_7	0.058705
F_8	0.098333
F_9	0.080461
F_{10}	0.074511
F_{11}	0.152905
F_{12}	0.271422
F_{13}	4.415561

1. GDE3 (Kukkonen and Lampinen 2009): The third version of the generalized differential evolution. The authors proposed a new concept called constraint domination to compare solutions for constrained problems. The authors didn't use an external archive for holding non-dominated solutions. Instead, they used a population space of size equals to the size of the approximation set. They also used a pruning technique in each generation if the size of the population space exceeds the allowable size since they added the target and trial individuals if they are non-dominated by each other. The F and CR were set to 0.5 and 0.0 respectively.
2. OWSaDE (Huang et al. 2009): An objective-wise self-adaptive differential evolution algorithm in which suitable mutation strategies and control parameters were learned for each objective. The authors used DE/rand/1 and DE/rand/2 with binomial crossover. The scaling factor F is generated using a normal distribution of a mean ranging from [1.0, 0.05] and a standard deviation of 0.1. The crossover rate CR is also generated using normal distribution of a mean changed dynamically based on the success during previous generations and a standard deviation of 0.1. The population size is set to 50 and for non-domination sorting they used harmonic average distance.
3. DECMOSA-SQP (Zamuda et al. 2009): A hybrid approach that combines Differential Evolution with Self-adaptation and sequential quadratic programming as local search. The authors assigned F and CR for each individual in the population and adapted their values in each generation using a Gaussian distribution between 0 and 1. The size of the population is set to the same size of the approximation set.
4. MO-ABC/DE (Rubio-Largo et al. 2012): A multi-objective artificial bee colony with differential evolution for unconstrained multi-objective optimization. In this hybrid approach, the collective intelligence of bee swarms and differential evolution properties are combined together to develop a new enhanced algorithm that is capable of solving multi-objective problems.

A comparison between MO*n*DE algorithm and the aforementioned MODE algorithms is presented in Table 4. The table shows the mean values of IGD performance metric over 30 independent runs. This comparison tests the validity and performance between our proposed algorithm and other variations of multi-objective differential evolution algorithms which follow in three categories. Generalized differential evolution with constant values for F and CR (GDE3), self-adaptive differential evolution (OWMOSaDE, DECMOSA-SQP and MO*n*DE) and hybrid differential evolution with local search (DECMOSA-SQP, MO-ABC/DE and MO*n*DE). The results show that MO*n*DE outperforms the three categories on all the benchmark functions. This affirms the effectiveness of using Cauchy and Normal distributions to adapt the F and CR values compared to the self-adaptive style used in OWMOSaDE and DECMOSA-SQP algorithms. Another perspective of comparison is the use of hybrid approaches. Our proposed technique outperforms other hybrid approaches that uses sequential quadratic programming in DECMOSA-SQP algorithm and artificial bee colony in MO-ABC/DE. The MO*n*DE algorithm showed its capability when optimizing problems of five objectives as shown in the last three functions compared with other MODE algorithms. Those functions are considered the most difficult problems in multi-objective benchmark problems in general because of their complex nature with 5 conflicting objectives that resemble sample complicated real-life applications. Hence it represents a good measure for studying the performance of any MOEA algorithm. It is clear that the proposed algorithm (MO*n*DE) was better than all other MODE algorithms in those functions with a big difference especially for F_{12} which is a round E+02.

5.4.2 *Comparison with other MOEAs algorithms*

Other recent multi-objective evolutionary algorithms are also used for comparisons and they are briefly described as follows:

1. NSGAIILS (Sindhya et al. 2009): The last version of non-sorting GA, which is hybridized with an achievement scalarizing function (ASF) as a local search to approximate pareto front points only.
2. MOEA/D (Zhang et al. 2009a, b): Multi-Objective Evolutionary Algorithm based on decomposition.

Table 4 Average IGD value for a comparison between MODE algorithms and MO*n*DE

Fun	GDE3	OWMOSaDE	DECMOSA-SQP	MO-ABC/DE	MOnDE
F_1	5.34E−03	1.22E−02	7.70E−02	6.32E−03	**1.32E−03**
F_2	1.20E−02	8.10E−03	2.83E−02	**6.14E−03**	6.37E−03
F_3	1.06E−01	1.03E−01	9.35E−02	4.55E−02	**4.92E−03**
F_4	2.65E−02	5.13E−02	3.39E−02	2.90E−02	**1.88E−03**
F_5	3.93E−02	4.30E−01	1.67E−01	**2.50E−02**	4.00E−02
F_6	2.51E−01	1.92E−01	1.26E−01	8.65E−02	**9.62E−03**
F_7	2.52E−02	5.85E−02	2.42E−02	5.61E−02	**1.88E−03**
F_8	2.49E−01	9.45E−02	2.16E−01	1.87E−01	**6.09E−03**
F_9	8.25E−02	9.83E−02	1.41E−01	2.77E−01	**3.16E−03**
F_{10}	4.33E−01	7.43E−01	3.70E−01	2.93E−01	**3.83E−03**
F_{11}	2.34E−01	3.95E−01	3.83E−01	N.A	**1.20E−02**
F_{12}	2.02E+02	7.35E+02	9.43E+02	N.A	**8.86E+00**
F_{13}	3.21E+00	3.26E+00	1.92E+00	N.A	**6.98E−02**

The best entries are marked in boldface

N.A not available

They converted the multi-objective problem to a number of single optimization problems using Tchebycheff approach.

3. MTS (Tseng and Chen 2009): Multiple trajectory search for the multi-objective optimization. The technique consists of three local search methods with different size of neighborhood regions to generate foreground and background solutions.
4. ClusteringMOEA (Wang et al. 2009): Clustering multi-objective evolutionary algorithm based on orthogonal and uniform design.
5. LiuLiAlgorithm (Liu and Li 2009): Multi-objective evolutionary algorithm based on determined weight and sub-regional search.
6. DMOEADD (Liu et al. 2009): Multi-objective evolutionary algorithm based on domain decomposition to divide the decision variable domain into different sub-domains.
7. MOPC (Waldock and Corne 2010): Multi-objective probability collectives. This algorithm uses the probability collectives to evaluate multi-objective optimization.
8. MOPC/D (Morgan et al. 2013) A new probability collectives algorithm for multi-objective optimization. This algorithm is an enhanced version of MOPC which uses decomposition strategies.
9. AMGA (Tiwari et al. 2009): A hybrid archive-based micro genetic algorithm with gradient-based optimizer.
10. OMOEAII (Gao et al. 2009): Multi-objective evolutionary algorithm based on orthogonal lower-dimensional crossover and linear breeding operator.
11. SNOVMOGA (Patel et al. 2011): An improved ranking scheme for selection of parents in multi-objective genetic algorithm.

The new algorithm (MO*n*DE) is compared with these MOEAs as the best performing algorithms from the literature on the used benchmarks. It is worth noticing that all the compared algorithms including MODE algorithms used the same general control parameters which are: number of function evaluations as 300,000, the size of external archive is set to 100, 150 and 800 for 2, 3 and 5 objectives-problems, respectively. And the same true pareto front have been used for computing IGD values.

The mean values for IGD are presented in Tables 4 and 5. As shown in the results, there is a clear difference between MO*n*DE and all the other algorithms. The MO*n*DE obtained the best results for all the functions except for F_5. MO*n*DE got the second best for this function with a difference equals to 0.0151 compared to MTS algorithm, which obtained the first rank when tested using this function. Tables 4 and 5 shows a significant difference in F_{11}, F_{12}, and F_{13}. Those problems are considered the most difficult problems as pointed earlier. Referring to those results, one can simply notice that the proposed algorithm outperformed all MODE and the other MOEAs algorithms that reported the best results in the literature on these benchmarks.

The results in Table 5 that the proposed algorithm is able to generate good solutions near to the true pareto front solutions in most of the functions. The final approximation set for F_3, F_4, and F_5 is small, as a result when a solution is considered as a non-dominated solution, a large number of weak solutions have been removed from the optimal true pareto set. The MOnDE algorithm was able to generate limited number of non-dominated solutions and this reflects the IGD value for those functions. That explains that our proposed algorithm is ranked the second in F_5 after the MTS algorithm. F6 follows the same behavior except that F_6 consists of two continuous regions as F_5.

Table 5 Average IGD value for a comparison between MOEAs algorithms

	NSGAIILS	MOEA/D	MTS	ClusteringMOEA	LiuLiAlgorithm	DMOEADD	MOPC	MOPC/D	AMGA	OMOEAII	SNOVMOGA	MOnDE
F_1	1.15E−02	4.35E−03	6.46E−03	2.99E−02	7.85E−03	1.04E−02	2.42E−02	9.70E−03	3.59E−02	8.56E−02	5.74E−02	**1.32E−03**
F_2	1.24E−02	6.79E−03	6.15E−03	2.28E−02	1.23E−02	6.79E−03	3.86E−02	1.01E−02	1.62E−02	3.06E−02	1.10E−02	**6.37E−03**
F_3	1.06E−01	7.42E−03	5.31E−02	5.49E−02	1.50E−02	3.34E−02	1.74E−01	1.28E−02	7.00E−02	2.71E−01	3.11E−02	**4.92E−03**
F_4	5.84E−02	6.39E−02	2.36E−02	5.85E−02	4.35E−02	4.27E−02	1.15E−01	4.26E−02	4.06E−02	4.62E−02	1.63E−02	**1.88E−03**
F_5	5.66E−01	1.81E−01	**1.49E−02**	2.47E−01	1.62E−01	3.15E−01	5.02E−01	1.46E−01	9.41E−02	1.69E−01	8.20E−01	4.00E−02
F_6	3.10E−01	5.87E−03	5.92E−02	8.71E−02	1.76E−01	6.67E−02	1.12E−01	7.24E−02	1.29E−01	7.34E−02	3.12E−01	**9.62E−03**
F_7	2.13E−02	4.44E−03	4.08E−02	2.23E−02	7.30E−03	1.03E−02	5.21E−02	1.13E−02	5.71E−02	3.35E−02	1.04E−02	**1.88E−03**
F_8	8.63E−02	5.84E−02	1.13E−01	2.38E−01	8.24E−02	6.84E−02	3.69E−01	7.71E−02	1.71E−01	1.92E−01	3.23E−01	**6.09E−03**
F_9	7.19E−02	7.90E−02	1.14E−01	2.93E−01	9.39E−02	4.90E−02	1.51E−01	7.87E−02	1.89E−01	2.32E−01	1.08E−01	**3.16E−03**
F_{10}	8.45E−01	4.74E−01	1.53E−01	4.11E−01	4.47E−01	3.22E−01	4.49E−01	3.41E−01	3.24E−01	6.28E−01	5.74E−02	**3.83E−03**
F_{11}	1.75E−01	1.10E−01	4.55E−01	1.24E+00	1.33E−01	1.20E+00	N.A	N.A	N.A	N.A	N.A	**1.20E−02**
F_{12}	1.58E+02	1.47E+02	3.05E+02	1.04E+03	4.45E+02	4.78E+02	N.A	N.A	N.A	N.A	N.A	**8.86E+00**
F_{13}	3.23E+00	1.85E+00	1.91E+00	3.40E+00	2.29E+00	2.00E+00	N.A	N.A	N.A	N.A	N.A	**6.98E−02**

The best entries are marked in boldface

N.A not available

An extensive statistical analysis was added for the purpose of evaluating the statistical significance of observed performance differences. For an input of n algorithms, this analysis employs a statistical test procedure to rank the performance of each compared algorithm. The test highlights whether there are statistical significant differences in the performance ranking of at least one pair of these compared algorithms. For this purpose, the Friedman test which is a non-parametric multiple comparison test is used to test the differences between the 15 compared algorithms (including MO*n*DE). Table 6 presents the average rankings using the Friedman test. The last two rows in Table 6 show the test-statistic for this test and the corresponding measured p value, respectively. These p values suggest that there are significant differences among the compared algorithms at the 0.05 significance level. Friedman test assigns the lowest rank to the best performing algorithm. As it can be seen in this table, the lowest rank score was obtained by MO*n*DE (rank = 1.6000) with a clear difference compared to MTS as the second-best performing algorithm (rank = 5.5000). Furthermore, a post hoc analysis is used to inspect which algorithm exhibits significant variation relative to the MO*n*DE as a base algorithm. Table 7 demonstrates the results of the post hoc Holm, Holland, Rom and Finner tests. Referring to the p value for each post hoc test with a p value ≤ 0.05, there is a statistical significant difference between the proposed work and all the other contestant algorithms.

Table 6 Average ranking of competitor algorithms, achieved by the Friedman test

Algorithm	Ranking
GDE3	8.6500
OWMOSaDE	11.2000
DECMOSA-SQP	10.4000
MO-ABC/DE	6.9000
NSGAIILS	10.8500
MOEA/D	5.6500
MTS	5.5000
ClusteringMOEA	11.3000
LiuLiAlgorithm	7.5000
DMOEADD	5.7500
MOPC	13.4000
MOPC/D	5.7000
AMGA	10.1000
OMOEAII	12.3000
SNOVMOGA	9.2000
MO*n*DE	**1.6000**
Statistic	6.5947E+01
p value	2.3371E−08

The best entries are marked in boldface

Table 7 A comparison of adjusted p values (control method: MOnDE)

i	hypothesis	p_{Holm}	$P_{Holland}$	P_{Rom}	P_{Finner}
1	MOPC	0.0033333	0.0034137	0.0035067	0.0034137
2	OMOEAII	0.0035714	0.0036571	0.0037572	0.0068158
3	ClusteringMOEA	0.0038462	0.0039379	0.0040461	0.0102062
4	OWMOSaDE	0.0041667	0.0042653	0.0043832	0.0135851
5	NSGAIILS	0.0045455	0.0046522	0.0047816	0.0169524
6	DECMOSA-SQP	0.0050000	0.0051162	0.0052597	0.0203083
7	AMGA	0.0055556	0.0056830	0.0058439	0.0236527
8	SNOVMOGA	0.0062500	0.0063912	0.0065741	0.0269856
9	GDE3	0.0071429	0.0073008	0.0075128	0.0303072
10	LiuLiAlgorithm	0.0083333	0.0085124	0.0087642	0.0336175
11	MO-ABC/DE	0.0100000	0.0102062	0.0105154	0.0369164
12	DMOEADD	0.0125000	0.0127415	0.0131094	0.0402041
13	MOPC/D	0.0166667	0.0169524	0.0166667	0.0434806
14	MOEA/D	0.0250000	0.0253206	0.0250000	0.0467459
15	MTS	0.0354772	0.0347995	0.0378457	0.0470143

Table 8 Average R value for a comparison between MOnDE and SNOV_IS algorithms over 30 independent runs

Fun	SNOV_IS	MOnDE
F_1	−1.07E−03	**−1.05E−02**
F_2	1.12E−06	**8.42E−07**
F_3	1.67E−07	**5.27E−08**
F_4	5.95E−04	**2.83E−05**
F_5	1.20E−03	**9.53E−04**
F_6	2.51E−06	**4.71E−07**
F_7	9.21E−05	**6.29E−06**
F_8	1.50E−03	**1.51E−05**
F_9	−2.91E−02	**−1.06E−01**
F_{10}	−1.03E−02	**−6.38E−01**
F_{11}	4.72E−07	**5.06E−08**
F_{12}	1.25E−05	**6.83E−07**
F_{13}	3.37E−02	**5.27E−03**
F_{14}	−1.27E−02	**−5.37E−01**
F_{15}	2.88E−04	**6.72E−06**

The best entries are marked in boldface

5.5 Comparison on the second benchmark test suits

For the sake of having a thorough comparison with the recent algorithms, we selected the algorithm named: multi-objective differential evolution based on the summation of normalized objectives and improved selection method namely SNOV_IS (Kukkonen and Deb 2006). This algorithm replaces the non-domination sorting with the use of SNOV method and diversified selection methods. They used fixed values for F and CR parameters and they set to 0.5 and 0.1, respectively. Since they didn't mention what the mutation strategy they used and because they used different benchmark problems, we were forced to run the algorithms on their functions in order to get a fair comparison. The used problems are the CEC2007 benchmark (Zhang et al. 2007). Another important performance measure which was used is the R indicator that can be expressed as follows:

$$I_{R2} = \frac{\sum_{\lambda \in \wedge} u^*(\lambda, A) - u^*(\lambda, R)}{|\wedge|} \tag{11}$$

where R is a reference set, u^* is the maximum value reached by the utility function u with weight vector λ on an approximation set A. We choose the augmented Tchebycheff function as the utility function. Table 8 shows the comparison for average R value for 30 independent runs. The proposed algorithm shows a superior performance in comparison to all the algorithms that competed in the CEC2007 benchmark problems but we did not add all their results from their tables and instead compared our work to the results from SNOV_IS algorithm which obtained the first rank in that completion. One can notice that the proposed algorithm shows good performance comparing to SNOV_IS algorithm for all the functions and this proves the power of using the normalization in MOnDE algorithm.

6 Conclusion

In this study, we introduce an improved mutation strategy for Differential Evolution (DE) that is based on the summation of normalized objective values (SNOV) method for solving multi-objective optimization problems. "DE/rand-to-$n$best" is used to help getting out of premature convergence and finding new feasible optimal solutions by choosing the best normalized individual for all the

objectives at the same time to guide the evolution process. The conflicting objectives might have different value ranges and using un-normalized solution might distort the DE distribution. For non-domination sorting, we used SNOV method to fill in the external archive with the best non-dominated solutions and to overcome the time complexity that other methods have. Normalization is capable of discarding the bad solutions which cannot locate the optimal front. The CEC 2009 benchmark suite for multi-objective optimization was used to test the performance of our approach. The IGD metric is used to assess the results. Experimental results indicate that, the proposed algorithm is better and more powerful than other well-known state-of-the-art MOEAs algorithms.

References

Abbass HA (2002) The self-adaptive pareto differential evolution algorithm. In: Proceedings congress on evolutionary computation, vol 1. Piscataway, NJ, pp 831–836

Adeyemo J, Olofintoye OO (2014) Evaluation of combined Pareto multiobjective differential evolution on tuneable problems. Int J Simul Model 13:279–287

Ahn CW (2006) Advances in evolutionary algorithms: theory design and practice (studies in computational intelligence). Springer-Verlag, New York, Inc., Secaucus

Ali M, Siarrya P, Pantb M (2012) An efficient differential evolution based algorithm for solving multi-objective optimization problems. Eur J Oper Res 217(2):404–416

Babu BV, Mathew M, Leenus J (2003) Differential evolution for multi-objective optimization. In: Proceedings of the congress on evolutionary computation, vol 4. IEEE Press, Canberra, Australia, pp 2696–2703

Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evolut Comput 10(6):646–657

Chen B, Zeng W, Lin Y, Zhong Q (2014) An enhanced differential evolution based algorithm with simulated annealing for solving multiobjective optimization problems. J Appl Math 2014: 931630. doi:10.1155/2014/931630

Chen B, Lin Y, Zeng W, Zhang D, Si Y-W (2015) Modified differential evolution algorithm using a new diversity maintenance strategy for multi-objective optimization problems. Appl Intell 43(1):49–73

Cheng M-Y, Tran D-H (2014) Two-phase differential evolution for the multiobjective optimization of time–cost tradeoffs in resource-constrained construction projects. IEEE Trans Eng Manag 61(3):450–461

Das S (2014) Data clustering using multi-objective differential evolution algorithms. In: Proceeding of the 15th annual conference on Genetic and evolutionary computation, GECCO

Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood based mutation operator. IEEE Trans Evol Comput 13(3):526–553

Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York

Deb K, Agrawal S, Pratap A, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197

Drozdik M (2014) Attempt to reduce the computational complexity in multi-objective differential evolution algorithms. In: Proceeding of the 15th annual conference on genetic and evolutionary computation, GECCO, pp 599–606

Ehrgott M (2005) Multicriteria optimization, 2nd edn. Springer, Berlin. ISBN 3-540-21398-8

Erbas C, Erbas S-C, Pimentel A (2006) Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. IEEE Trans Evol Comput 13:945–958

Fan Q, Yan X (2015) Multi-objective modified differential evolution algorithm with archive-base mutation for solving multi-objective p-xylene oxidation process. J Intell Manuf 1–15. doi:10.1007/s10845-015-1087-8

Fogel L (1999) Artificial intelligence through simulated evolution. Wiley, New York

Gamperle R, Muller SD, Koumoutsakos P (2002) A parameter study for differential evolution. In: Proceedings of advances intell Syst Fuzzy Syst Evol Comput, pp. 293–298. Greece

Gao S, Zeng S, Xiao B, Zhang L, Shi Y, Tian X, Yang Y, Long H, Yang X, D. Yu, Yan Z (2009) An orthogonal multi-objective evolutionary algorithm with lower-dimensional crossover. In: Proceeding of IEEE congress of evolutionary computation, 1959–1964, Trondheim

Ghasemi M, Ghanbarian MM, Ghavidel S, Rahmani S, Moghaddam EM (2014) Modified teaching learning algorithm and double differential evolution algorithm for optimal reactive power dispatch problem: a comparative study. Inf Sci 278:231–249

Goldberg DE (1989) Genetic algorithm in search optimization and machine learning. Addison-Wesley Longman Publishing Co., Inc., Boston

Hamdi-Cherif A, Kara-Mohammed C (2011) Evolutionary multiobjective optimization for medical classification in Proceedings of IEEE congress GCC conference and exhibition, pp 441–444

Huang VL, Qin AK, Suganthan PN, Tasgetiren MF (2007) Multi-objective optimization based on self-adaptive differential evolution algorithm. In: Proceedings of the congress on evolutionary computation, Singapore

Huang VL, Zhao SZ, Mallipeddi R, Suganthan PN (2009) Multi-objective optimization based on self-adaptive differential evolution algorithm. Proceedings of IEEE congress of evolutionary computation, pp 190–194. Trondheim

Huband S, Hingston P, Barone L, While L (2006) A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans Evol Comput 10(5):477–506

Iorio AW, Li X (2004) Solving rotated multi-objective optimization problems using differential evolution. Proc Adv Artif Intell 3339:861–872

Joshi R, Sanderson AC (1999) Minimal representation multisensor fusion using differential evolution. IEEE Trans Syst Man Cybern Part A 29(1):63–76

Knowles JD, Corne DW (1999) The Pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization. In: Proceedings of IEEE congress of evolutionary computation. Washington, DC

Knowles JD, Corne DW (2000) Approximating the nondominated front using the Pareto archived evolution strategy. Evol Comput 8(2):149–172

Kukkonen S, Deb K (2006) A fast and effective method for pruning of non-dominated solutions in many-objective problems. Parallel Problem Solving Nat 4193:553–562

Kukkonen S, Lampinen J (2004) An extension of generalized differential evolution for multi-objective optimization with constraints. Parallel Problem Solving Nat 3242:752–761

Kukkonen S, Lampinen J (2009) Performance assessment of generalized differential evolution 3 with a given set of constrained

multi-objective test problems. In: Proceedings of IEEE congress of evolutionary computation, Trondheim, pp 1943–1950
Kumar S, Sharma VK, Kumari R (2014) Memetic search in differential evolution agorithm. Int J Comput Appl 90:6
Liu H-L, Li X (2009) The multiobjective evolutionary algorithm based on determined weight and sub-regional search. In: Proceedings of IEEE congress of evolutionary computation, Trondheim, pp 1928–1934
Liu M, Zou X, Chen Y, Wu Z (2009) Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances. In: Proceedings of IEEE congress of evolutionary computation, Trondheim, pp 2913–2918
Madavan NK (2002) Multiobjective optimization using a pareto diferential evolution approach. Proc Congr Evol Comput 2: 1145–1150
Mezura-Montes E, Reyes-Sierra M, Coello-Coello CA (2008) Multi-objective optimization using differential evolution. A survey of the state-of-the-art. In: Chakraborty UK (ed) Advances in differential evolution, vol 143. pp 173–196
Michalewicz Z (1994) Genetic algorithms + data structures = evolution programs. Springer, New York
Miettinen KM (1999) Nonlinear multiobjective optimization. Kluwer Academic Publishers, Boston
Morgan D, Waldock A, Corne D (2013) MOPC/D: a new probability collectives algorithm for multiobjective optimisation. In: IEEE symposium on computational intelligence in multi-criteria decision-making
Parsopoulos KE, Taoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN (2004) Vector evaluated differential evolution for multiobjective optimization. In: Proceedings of congress on evolutionary computation, vol 1, Portland, Oregon, USA, IEEE Service Center, pp 204–211
Patel R, Raghuwanshi MM, Malik LG (2011) An improved ranking scheme for selection of parents in multi-objective genetic algorithm. In International conference on communication systems and network technologies (CSNT), pp 734–739
Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, New York
Qu BY, Suganthan PN (2010) Multi-objective evolutionary algorithms based on the summation of normalized objectives and diversified selection. Inf Sci 180(17):3170–3181
Rechenberg I (1965) Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library Translation 1122, Farnborough
Rubio-Largo A, Gonzalez-Alvarez DL, Vega-RodrIguez MA, Gomez-Pulido JA and Sanchez-Perez JM (2012) MO-ABC/DE—multiobjective artificial bee colony with differential evolution for unconstrained multiobjective optimization. In: IEEE 13th international symposium on computational intelligence and informatics
Santana-Quintero LV, Hernandez-Diaz AG, Molina J, Coello-Coello CA, Caballero R (2010) DEMORS: a hybrid multi-objective optimization algorithm using differential evolution and rough set theory for constrained problems. Comput Oper 37(3):470–480
Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of an international conference on genetic algorithms and their application, pp 93–100. Pittsburgh
Sindhya K, Sinha A, Deb K, Miettinen K (2009) Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In: Proceedings of IEEE congress of evolutionary computation, pp. 2919–2926. Trondheim
Singh H, Srivastava L (2014) Modified differential evolution algorithm for multi-objective VAR management. Int J Electr Power Energy Syst 55:731–740
Sivanandam S, Deepa S (2008) Introduction to genetic algorithms. Springer, Berlin
Storn R, Price KV (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous Spaces. J Global Optim 11:341–359
Swagatam D, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution—an updated survey. Swarm Evol Comput 27:1–30
Tiwari S, Fadel G, Koch P, Deb K (2009) Performance assessment of the hybrid archive-based micro genetic algorithm (AMGA) on the CEC09 test problems. In: Proceeding of IEEE congress of evolutionary computation, 1935–1942, Trondheim
Tseng L-Y, Chen C (2009) Multiple trajectory search for unconstrained/constrained multi-objective optimization. In: Proceedings of IEEE congress of evolutionary computation, Trondheim, pp 1951–1958
Waldock A, Corne D (2010) Multi-objective probability collectives. In: Proceedings of the 2010 international conference on applications of evolutionary computation, vol 6024. pp 461–470. doi:10.1007/978-3-642-12239-2_48
Wang Y, Dang C, Li H, Han L, Wei J (2009) A clustering multi-objective evolutionary algorithm based on orthogonal and uniform design. In: Proceedings of IEEE congress of evolutionary computation, Trondheim, pp 2927–2933
Xue F (2003) Multi-objective differential evolution and its application to enterprise planning. Proc IEEE Int Conf Robot Autom 3:3535–3541
Xue B, Fu W, Zhang M (2014) Multi-objective feature selection in classification: a differential evolution approach. Springer International Publishing, Basel, pp 516–528
Zamuda A, Brest J, Boskovi B, Zumer V (2009) Differential evolution with self-adaptation and local search for constrained multiobjective optimization. In: Proceedings of IEEE congress of evolutionary computation, Trondheim, pp 195–202
Zhang Q, Zhou A, Zhaoy S, Suganthany PN, Liu W, Tiwar S (2007) Problem definitions for performance assessment on, multi-objective optimization algorithms. Technical report
Zhang J, Avasarala V, Sanderson AC, Mullen T (2008) Differential evolution for discrete optimization: an experimental study on combinatorial auction problems. In: Proceedings IEEE World Congr Comput Intell, Hong Kong, China, pp 2794–2800
Zhang Q, Liu W, Li H (2009) The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In: Proceedings of IEEE congress of evolutionary computation, Trondheim, pp 203–208
Zhang Q, Zhou A, Zhaoy S, Suganthany PN, Liu W, Tiwar S (2009) Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical report
Zhang Y, Gong D-W, Rong M (2015) Multi-objective differential evolution algorithm for multi-label feature selection in classification. Adv Swarm Comput Intell 9140:339–345
Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm. In: Evolutionary methods for design, optimisation and control with application to industrial problems (EUROGEN 2001). pp 95–100