Int. J. Mach. Learn. & Cyber. (2017) 8:989–1007
DOI 10.1007/s13042-015-0479-6

ORIGINAL ARTICLE

An efficient modified differential evolution algorithm for solving constrained non-linear integer and mixed-integer global optimization problems

Ali Wagdy Mohamed[1]

Received: 12 August 2015 / Accepted: 9 December 2015 / Published online: 23 December 2015

Abstract In this paper, an efficient modified Differential Evolution algorithm, named EMDE, is proposed for solving constrained non-linear integer and mixed-integer global optimization problems. In the proposed algorithm, new triangular mutation rule based on the convex combination vector of the triplet defined by the three randomly chosen vectors and the difference vectors between the best,better and the worst individuals among the three randomly selected vectors is introduced. The proposed novel approach to mutation operator is shown to enhance the global and local search capabilities and to increase the convergence speed of the new algorithm compared with basic DE. EMDE uses Deb's constraint handling technique based on feasibility and the sum of constraints violations without any additional parameters. In order to evaluate and analyze the performance of EMDE, Numerical experiments on a set of 18 test problems with different features, including a comparison with basic DE and four state-of-the-art evolutionary algorithms are executed. Experimental results indicate that in terms of robustness, stability and efficiency, EMDE is significantly better than other five algorithms in solving these test problems. Furthermore, EMDE exhibits good performance in solving two high-dimensional problems, and it finds better solutions than the known ones. Hence, EMDE is superior to the compared algorithms.

✉ Ali Wagdy Mohamed
aliwagdy@gmail.com

[1] Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Giza 12613, Egypt

Keywords Evolutionary computation · Global optimization · Differential evolution · Triangular mutation · Mixed-integer non-linear programming (MINLP)

1 Introduction

Generally, optimization is the process of finding the best result for a given problem under certain conditions. In real world problems, applications and different fields of science and engineering, the optimization problems are subject to different types of objective functions and constraints with different type of variables [1]. Thus, most of these problems can be formulated as mixed integer non-linear programming problems (MINLP) that involve continuous as well as integer decision variables. These problems are recognized as a class of NP complete problems plus due to their combinatorial nature, are considered difficult problems [2, 3]. This class of optimization problems frequently arise in various real-world problems and application fields such as mechanical design [4], Synthesis of chemical process flow sheets and design of materials [5], scheduling [6], network design [7], feature selection [8], vehicle routing [9], strategic planning [10], data classification [11] and many more [12, 13]. There are two types of MINLP according to the nature of the objective function and constraints (feasible region): convex MINLP and non-convex MINLP. The former involves minimizing a convex objective function over a convex feasible region and the latter involves minimizing an objective function and/or the continuous relaxation of the feasible region that is not convex [14]. Due to the above mentioned difficulties characteristics and such diverse area, during past decades, many attempts have been made to solve MINLP using various types of deterministic or exact approaches and stochastic or metaheuristic algorithms [15]. There are several

deterministic approaches for solving convex MINLP including Branch-and-Bound Method [16, 17], Generalized Bender Decomposition [18], Outer-Approximation [19, 20], LP/NLP Based Branch-and-Bound [21], Extend Cutting Plane Method [22], Generalized Disjunctive Programming [23] and hybrid methods [24, 25] that combine classical techniques and it is considered most efficient methods [26]. On the other hand, for non-convex MINLPs, the methods are mainly based on the use of exact reformulation approach [27] or using convex envelopes or under-estimators of the non-convex feasible Region [28]. The former one allows obtaining an equivalent convex formulation of the non-convex MINLP in limited cases and all the previous techniques can be applied. The latter includes Piecewise Linear Approximations and Spatial Branch-and-Bound Methods which applies to a larger subset of non-convex MINLPs [29] and many more [30]. More details of convex MINLPs, non-convex MINLPs and their exact methods of solution are referred to the survey papers [26, 29–31]. However, the main drawbacks of these deterministic methods are such that it requires derivatives of the objective function and assumptions of continuity and convexity for the objective function and constraints. Thus, they are only applicable to problems with well-structured problems with good analytical properties. Moreover, most methods can only tackle relatively small-scale problems within a reasonable time. However, for larger problems, the global optimization method also cannot find a feasible solution within an acceptable time period [14] i.e. because MINLPs are generally NP-hard; there are no efficient exact algorithms with polynomial-time complexity for solving them. Thus there are a lot of limitations in these methods. Hence, there has been a need for developing general purpose stochastic approaches that can solve all types of MINLPs especially for handling high-dimensional, highly combinatorial and highly nonlinear problems. Metaheuristics have attracted wide attention as an alternative or hybrid methods to the exact approaches. The main advantages of these methods are that they do not require derivatives information of the objective function and constraints, or assumptions of continuity and convexity for the objective function and constraints. Besides, although these methods cannot guarantee a global solution, near-optimal solutions can often be found with reasonable computational time. However, the drawbacks of these methods are their parameter tunings. There are many stochastic approaches to solve MINLPs, such as simulated annealing algorithms [32, 33], tabu search algorithms [34], genetic algorithms [35–40], evolutionary strategies [41], evolutionary programming [42, 43], controlled random search [44], filled function techniques [45, 46], ant colony [47, 48],particle swarm optimization [49, 50], and differential evolution [51–53]. Virtually, metaheuristics have been originally proposed to overcome the challenges of global optimization problems over continuous space such as nonlinearity, non-convexity, non-continuity, non-differentiability, and/or multimodality, while traditional numerical optimization techniques had difficulties with these problems [54]. Hence, motivated by their success, more research is essentially needed trying to find solutions to MINLPs. However, from the literature, few attempts have been made to develop and extend this work in discrete space. Differential Evolution (DE) is a stochastic population-based search method, proposed by Storn and Price [55]. DE is considered the most recent evolutionary algorithms (EAs) for solving real-parameter optimization problems [56]. While DE shares similarities with other EAs, it differs significantly in the sense that in DE, distance and direction information is used to guide the search process [57]. Das and Suganthan [58] made a comprehensive review for the DE algorithm and introduced its applications to unconstrained/constrained, combinatorial, and multi-criteria optimization problems in detail. Consequently, in this article, motivated by this analysis and discussion, Constrained MINLPs are solved using an efficient modified Differential Evolution algorithm, named (EMDE). The proposed algorithm introduces a new triangular mutation rule based on the convex combination vector of the triangle and the difference vector between the best and the worst individuals among the three randomly selected vectors. The proposed novel approach to mutation operator is shown to enhance the global and local search capabilities and to increase the convergence speed of the new algorithm compared with conventional DE. EMDE uses Deb's constraint handling technique [59] based on feasibility and the sum of constraints violations without any additional parameters. A novel triangular mutation rule without any extra parameters is proposed to balance both exploration capability and exploitation tendency and enhance the convergence speed of the algorithm. The proposed algorithm is tested and analyzed by solving a set of well-known benchmark functions. The proposed algorithm shows a superior and competitive performance to Basic DE and other recent optimization algorithms. It is worth noting that this work is an extension and modifications of our previous work in [54], there are significant differences as follows: (1) Previous work in [54] is proposed for unconstrained problems, whereas this work is proposed for constrained problems. (2) Previous work in [54] optimizes function with continuous variables only, but this work is proposed for optimizing function with mixed variables continuous and integer variables (3) The crossover rate in [54] is given by a dynamic non-linear increased probability scheme, but in this work, the crossover rate is fixed 0.5. (4) in [54], only one difference vector between the best and the worst individuals among the three randomly selected vectors with one scaling factor, uniformly random number in (0,1), is used in the mutation, but in this work, three difference vectors between the tournament best, better and worst of the selected vectors

with corresponding three scaling factors, which are independently generated according to uniform distribution in (0,1), are used in the mutation scheme.(5) the triangular mutation rule is only used in this work, but in the previous work [54], The triangular mutation strategy is embedded into the DE algorithm and combined with the basic mutation strategy DE/rand/1/bin through a non-linear decreasing probability rule. (6) in previous work [54] a restart mechanism based on Random Mutation and modified BGA mutation is used to avoid stagnation or premature convergence, whereas this work does not.

The rest of the paper is organized as follows. Section 2 presents the formulation of the MINLPs. Section 3 introduces the DE algorithm. Section 4 the proposed EMDE is presented in details. Section 5 reports on the computational results of testing benchmark functions. Besides, the comparison with Basic DE and other techniques is discussed. Finally, in Sect. 6, the paper is concluded and some possible future research is pointed out.

2 Problem Statement and constraint handling

In general, Mixed Integer Nonlinear Programming problem can be expressed as follows (without loss of generality minimization is considered here [51, 52]:

$$\begin{aligned} \text{minimize } f(\vec{x},\vec{y}), \vec{x} = (x_1, x_2, \ldots, x_{n_C}) \in \Re^{n_C}, \\ \vec{y} = (y_1, y_2, \ldots, y_{n_I}) \in \Re^{n_I} \end{aligned} \tag{1}$$

subject to:

$$g_j(\vec{x},\vec{y}) \leq 0\,,\ j = 1, \ldots, q \tag{2}$$

$$h_j(\vec{x},\vec{y}) = 0\,, j = q+1, \ldots, m \tag{3}$$

where $(\vec{x},\vec{y}) \in \Omega \subseteq S^{n_C+n_I}$, n_C is the number of continuous variables, $n_I$is the number of integer variables, Ω is the feasible region, and $S$is an $(n_C + n_I)$-dimensional rectangular space in $\Re^{n_C} \cup \Re^{n_I}$ defined by the parametric constraints $l_i \leq x_i \leq u_i$, $1 \leq i \leq n_C$ where l_i and u_i are lower and upper bounds for a continuous decision variable x_i, $l_i \leq y_i \leq u_i$, $1 \leq i \leq n_I$ where l_i and u_i are lower and upper bounds for an integer decision variable y_i, respectively. For an inequality constraint that satisfies $g_j(\vec{x}) = 0$ $(j \in 1, \ldots, q)$at any point $(\vec{x},\vec{y}) \in \Omega$, we say it is active at $(\vec{x},\vec{y})$. All equality constraints $h_j(\vec{x},\vec{y})$, $(j = q+1, \ldots, m)$ are considered active at all points of Ω. Most constraint-handling approaches used with EAs tend to deal only with inequality constraints. Therefore equality constraints are transformed into inequality constraints of the form $\left|h_j(\vec{x},\vec{y})\right| - \varepsilon \leq 0$ where ε is the tolerance allowed (a very small value). In order to handle constraints, we use Deb's constraint handling procedure. Deb [59] proposed a new efficient feasibility-based rule to handle constraint for genetic algorithm where pair-wise solutions are compared using the following criteria:

- Between two feasible solutions, the one with the highest fitness values wins.
- If one solution is feasible and the other one is infeasible, the feasible solution wins.
- If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred.

As a result, Deb [59] has introduced the superiority of feasible solutions selection procedure based on the idea that any individual in a constrained search space must first comply with the constraints and then with the objective function. Practically, Deb's selection criterion does not need to fine-tune any parameter. Typically, from the problem formulation above, there are m constraints and hence the amount of constraint violation for an individual is represented by a vector of m dimensions. Using a tolerance (ε) allowed for equality constraints, the constraint violation of a decision vector or an individual $(\vec{x},\vec{y})$ on the jth constraint is calculated by

$$cv_j(\vec{x},\vec{y}) = \begin{cases} \max\left(0, g_j(\vec{x},\vec{y})\right), & j = 1, \ldots, q \\ \max\left(0, \left|h_j(\vec{x},\vec{y})\right| - \varepsilon\right), & j = q+1, \ldots, m \end{cases} \tag{4}$$

If a decision vector or an individual $(\vec{x},\vec{y})$ satisfies the *jth* constraint, $cv_j(\vec{x},\vec{y})$ is set to zero, it is greater than zero. As discussed [60], in order to consider all the constraints at the same time or to treat each constraint equally, each constraint violation is then normalized by dividing it by the largest violation of that constraint in the population. Thus, the maximum violation of each constraint in the population is given by

$$cv^j_{\max} = \max_{\vec{x} \in s} cv_j(\vec{x},\vec{y}) \tag{5}$$

These maximum constraint violation values are used to normalize each constraint violation. The normalized constraint violations are added together to produce a scalar constraint violation $cv(\vec{x},\vec{y})$ for that individual which takes a value between 0 and 1

$$cv(\vec{x},\vec{y}) = \frac{1}{m}\sum_{j=1}^{m} \frac{cv_j(\vec{x},\vec{y})}{cv^j_{\max}} \tag{6}$$

3 Differential evolution

This section provides a brief summary of the basic Differential Evolution (DE) algorithm. In simple DE, generally known as DE/rand/1/bin [61, 62], an initial random

population consists of NP vectors $\vec{X}_i, \forall\, i = 1, 2, \ldots, NP$, is randomly generated according to a uniform distribution within the lower and upper boundaries $(x_j^{\mathrm{L}}, x_j^{U})$. After initialization, These individuals are evolved by DE operators (mutation and crossover) to generate a trial vector. A comparison between the parent and its trial vector is then done to select the vector which should survive to the next generation [58]. DE steps are discussed below:

3.1 Initialization

In order to establish a starting point for the optimization process, an initial population must be created. Typically, each decision parameter in every vector of the initial population is assigned a randomly chosen value from the boundary constraints:

$$x_{ij}^{0} = \mathrm{a}_j + rand_j \cdot (b_j - \mathrm{a}_j) \tag{7}$$

where $rand_j$ denotes a uniformly distributed number between [0, 1], generating a new value for each decision parameter.

3.2 Mutation

At generation G, for each target vector x_i^G, a mutant vector v_i^{G+1} is generated according to the following:

$$v_i^{G+1} = x_{r_1}^G + F * (x_{r_2}^G - x_{r_3}^G), \quad r_1 \neq r_2 \neq r_3 \neq i \tag{8}$$

with randomly chosen indices $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$. F is a real number to control the amplification of the difference vector $(x_{r_2}^G - x_{r_3}^G)$. According to Storn and Price [61], the range of $F$is in [0, 2]. In this work, in order to verify the boundary constraints, If a component of a mutant vector goes off the search space i.e. if a component of a mutant vector violate the boundary constraints, then the new value of this component is generated using (7).

3.3 Crossover

There are two main crossover types, binomial and exponential. In the binomial crossover, the target vector is mixed with the mutated vector, using the following scheme, to yield the trial vectoru_i^{G+1}.

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^{G+1}, & rand(j) \leq CR\text{orj=rand(i)}, \\ x_{ij}^{G}, & rand(j) > CR\text{andj}\neq\text{rand(i)}, \end{cases} \tag{9}$$

where $j = 1, 2, \ldots, D$, $rand(j) \in [0, 1]$ is the jth evaluation of a uniform random generator number. $CR \in [0, 1]$ is the crossover rate, $rand(i) \in \{1, 2, \ldots, D\}$ is a randomly chosen index which ensures that u_i^{G+1} gets at least one element from v_i^{G+1}; otherwise no new parent vector would be produced and the population would not alter.

In an exponential crossover, an integer value l is randomly chosen within the range $\{1, D\}$. This integer value acts as a starting point in x_i^G, from where the crossover or exchange of components with v_i^{G+1} starts. Another integer value L (denotes the number of components) is also chosen from the interval $\{1, D\text{-}l\}$. The trial vector u_i^{G+1} is created by inheriting the values of variables in locations l to $l + L$ from v_i^{G+1} and the remaining ones from the x_i^G.

3.4 Selection

DE adapts a greedy selection strategy. If and only if the trial vector u_i^{G+1} yields as good as or a better fitness function value than x_i^G, then u_i^{G+1} is set to x_i^{G+1}. Otherwise, the old vector x_i^G is retained. The selection scheme is as follows (for a minimization problem):

$$x_i^{G+1} = \begin{cases} u_i^{G+1}, & f(u_i^{G+1}) \leq f(x_i^G) \\ x_i^G, & \text{otherwise} \end{cases} \tag{10}$$

A detailed description of standard DE algorithm is given in Fig. 1.

1. Begin
2. G=0
3. Create a random initial population $\vec{x}_i^{\,G}\ \forall i, i = 1, \ldots, NP$
4. Evaluate $f(\vec{x}_i^{\,G})\ \forall i, i = 1, \ldots, NP$
5. **For** G=1 to GEN **Do**
6. **For** i=1 to NP **Do**
7. Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, NP]$
8. j_{rand}= randint(1,D)
9. **For** j=1 to D **Do**
10. **If** ($rand_j[0,1] <$ CR **or** j= j_{rand}) **Then**
11. $u_{i,j}^{G+1} = x_{r1,j}^{G} + F \cdot (x_{r2,j}^{G} - x_{r3,j}^{G})$
12. **Else**
13. $u_{i,j}^{G+1} = x_{i,j}^{G}$
14. **End If**
15. **End For**
16. **Verify Boundary constraints**
17. **If**($f(\vec{u}_i^{\,G+1}) \leq f(\vec{x}_i^{\,G})$)**Then**
18. $\vec{x}_i^{\,G+1} = \vec{u}_i^{\,G+1}$
19. **Else**
20. $\vec{x}_i^{\,G+1} = \vec{x}_i^{\,G}$
21. **End If**
22. **End For**
23. G=G+1
24. **End For**
25. **End**

Fig. 1 Description of standard DE algorithm. rand (0,1) is a function that returns a real number between 0 and 1. randint (min, max) is a function that returns an integer number between min and max. *NP, GEN, CR* and *F* are user-defined parameters. *D* is the dimensionality of the problem

4 The proposal: EMDE

In this section, an efficient modified DE algorithm EMDE, the constraint-handling technique and handling of integer variables used in this research are described and explain the pseudo-code of the algorithm in details.

4.1 Triangular mutation scheme

DE/rand/1 is the fundamental mutation strategy developed by Storn and Price [61, 63], and it is reported to be the most successful and widely used scheme in the literature [58]. Obviously, in this strategy, the three vectors are chosen from the population at random for mutation and the base vector is then selected at random among the three. The other two vectors forms the difference vector that is added to the base vector. Consequently, it is able to maintain population diversity and global search capability with no bias to any specific search direction, but it slows down the convergence speed of DE algorithms [64]. DE/rand/2 strategy, like the former scheme with extra two vectors that forms another difference vector, which might lead to better perturbation than one-difference-vector-based strategies [64]. Furthermore, it can provide more various differential trial vectors than the DE/rand/1/bin strategy which increase its exploration ability of the search space. On the other hand, greedy strategies like DE/best/1, DE/best/2 and DE/current-to-best/1 incorporate the information of the best solution found so far in the evolutionary process to increase the local search tendency that lead to fast convergence speed of the algorithm [64]. However, the diversity of the population and exploration capability of the algorithm may deteriorate or may be completely lost through a very small number of generations i.e. at the beginning of the optimization process, that cause problems such stagnation and/or premature convergence. Consequently, in order to overcome the shortcomings of both types of mutation strategies, most of the recent successful algorithms utilize the strategy candidate pool that combines different trail vector generation strategies, that have diverse characteristics and distinct optimization capabilities, with different control parameter settings to be able to deal with a variety of problems with different features at different stages of evolution [64, 65, 66]. Contrarily, taking into consideration the weakness of existing greedy strategies, [67] introduced a new differential evolution (DE) algorithm, named JADE, to improve optimization performance by implementing a new mutation strategy "DE/current-to-*p* best" with optional external archive and updating control parameters in an adaptive manner. Consequently, proposing new mutations strategies that can considerably improve the search capability of DE algorithms and increase the possibility of achieving promising and successful results in complex and large scale optimization problems is still an open challenges for evolutionary computation research. Therefore, this research uses a new triangular mutation rule with a view of balancing the global exploration ability and the local exploitation tendency and enhancing the convergence rate of the algorithm. The proposed mutation strategy is based on the convex combination vector of the triplet defined by the three randomly chosen vectors and three difference vectors between the tournament best, better and worst of the selected vectors. The proposed mutation vector is generated in the following manner:

$$v_i^{G+1} = \bar{x}_c^G + F1 \cdot (x_{best}^G - x_{better}^G) + F2 \cdot (x_{best}^G - x_{worst}^G) + F3 \cdot (x_{better}^G - x_{worst}^G) \tag{11}$$

where $\bar{x}_c^G$ is a convex combination vector of the triangle and $F1$, $F2$ and $F3$ are the mutation factors that are associated with x_i and are independently generated according to uniform distribution in (0,1) and $x_{best}^G, x_{btter}^G and\ x_{worst}^G$ are the tournament best, better and worst three randomly selected vectors, respectively. The convex combination vector $\bar{x}_c^G$ of the triangle is a linear combination of the three randomly selected vectors and is defined as follows:

$$\bar{x}_c^G = w_1 \cdot x_{best} + w_2 \cdot x_{better} + w_3 \cdot x_{worst} \tag{12}$$

where the real weights w_i satisfy $w_i \geq 0$ and $\sum_{1=j}^{3} w_i = 1$. Where the real weights w_i are given by $w_i = p_i / \sum_{i=1}^{3} p_i$, $i = 1, 2, 3$. Where $p_1 = 1$, $p_2 = rand(0.75, 1)$ and $p_3 = rand(0.5, p(2))$, $rand\ (a,b)$ is a function that returns a real number between a and b, where a and b are not included. for constrained optimization problems at any generation $g > 1$, the tournament selection of the three randomly selected vectors and assigning weights follow one of the following three scenarios that may exist through generations, Without loss of generality, we only consider minimization problems:

Scenario 1 If the three randomly selected vectors are feasible, then sort them in ascending according to their objective function values and assign w_1, w_2, w_3 to x_{best}^G, x_{better}^G and x_{worst}^G, respectively

Scenario 2 If the three randomly selected vectors are infeasible, then sort them in ascending order according to their constraint violations *(CV)*values and assign w_1, w_2,$w_3$to x_{best}^G, $x_{better}^G$and x_{worst}^G, respectively

Scenario 3 If the three randomly selected vectors are mixed (feasible and infeasible), then the vectors are sorted by using the three criteria: (a) Sort feasible vectors in front of infeasible solutions (b) Sort feasible solutions according to their objective function values (c) Sort infeasible solutions according to their constraint violations. Accordingly, assign w_1, w_2,w_3 to x^G_{best}, x^G_{better} and x^G_{worst}, respectively. Obviously, from mutation Eq. (11) and (12), it can be observed that the incorporation of the objective function value and constraints violation in the mutation scheme has two benefits. Firstly, the perturbation part of the mutation is formed by the three sides of the triangle in the direction of the best vector among the three randomly selected vectors. Therefore, the directed perturbation in the proposed mutation resembles the concept of gradient as the difference vector is oriented from the worst to the best vectors [68]. Thus, it is considerably used to explore the landscape of the objective function being optimized in different sub-region around the best vectors within constrained search-space through optimization process. Secondly, the convex combination vector $\bar{x}^G_c$ is a weighted sum of the three randomly selected vectors where the best vector has the highest contribution. Therefore, $\bar{x}^G_c$ is extremely affected and biased by the best vector more than the remaining two vectors. Consequently, the global solution can be easily reached if all vectors follow the direction of the best vectors besides they also follow the opposite direction of the worst vectors among the randomly selected vectors. Indeed, the new mutation process exploits the nearby region of each $\bar{x}^G_c$ in the direction of $(x^G_{best} - x^G_{worst})$for each mutated vector. In a nutshell, it concentrates the exploitation of some sub-regions of the search space. Thus, it has better local search tendency so it accelerates the convergence speed of the proposed algorithm. Besides, the global exploration ability of the algorithm is significantly enhanced by forming many different sizes and shapes of triangles in the feasible region through the optimization process. Thus, the proposed directed mutation balances both global exploration capability and local exploitation tendency. Illustrations of the local exploitation and global exploration capabilities of the new triangular mutation are illustrated in Figs. 2 and 3, respectively.

4.2 Constraint handling

In this paper, Deb's feasibility rules [59] are used to handle the constraints. As aforementioned in Sect. 2, in the constraint-handling technique, the equality constraints are transformed to inequality constraints by a tolerance value ε. In the experiments, the tolerance value $\varepsilon$is adopted like [69] and used in the [70]. The parameter ε decreases from generation to generation using the following equation:

$$\varepsilon(g+1) = \begin{cases} \dfrac{\varepsilon(g)}{1.0168}, & if\, \varepsilon > 10^{-4} \\ 10^{-4}, & \text{otherwise}\ . \end{cases} \tag{13}$$

where g is the generation number and the initial $\varepsilon(0)$ is set to 5. Regarding boundary-handling method, the re-initialization method is used (see Eq. 7), i.e. when one of the decision variable violates its boundary constraint, it is generated within the uniform distribution within the boundary, as mentioned in Sect. 3.2. The pseudo-code of NDE is presented in Fig. 4.

4.3 Handling of integer variables

In its canonical form, the Differential Evolution algorithm and EMDE algorithm are only capable of optimizing unconstrained problems with continuous variables. However, there are very few attempts to transform the canonical DE and proposed EMDE algorithms to handle integer variables [51, 53, 71, 72]. In order to enhance the convergence speed and to avoid for avoiding unnecessary searches in real number field, integer variables are directly searched in the integer space, to ensure that the vector is controlled in the integer space in the evolution procedure of the EMDE. In this research, only a couple of simple modifications are required, the new generation of initial population and boundary constraints verification, the trial vector generated after novel mutation operation and the basic mutation schemes and cross over operations use rounding operator, where the operator round(x) rounds the elements of x to the nearest integers. Therefore, the initialization and mutation are as follows:

Initialization: $x^0_{ij} = round(l_j + rand_j * (u_j - l_j))$.
Boundary constraint verification: $x^{G+1}_{ij} = round(l_j + rand_j * (u_j - l_j))$.
New mutation: $u^{G+1}_{ij} = round\,(\bar{x}^G_c + F1 \cdot (x^G_{best} - x^G_{better}) + F2 \cdot (x^G_{best} - x^G_{worst}) + F3 \cdot (x^G_{better} - x^G_{worst}))$.
Basic mutation: $u^{G+1}_{ij} = round(x^G_{r1} + F_g * (x^G_{r2} - x^G_{r3}))$.

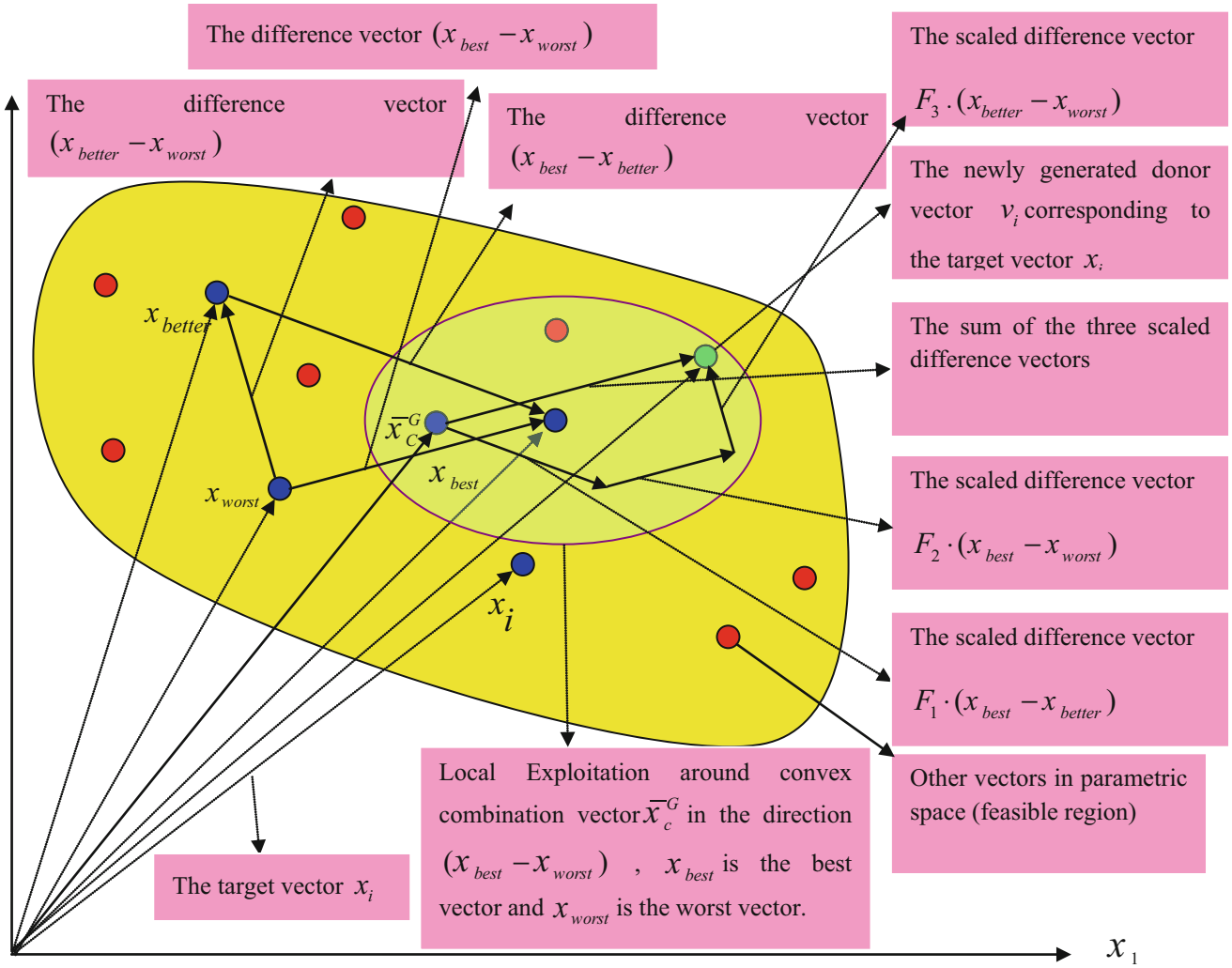

Fig. 2 An illustration of the new triangular mutation scheme in two-dimensional parametric space (Local Exploitation)

5 Experiments and discussion

To demonstrate the feasibility and efficiency of the proposed EMDE algorithm, 18 integer and mixed integer constrained optimization problems in the literature [40] are tested. All (except problem 16) are nonlinear. The number of unknown decision variables in these problems varies from 2 to 100. However, based on self check, taking into consideration that there are few differences in problems 6, 16 and 17 in Ref. [40] from the original documents. Thus, the corrections on these problems are presented in "Appendix A". The three main control parameters of EMDE algorithm, population size *NP* is 50, crossover rate *Cr* is fixed 0.5 and the scale factors *F*1, *F*2and *F*3 are uniformly random numbers in (0.1) as aforementioned. The scale factor *F* of the basic DE is fixed 0.5. The maximum number of generations is 1000. Thus, The maximum number of function evaluations (FEs) for all benchmark problems are set to 50,000. The experiments were executed on an Intel Pentium core i7 processor 1.6 GHz and 4 GB-RAM. EMDE algorithm is coded and realized in MATLAB. For each problem, 50 independent runs are performed and statistical results are provided including the best, median, mean, worst results and the standard deviation. Additionally, A run is considered a success if achieved value of the objective function f(x) is within 1 % of the known best solution or optimal value f(x*), where x* is the best solution or optimal value found i.e.$(f(x)-f(x^*)) < 0.01$. For each problem, the percentage of success (obtained as the ratio of the number of successful runs to total number of runs), the statistical results about number of function evaluations (FEs) in the case of successful runs are provided including the best, median, mean, worst and standard deviation used by the algorithm in achieving the optimal solution in the case of the successful runs are also listed [40]. In order to evaluate the benefits of the proposed

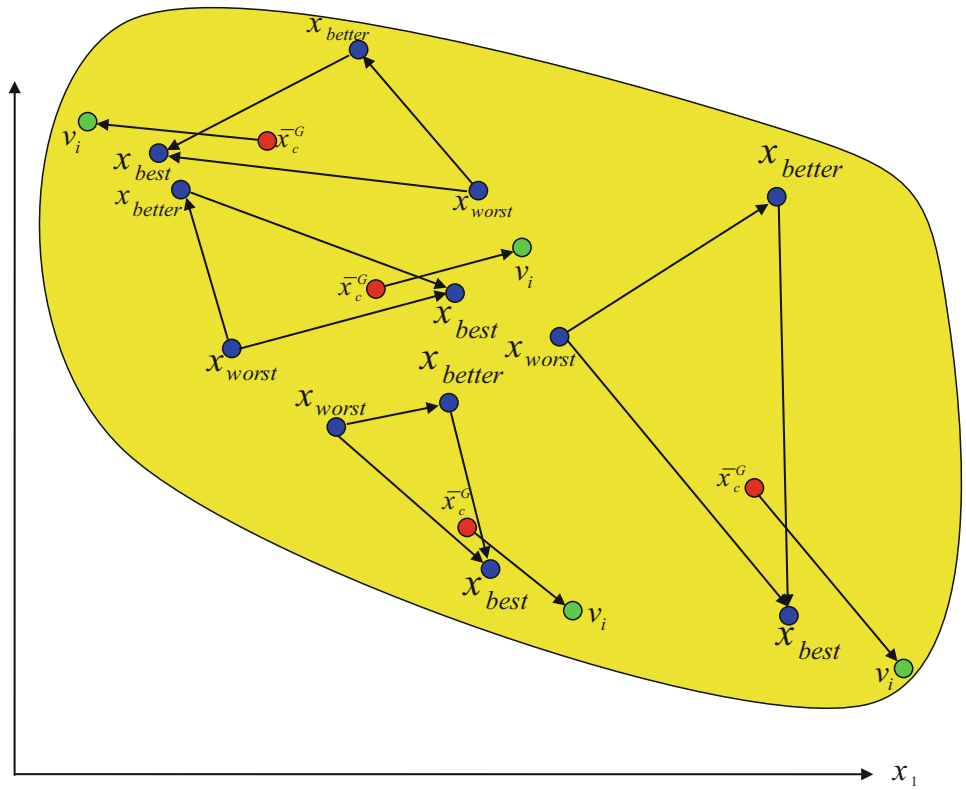

Fig. 3 An illustration of the new triangular mutation scheme with collection of convex combinations vectors and the newly generated donor vectors $v_i$corresponding to the target vectors x_i in two-dimensional parametric space (Global Exploration)

modifications, two experiments are conducted. Firstly, EMDE is compared with basic DE where they only differ in mutation scheme used. Then The proposed approach is compared against the following four state-of-the-art evolutionary algorithms.

- Real Coded Genetic algorithm with Laplace crossover and Power mutation (MI-LXPM) due to Deep et al. [40]. MI-LXPM algorithm is an extension of LXPM algorithm, which is efficient to solve integer and mixed integer constrained optimization problems. In MI-LXPM, Laplace crossover and Power mutation are modified and extended for integer decision variables. Moreover, a special truncation procedure for satisfaction of integer restriction on decision variables and a 'parameter free' penalty approach for constraint handling are used in MI-LXPM algorithm [40].
- The controlled random search technique incorporating the simulated annealing concept (the RST2ANU algorithm) due to Mohan and Nguyen [44]: this algorithm, which is primarily based on the original controlled random search approach of Price, incorporates a simulated annealing type acceptance criterion in its working so that not only downhill moves but also occasional uphill moves can be accepted. In its working it employs a special truncation procedure which not only ensures that the integer restrictions imposed on the decision variables are satisfied, but also creates greater possibilities for the search leading to a global optimal solution.
- Real Coded Genetic Algorithm with Arithmetic crossover and Non-uniform mutation (AXNUM) due to Li and Gen [73]: In AXNUM, Arithmetic crossover and Non-uniform mutation with tournament selection operator are used. It uses always rounding rule for satisfaction of integer restrictions on decision variables. Besides, a new adaptive penalty approach for the evaluation function of genetic algorithm by introducing the degree of the constraints satisfactory in order to solve constrained optimization problems.
- and Modified differential evolution algorithm of constrained nonlinear mixed integer programming (MIPDE) due to Gao et al. [51].: a variant of differential evolution (the framework is based on DE/rand/1/bin combined linearly with best solution found so far. Thus, the positions of the variation particles are self-adaptively adjusted so that the particles evolve in better direction and the feasible basis rule [59] and the

Fig. 4 Description of EMDE algorithm

1.	Begin
2.	G=0
3.	Create a random initial population $\vec{x}_i^G \ \forall i, i = 1,..., NP$, CR= 0.5 .
4.	Evaluate $f(\vec{x}_i^G), cv(\vec{x}_i^G), \forall i, i = 1,..., NP$
5.	**For** G=1 to GEN **Do**
6.	Compute the Factor of the tolerance value (ε) using equation (13).
7.	**For** i=1 to NP **Do**
8.	F_1=rand[0,1] , F_2 =rand[0,1] , F_3 =rand[0,1]
9.	j_{rand}= randint(1,D)
10.	Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, NP]$
11.	**For** j=1 to D **Do**
12.	**If** (rand$_i$[0,1]< CR **or** j= j_{rand}) **Then**
13.	Determine the tournament x_{best}^G, x_{better}^G and x_{worst}^G based on $f(\vec{x}_i^G), cv(\vec{x}_i^G), i = 1,2,3$, (three randomly selected vectors) using one of three scenarios and compute $\overline{x}_c^G$ according to eq.(11).
14.	$u_{i,j}^{G+1} = \overline{x}_{c,j}^G + F_1(x_{best,j}^G - x_{better,j}^G) + F_2(x_{best,j}^G - x_{worst,j}^G) + F_3(x_{better,j}^G - x_{worst,j}^G)$
15.	**Else**
16.	$u_{ij}^{G+1} = x_{ij}^G$
17.	**End If**
18.	**End For**
19.	**Verify Boundary constraints**
20.	**If** ($\vec{u}_i^{G+1}$ is better than $\vec{x}_i^G$ (based on Deb's selection criteria)) **Then**
21.	$\vec{x}_i^{G+1} = \vec{u}_i^{G+1}$
22.	**Else**
23.	$\vec{x}_i^{G+1} = \vec{x}_i^G$
24.	**End If**
25.	Evaluate $f(\vec{x}_i^{G+1})$ and $cv(\vec{x}_i^{G+1}), \forall i, i = 1,..., NP$.
26.	**End For**
27.	G=G+1
28.	**End For**
29.	**End**

Table 1 Results obtained by EMDE in 50 runs

Problem	Known optimal	Best	Median	Mean	Worst	S.D
1	2	2	2	2	2	0.00E + 00
2	2.124	2.124	2.124	2.124	2.124	0.00E + 00
3	1.076543	1.076543	1.076543	1.076543	1.076543	0.00E + 00
4	−6961.81381	−6961.81381	−6961.81381	−6961.81381	−6961.81381	0.00E + 00
5	−68	−68	−68	−68	−68	0.00E + 00
6	−6.00	−6.00	−6.00	−6.00	−6.00	0.00E + 00
7	99.245209	99.239635	99.239635	99.239635	99.239635	0.00E + 00
8	3.557463	3.5574612	3.5574612	3.5589360	3.57958240	5.71E − 03
9	32217.4	32217.4	32217.4	32217.4	32217.4	0.00E + 00
10	0.94347	0.953197	0.953197	0.953197	0.953197	0.00E + 00
11	8	8	8	8	8	0.00E + 00
12	14	14	14	14	14	0.00E + 00
13	−42.632	−42.63212	−42.63212	−42.63212	−42.63212	0.00E + 00
14	0	0	0	0	0	0.00E + 00
15	807	807	807	807	807	0.00E + 00
16	1030361	1352439	1352439	1352439	1352439	0.00E + 00
17	303062432	304153834	304153280	304153230.26	304152465	435.849
18	0.999955	0.99995467	0.99995467	0.99995467	0.99995467	3.83E − 12

Table 2 Results obtained by Basic DE in 50 runs

Problem	Known optimal	Best	Median	Mean	Worst	S.D
1	2	2	2	2.0236	2.236	2.07E − 02
2	2.124	2.124	2.124	2.124	2.124	0.00E + 00
3	1.076543	1.076543	1.076543	1.076543	1.076543	0.00E + 00
4	−6961.81381	−6961.81381	−6961.81381	−6961.81381	−6961.81381	0.00E + 00
5	−68	−68	−68	−68	−68	0.00E + 00
6	−6.00	−6.00	−6.00	−6.00	−6.00	0.00E + 00
7	99.245209	99.243541	99.427363	106.633258	130.1724734	12.35435867
8	3.557463	3.5574612	3.5574612	3.5828044	3.714353429	5.42E − 02
9	32217.4	32217.4	32217.4	32217.4	32217.4	0.00E + 00
10	0.94347	0.953197	0.953197	0.953197	0.953197	0.00E + 00
11	8	8	8	8	8	0.00E + 00
12	14	14	15	14.6666666	16	0.5466722
13	−42.632	−42.632	−42.632	−42.632	−42.632	0.00E + 00
14	0	0	0	0	0	0.00E + 00
15	807	807	807	807	807	0.00E + 00
16	1030361	1352439	1351416.5	1351388.1667	1348739	1027.466421
17	303062432	304139472	304138183	304138406.73	304137318	749.273
18	0.999955	0.99995467	0.99995467	0.99995467	0.99995467	2.41E − 10

Table 3 Better solutions obtained by EMDE than MI-LXPM [40]

problem	Best solution x	Objective function
10	(1,1,0,1,0,1,1,0)	0.953197
16	99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99, 99,99,99,99,99,99,99,99,99,99	1352439
17	(96,99,99,99,99,43,31,4,6,2,99,99,99,99,99,45,78,88,66,99,98,99,99,99,99,99,99,99,99,99,99, 5,99,98,99,99,0,0,99,99,98,99,2,7,82,4,1,10,2,10,99,99,99,97,98,87,62,98,61,81,91,98,96, 81,80,98,99,43,95,80,99,98,98,65,90,32,96,97,95,92,92,91,98,95,99,88,99,91,62,21,97,64,92,0,12, 96,92,72,12,70)	304153834

Table 4 Better solutions obtained by basic-DE than MI-LXPM [40]

problem	Best solution x	Objective function
10	1,1,0,1,0,1,1,0	0.953197
16	99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99, 99,99,99,99,99,99,99,99,99,99,99,99	1352439
17	(94,99,94,98,97,4,7,70,7,13,98,90,99,99,97,35,82,80,91,90,89,99,97,98,88,96,99,99,98, 78,99,6,99,96,99,99,4,21,97,99,94,99,2,41,92,15,9,16,8,3,99,96,95,45,97,63,14,28,69, 90,95,38,50,61,20,58,77,95,61,71,80,89,22,75,67,56,87,94,86,96,82,57,82,94,91,42, 85,90,64,33,85,62,92,7,34,83,40,47,45,53)	304139472

Table 5 Number of FEs to achieve the fixed accuracy level (f(x)-f(x*)) < 0.01), successful rate (Ps) provided by EMDE

Problem	Best	Median	Mean	Worst	Std	Ps
1	50	575	570	1550	399.266569	100
2	50	100	126.666667	300	84.233236	100
3	1050	1450	1393.333333	1750	217.835674	100
4	7650	7900	7993.333333	8950	332.665999	100
5	50	450	448.333333	700	169.422617	100
6	50	50	50	50	0.0E + 00	100
7	1500	2175	2192.857143	3100	560.170696	100
8	3400	5300	5504.166667	7650	954.954662	90
9	250	850	850	1250	263.899331	100
10	50	50	55	200	27.386128	100
11	50	250	173.333333	350	103.279556	100
12	350	800	900	1900	422.447084	100
13	50	50	50	50	0.0E + 00	100
14	250	750	776.666667	1100	183.095083	100
15	1700	2950	3000	4450	731.681039	100
16	50	400	380	550	136.014705	100
17	150	550	565	1000	194.825298	100
18	50	100	116.666667	200	58.756965	100

dynamic constraint handling technology [74] are added to improve particles optimization' ability.

To the best of our knowledge, It is worth noting that this is the first study compared with [40] using the complete set of test problems since its publication in 2009.

5.1 Comparison with basic DE algorithm

In this section, in order to investigate the effect of the proposed mutation, EMDE algorithm is compared with basic DE algorithm that uses DE/rand/1/bin mutation scheme. The statistical results of the EMDE and Basic DE

Table 6 Number of FEs to achieve the fixed accuracy level ($f(x) - f(x^*)) < 0.01$), Successful rate (Ps) provided by Basic DE

Problem	Best	Median	Mean	Worst	Std	Ps
1	100	650	620	1100	283.976860	96.67
2	50	150	226.666667	650	200.772318	100
3	1400	2100	1996.153846	2400	267.286939	86.67
4	7350	8900	8856.666667	10900	893.401739	100
5	150	475	538.333333	950	238.053650	100
6	50	50	50	50	0.00E + 00	100
7	1800	3250	3553.333333	7000	1414.398745	86.67
8	7950	14975	16692.5	23950	4335.601549	66.67
9	850	1550	2165	2750	220.950143	100
10	250	1450	1958.333333	2950	656.805662	100
11	50	475	448.333333	950	296.672155	100
12	1000	2500	3117.857143	4750	1113.485025	46.67
13	50	50	50	50	0.0E + 00	100
14	100	800	773.333333	1350	275.660127	100
15	3350	4850	5768.333333	10150	1425.233395	100
16	300	500	511.666667	800	110.393757	100
17	200	750	753.333333	1100	212.104940	100
18	50	100	141.666667	450	102.623629	100

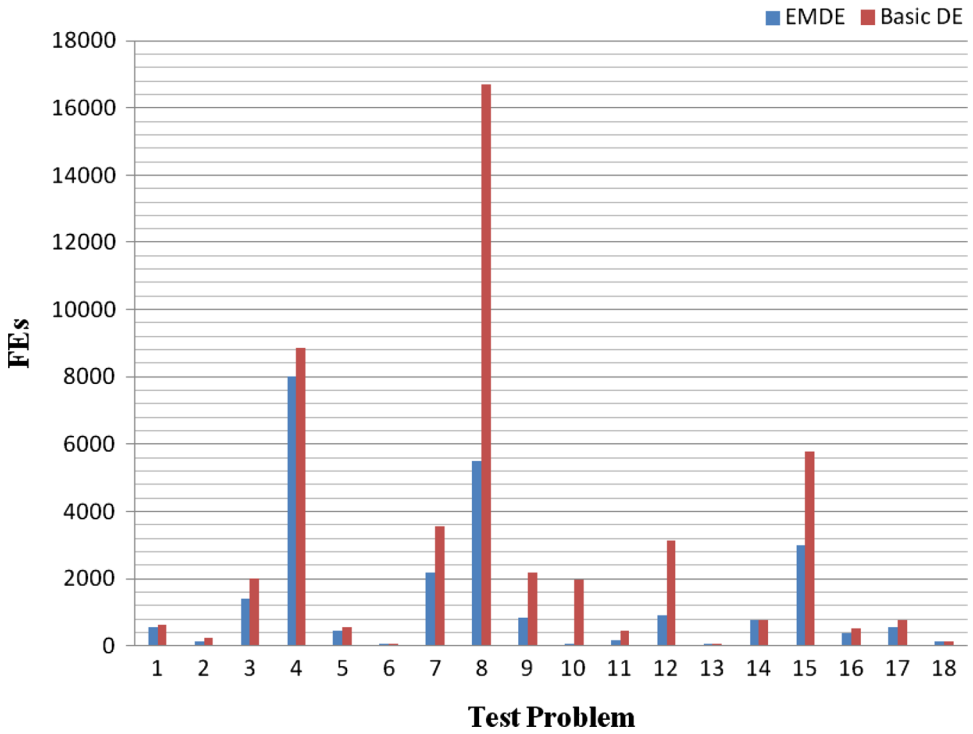

Fig. 5 the performance of EMDE and Basic DE algorithms in terms of mean FEs in all test functions

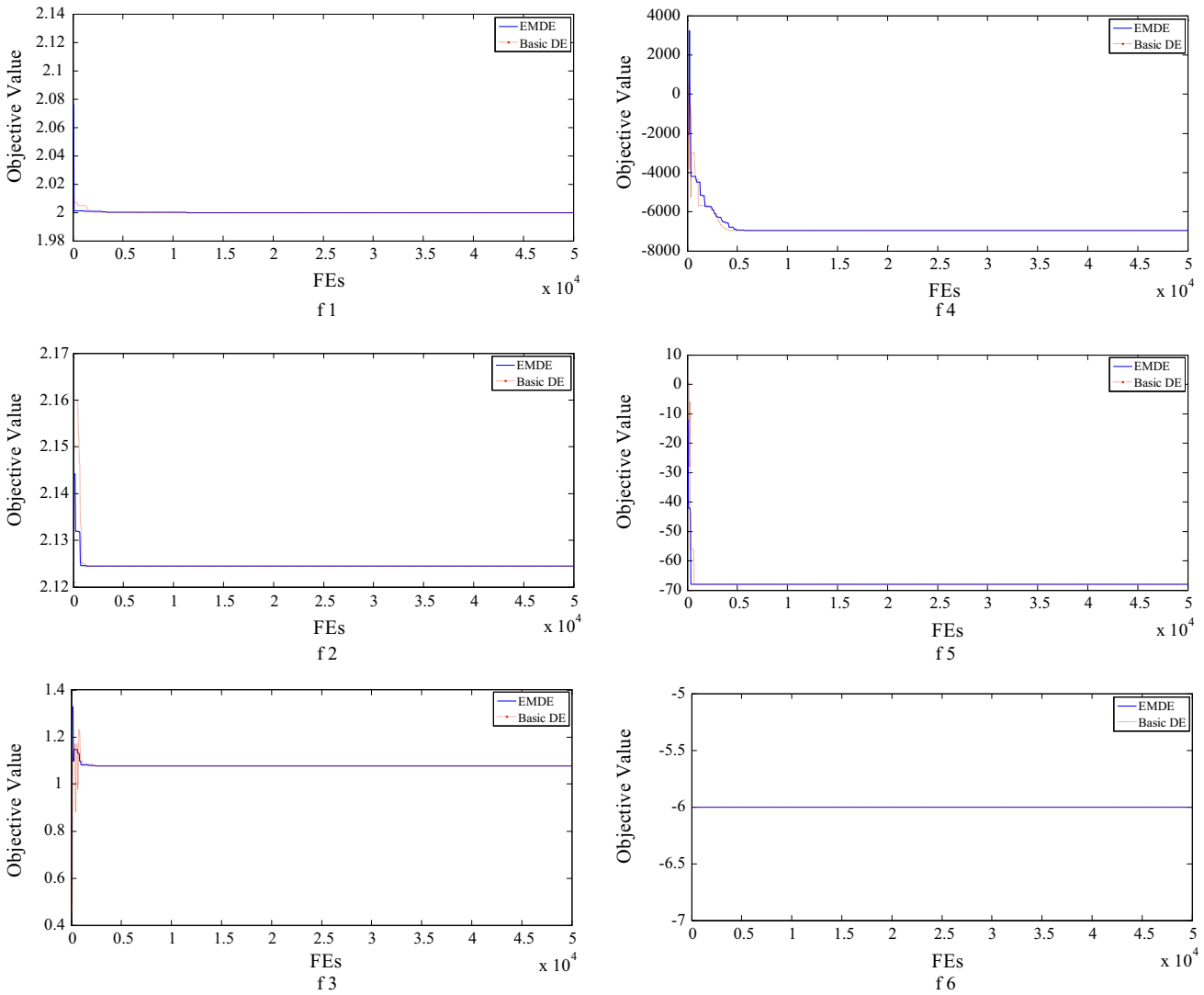

Fig. 6 Convergence graph (median curves) of EMDE and Basic DE on test functions f 1–f 18

on the benchmarks are summarized in Tables 1 and 2. It includes the known optimal solution or best solution found for each test problem and the obtained best, median, mean, worst values and the standard deviations. As shown in Table 1, EMDE is able to find the global optimal solution consistently over 50 runs with the exception of test function 8. With respect to this test function, although the optimal solutions are not consistently found, the best result achieved is very close to the global optimal solution which can be verified by the very small standard deviation. Additionally, it is able to provide the competitive results. Regarding basic DE, from Table 2, basic DE is able to find the global optimal solution consistently in the majority test functions over 30 runs with the exception of test functions 1, 7, 8, 12 and 16). EMDE and basic DE are also able to find new solution to test functions 10, 16 and 17 which is better than the best known solutions from [40]. Actually, EMDE was able to reach the new global solutions consistently for problems 10 and 16. The Better solutions obtained by EMDE and Basic DE than MI-LXPM [40] are listed in Tables 3 and 4, respectively. It is noteworthy to mentioning that problems 16 and 17 are 40 and 100 dimensions, respectively. Hence, it can be concluded that EMDE can globally optimize MINLPs with high dimensions.

Thus, from Table 3, it can be clearly observed that EMDE and basic DE find the global optimal value 1352439 in Problem 16. Actually, the known optimal value 1030361 is a local one. Also, a better objective function value 304153834 and 304139472 (the known one is 303062432)

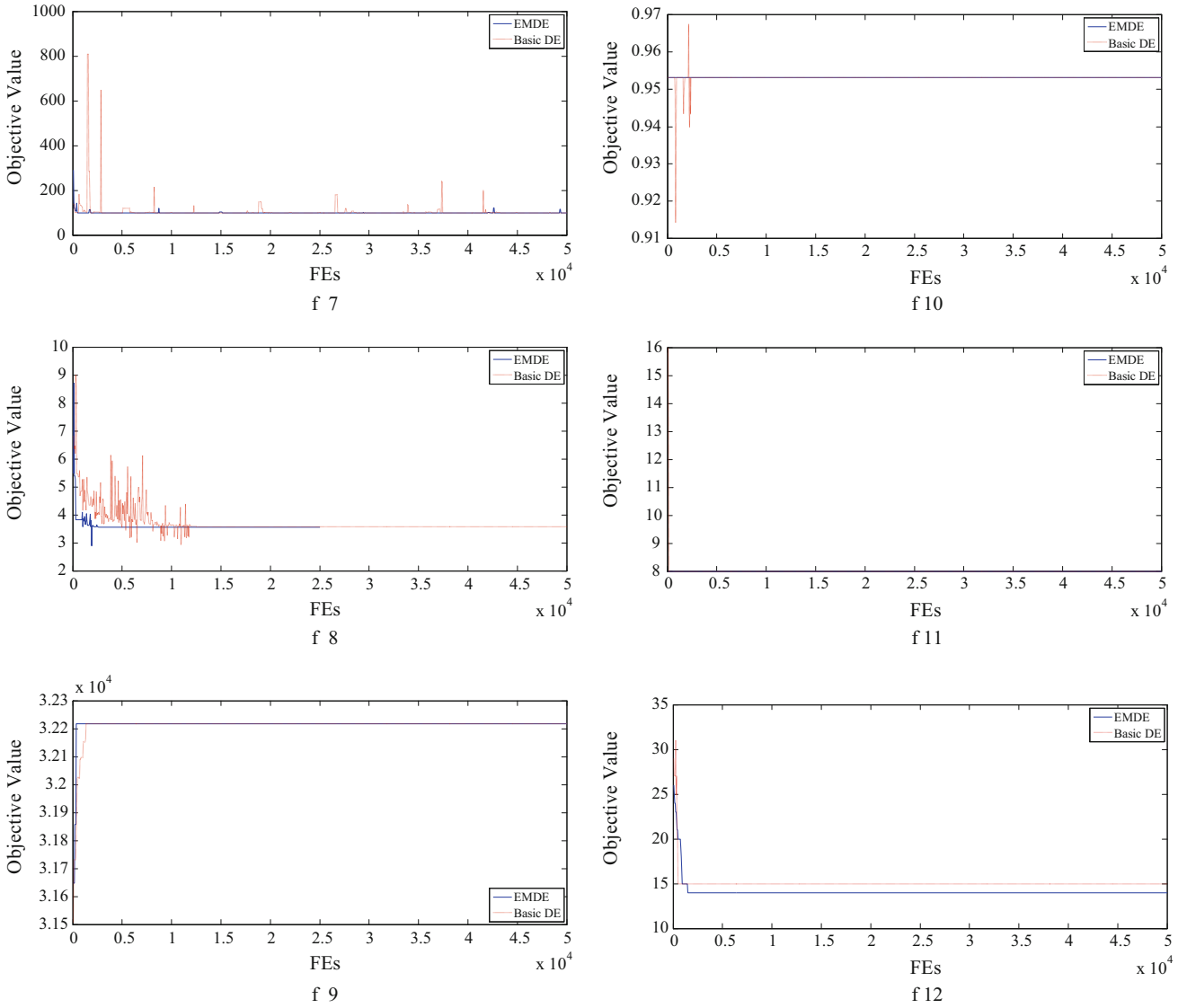

Fig. 6 continued

are found by EMDE and Basic DE, respectively, in Problem 17. However, EMDE produces better solutions than basic DE algorithm.

The computational results of EMDE and basic DE algorithm are measured by using the following two performance measures [53]:

- The success rate (SR), i.e. the proportion of convergence to the known optimal solution, which can be used to evaluate the stability and robustness of proposed algorithm.
- The mean number of fitness evaluations (FES), which can be used to assess the computational cost or efficiency of the proposed algorithm.

The statistical results about number of function evaluations (FEs) in the case of successful runs are provided including the best, median, mean, worst and standard deviation used by EMDE and basic DE algorithms in achieving the optimal solution in the case of the successful runs and the percentage of success are also listed in Tables 5 and 6, respectively. It can be obviously seen from Tables 5 and 6 that The success rate of EMDE algorithm reaches 100 % except problem 8 which is 90 %. However, the success rate of basic DE reaches 100 % with exception to problems 1,3,7,8 and 12. Besides, in terms of Number of FEs, Fig. 5 show the performance of EMDE and Basic DE algorithms in terms of Mean FEs for 18 test functions. From Fig. 5, it is clearly that mean FEs provided by EMDE

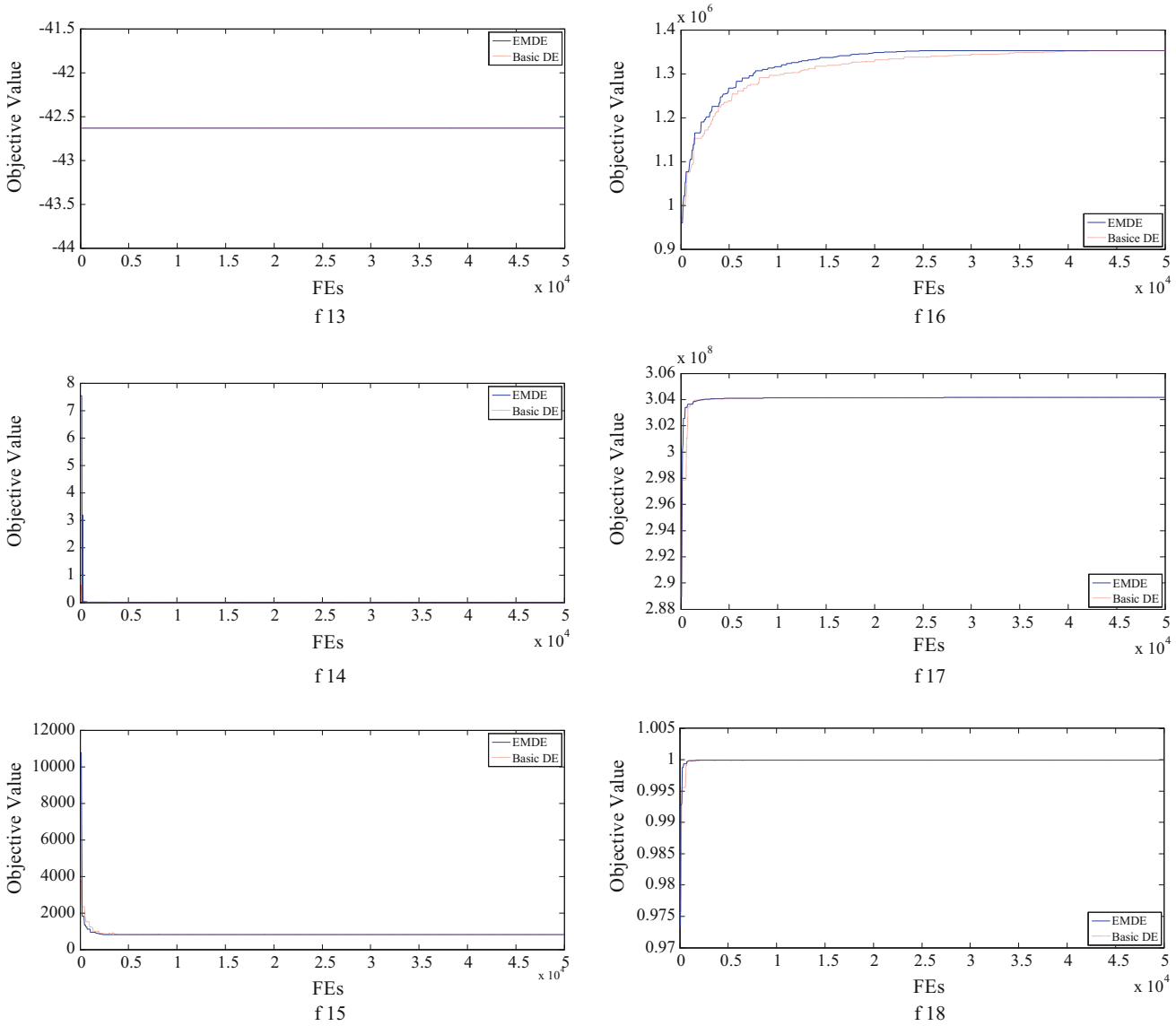

Fig. 6 continued

is better than mean FEs provided by Basic DE in almost test problems.

Thus, in order to measure the efficiency of EMDE in solving all test problems, the sum of the mean FEs provided by Basic DE is 48221.51099 while the sum of the mean FEs produced by EMDE is 25145.36714. Therefore, on average, the improvement percentage of EMDE in terms of FEs in comparison to Basic DE is 47.85 %. Therefore, EMDE is considered the most efficient with the smallest total mean (FEs) which prove the searching ability of the proposed triangular mutation in balancing both exploration capability and exploitation tendency and enhance the convergence speed of the algorithm. Additionally, Fig. 6 show the convergence graphs of f(x)) over FEs at the median run of EMDE and Basic DE algorithms for each test problem with 50,000 FEs. From Fig. 6, it can be obviously seen that a fast convergence of both EMDE and basic DE but it shows that EMDE algorithms converge to better or global solution faster than basic DE in all cases, for function 12 the basic DE is easily trapped into local optimum, and functions can jump out of local optima with EMDE to find the global optimum, indicating that EMDE has strong searching ability and its stability is very good. From the above analysis, it can be concluded that EMDE has a fast convergence speed for these 18 test functions. Accordingly, the main benefits of the new triangular mutation are the fast convergence speed and the extreme robustness which are the weak points of all evolutionary algorithms. Therefore, the proposed EMDE algorithm is proven to be an effective and powerful approach to solve

Table 7 Results obtained by using EMDE, MI-LXPM, RST2ANU, AXNUM and MIPDE algorithms

Problem	Dim	EMDE		MI-LXPM [40]		RST2ANU [40]		AXNUM [40]		MIPDE [51]	
		Mean FEs	ps	Mean FEs	ps	Mean FEs	ps	Mean FEs	ps	Mean FEs	ps
1	2	570	100	172	84	173	47	1728	86	9493	100
2	2	126.667	100	64	85	657	57	82	67	1683	100
3	3	1393.33	100	18608	43	221129	4	65303	35	9828	100
4	2	7993.33	100	10933	95	1489713	2	45228	82	10644	97
5	3	448.333	100	671	100	2673	75	13820	95	565	100
6	4	50	100	84	100	108	100	432	100	36	100
7	3	2192.85	100	7447	59	–	–	16077	45	1265	100
8	7	5504.16	90	3571	41	180859	15	1950	3	13310	95
9	5	850	100	100	100	189	100	4946	100	451	100
10	8	55	100	258	93	545	100	700	33	1375	100
11	5	173.333	100	171	100	2500	100	863	97	204	100
12	7	900	100	299979	71	6445	29	380115	19	6293	84
13	2	50	100	77	99	35	100	456	91	27	100
14	3	776.667	100	78	100	214	100	1444	100	–	–
15	5	3000	100	2437	92	3337	19	267177	9	4195	97
16	40	380	100	1075	100	1114	100	2950	100	–	–
17	100	565	100	600	100	2804	100	600	100	–	–
18	8	116.667	100	250	100	697	100	256	102	–	–
Total mean FEs	25,145.36714		346575		1913761		804127		59369		

constrained mixed integer nonlinear programming problems.

5.2 Comparison with other algorithms in the literature

To further verify the performance of EMDE, comparisons are carried out with aforementioned four competitive state-of-the-art evolutionary algorithms. Experimental results obtained by five algorithms are given in Table 7, where (ps) is success rate representing percentage of the successful runs to total runs for the algorithm, and (mean FEs) represents average number of objective function evaluations used by the algorithm in successful runs. The results provided by MI-LXPM, RST2ANU and AXNUM approaches were directly taken from Ref. [40] while for MIPDE was directly taken from Ref. [51]. Note that MIPDE algorithm solved only 14 benchmark problems with exception to problems 14, 16,17 and 18. According to the results shown in Table 7, it can be obviously seen that that EMDE algorithm can obtain the optimal solution for different kinds of linear or nonlinear mixed integer programming problems with a success rate of 100 % for all test function except problem 8 with 90 %, which verifies the stability and robustness of EMDE again, in fact, with no need for parameters to be fine-tuned. Furthermore, regarding mean FEs, it can be observed that EMDE produces 11, 14, 16 and 10 significantly better and 7, 4,2 and 4 slightly worse than MI-LXPM, RST2ANU, AXNUM and MIPDE algorithms for test problems, respectively. Thus, in order to measure the efficiency of EMDE in solving all test problems, the sum of the mean FEs provided by MI-LXPM, RST2ANU, AXNUM and MIPDE algorithms are 346575, 1913761, 804127 and 59369 while the sum of the mean FEs produced by EMDE is 25145.36714. Therefore, EMDE is considered the most efficient with the remarkable smallest total mean (FEs). Figure 7 show the performance of EMDE and MI-LXPM, RST2ANU, AXNUM and MIPDE algorithms in terms of total Mean FEs for all test functions. All in all, EMDE is superior to all compared EAs algorithms in terms of both measures success rate and efficiency.

6 Conclusion

In this paper, an efficient modified Differential Evolution algorithm EMDE is proposed for solution of constrained, integer and mixed integer optimization problems which are considered difficult optimization problems. In this algorithm new triangular mutation rule based on the convex combination vector of the triplet defined by the three randomly chosen vectors and the difference vectors between the best,better and the worst individuals among the three randomly selected vectors is introduced. The proposed novel approach to mutation operator is shown to enhance

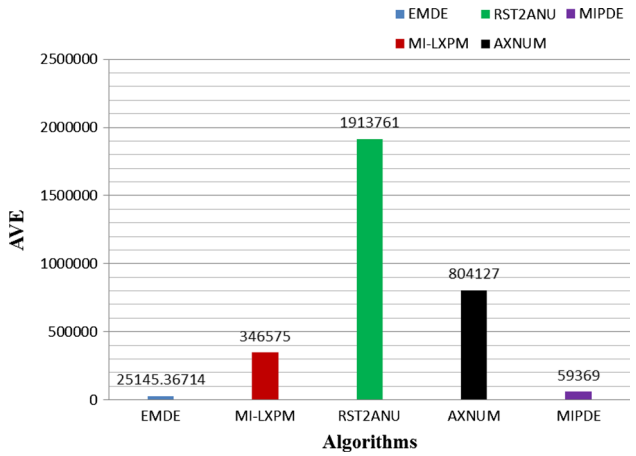

Fig. 7 the performance of EMDE, MI-LXPM, RST2ANU, AXNUM a and MIPDE algorithms in terms of total Mean of FEs

the global and local search capabilities and to increase the convergence speed of the new algorithm compared with basic DE. EMDE uses "parameter free" Deb's constraint handling technique based on feasibility and the sum of constraints violations. Integer variables are directly searched in the integer space by using round operator, to ensure that the vector is controlled in the integer space in the evolution procedure of the EMDE. The performance of the proposed EMDE is compared with basic DE, MI-LXPM, RST2ANU, AXNUM and MIPDE algorithms on a set of 18 test problems. Our results show that the proposed EMDE algorithm outperforms remarkably other compared algorithms in terms of stability, robustness and efficiency. Accordingly, Firstly, Current research effort focuses on how to control the crossover rate by self-adaptive mechanism. Additionally, future research will investigate the performance of the EMDE algorithm in solving unconstrained and constrained multi-objective optimization problems. In future the proposed EMDE algorithm may be compared with other stochastic approaches like harmony search and particle swarm optimization, artificial bee colony, bees algorithm and ant colony optimization. Additionally, it is also worth trying to apply the present approach to the solution of real-world optimization problems.

Appendix

Corrections on problems 6, 16 and 17.

Based on self check, there are few differences in problems 6, 16 and 17 in Ref. [40] from the original documents. Thus, the corrections on these problems are as follows.

Problem	Objective function or constraints	Ref. [40]	Correction	Original references
6	Constraint	x_4	$3x_4$	[32]
16	Constraint 1	$9x_1$	$8x_1$	[75]
17	Objective function	$15x_{35}^{35}, 3x_{94}^{2}$	$15x_{35}^{3}, 3x_{94}^{2}$	[75]

References

1. Mohamed AW, Sabry HZ (2012) Constrained optimization based on modified differential evolution algorithm. Inf Sci 194:171–208
2. Costa L, Oliveira P (2001) Evolutionary algorithms approach to the solution of mixed non-linear programming. Comput Chem Eng 25:257–266
3. Lin YC, Hwang KS, Wang FS (2004) A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems. Comput Math Appl 47:1295–1307
4. Sandgren E (1990) Nonlinear integer and discrete programming in mechanical design, ASME Y. Mech Des 112:223–229
5. Dua V, Pistikopoulos EN (1998) Optimization techniques for process synthesis and material design under uncertainty. Chem Eng Res Des 76(3):408–416
6. Catalão JPS, Pousinho HMI, Mendes VMF (2010) Mixed-integer nonlinear approach for the optimal scheduling of a head-dependent hydro chain. Electr Power Syst Res 80(8):935–942
7. Garroppo RG, Giordano S, Nencioni G, Scutellà MG (2013) Mixed integer non-linear programming models for green network design. Comput Oper Res 40(1):273–281
8. Maldonado S, Pérez J, Weber R, Labbé M (2014) Feature selection for support vector machines via mixed integer linear programming. Inf Sci 279(20):163–175
9. Çetinkaya C, Karaoglan I, Gökçen H (2013) Two-stage vehicle routing problem with arc time windows: a mixed integer programming formulation and a heuristic approach. Eur J Oper Res 230(3):539–550
10. Liu P, Whitaker A, Pistikopoulos EN, Li Z (2011) A mixed-integer programming approach to strategic planning of chemical centres: a case study in the UK. Comput Chem Eng 35(8):1359–1373
11. Xu G, Papageorgiou LG (2009) A mixed integer optimisation model for data classification. Comput Ind Eng 56(4):1205–1215
12. Grossmann IE, Sahinidis NV (eds) (2002) Special issue on mixed-integer programming and its application to engineering, Part I, Optim. Eng., Kluwer Academic Publishers, Netherlands, vol. 3 (4)
13. Grossmann IE, Sahinidis NV (eds) (2002) Special issue on mixed-integer programming and its application to engineering, Part II, Optim. Eng., Kluwer Academic Publishers, Netherlands, vol. 4 (1)
14. Hsieh Y-C et al (2015) Solving nonlinear constrained optimization problems: an immune evolutionary based two-phase approach. Appl Math Model. doi:10.1016/j.apm.2014.12.019
15. Ng CK, Zhang LS, Li D, Tian WW (2005) Discrete filled function method for discrete global optimization. Comput Optim Appl 31(1):87–115
16. Gupta OK, Ravindran A (1985) Branch and bound experiments in convex nonlinear integer programming. Manag Sci 31(12):1533–1546

17. Borchers B, Mitchell JE (1994) An improved branch and bound algorithm for mixed integer nonlinear programming. Comput Oper Res 21:359–367
18. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Theory Appl 10:237–260
19. DuranMA GI (1986) An outer approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36(3):307–339
20. Fletcher R, Leyffer S (1994) Solving mixed-integer programs by outer approximation. Math Program 66(1–3):327–349
21. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16(10–11):937–947
22. Westerlund T, Pettersson F (1995) A cutting plane method for solving convex MINLP problems. Comput Chem Eng 19:S131–S136
23. Lee S, Grossmann IE (2000) New algorithms for nonlinear generalized disjunctive programming. Comput Chem Eng 24:2125–2142
24. Bonami P, Biegler LT, Conn AR, Cornuéjols G, Grossmann IE, Laird CD, Lee J, Lodi A, Margot F, Sawaya N, Wächter A (2008) An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optim 5:186–204
25. Abhishek K, Leyffer S, Linderoth JT (2010) FilMINT: an outer-approximation-based solver for nonlinear mixed integer programs. Inf J Comput 22:555–567
26. Belotti P, Kirches C, Leyffer S, Linderoth J, Luedtke J, Mahajan A (2013) Mixed-integer nonlinear optimization. Acta Num. 22:1–131
27. Liberti L, Cafieri S, Tarissan F (2009) Reformulations in mathematical programming: a computational approach. In: Abraham A, Hassanien AE, Siarry P (eds) Foundations on computational intelligence, studies in computational intelligence, vol 203. Springer, New York, pp 153–234
28. D'Ambrosio C, Lodi A (2011) Mixed integer nonlinear programming tools: a practical overview. In: 4OR 9, No. 4, 2011, pp. 329–349 (cit. on p. 13)
29. Trespalacios F, Grossmann IE (2014) Review of mixed-integer nonlinear and generalized disjunctive programming Methods. Chem Ing Tech 86:991–1012
30. Burer S, Letchford AN (2012) Non-convex mixed-integer nonlinear programming: a survey. Surveys Oper Res Manag Sci 17(2):97–106
31. Grossmann IE (2002) Review of non-linear mixed integer and disjunctive programming techniques. Optim Eng 3:227–252
32. Cardoso MF, Salcedo RL, Feyo de Azevedo S, Barbosa D (1997) A simulated annealing approach to the solution of minlp problems. Comput Chem Eng 21(12):1349–1364
33. Rosen SL, Harmonosky CM (2005) An improved simulated annealing simulation optimization method for discrete parameter stochastic systems. Comput Oper Res 32:343–358
34. Glover F (2006) Parametric tabu-search for mixed integer programs. Comput Oper Res 33(9):2449–2494
35. Hua Z, Huang F (2006) A variable-grouping based genetic algorithm for large-scale integer programming. Inf Sci 176(19):2869–2885
36. Kesen SE, Das SK, Güngör Z (2010) A genetic algorithm based heuristic for scheduling of virtual manufacturing cells (VMCs). Comput Oper Res 37(6):1148–1156
37. Turkkan N (2003) Discrete optimization of structures using a floating-point genetic algorithm. In: Annual conference of the Canadian society for civil engineering, Moncton, Canada
38. Yokota T, Gen M, Li YX (1996) Genetic algorithm for non-linear mixed integer programming problems and its applications. Comput Ind Eng 30:905–917
39. Wasanapradit T, Mukdasanit N, Chaiyaratana N, Srinophakun T (2011) Solving mixed-integer nonlinear programming problems using improved genetic algorithms. Korean J Chem Eng 28(1):32–40
40. Deep K, Singh KP, Kansal ML, Mohan C (2009) A real coded genetic algorithm for solving integer and mixed integer optimization problems. Appl Math Comput 212(2):505–518
41. Cai J, Thierauf G (1996) Evolution strategies for solving discrete optimization problems. Adv Eng Softw 25:177–183
42. Costa L, Oliveira P Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. Comput Chem Eng, 25(2–3):257-266
43. Cao YJ, Jiang L, Wu QH (2000) An evolutionary programming approach to mixed-variable optimization problems. Appl Math Model 24:931–942
44. Mohan C, Nguyen HT (1999) A controlled random search technique incorporating the simulating annealing concept for solving integer and mixed integer global optimization problems. Comput Opti Appl 14:103–132
45. Woon SF, Rehbock V (2010) A critical review of discrete filled function methods in solving nonlinear discrete optimization problems. Appl Math Comput 217(1):25–41
46. Yongjian Y, Yumei L (2007) A new discrete filled function algorithm for discrete global optimization. J Comput Appl Math 202(2):280–291
47. Socha K (2004) ACO for continuous and mixed-variable optimization. Ant colony, optimization and swarm intelligence. Springer, Berlin, pp 25–36
48. Schlüter M, Egea JA, Banga JR (2009) Extended ant colony optimization for non-convex mixed integer nonlinear programming. Comput Oper Res 36(7):2217–2229
49. Yiqing L, Xigang Y, Yongjian L (2007) An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints. Comput Chem Eng 31(3):153–162
50. Yue T, Guan-zheng T, Shu-guang D (2014) Hybrid particle swarm optimization with chaotic search for solving integer and mixed integer programming problems. J Central South Univ 21:2731–2742
51. Gao Y, Ren Z, Gao Y (2011) Modified differential evolution algorithm of constrained nonlinear mixed integer programming problems. Inf Technol J 10(11):2068–2075
52. Lin YC, Hwang KS, Wang FS (2004) A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems. Comput Math Appl 47(8–9):1295–1307
53. Li H, Zhang L (2014) A discrete hybrid differential evolution algorithm for solving integer programming problems. Eng Optim 46(9):1238–1268
54. Mohamed AW (2015) An improved differential evolution algorithm with triangular mutation for global numerical optimization. Comput Ind Eng 85:359–375
55. Storn R, Price K (1995) Differential evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report TR-95-012, ICSI http://http.icsi.berkeley.edu/~storn/litera.html
56. Storn R, Price K (1997) Differential Evolution- a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
57. Engelbrecht AP (2005) Fundamentals of computational swarm intelligence. Wiley, Hoboken
58. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31
59. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186:311–338
60. Venkatraman S, Yen GG (2005) A generic framework for constrained optimization using genetic algorithms. IEEE Trans Evol Comput 9(4):424–435
61. Storn R, Price K (1997) Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359

62. Fan HY, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Global Optim 27(1):105–129
63. Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization, 1st edn. Springer, New York
64. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
65. Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation Strategies. Appl Soft Comput 11(2):1679–1696
66. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66
67. Zhang JQ, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
68. Feoktistov V (2006) Differential evolution. Berlin, Germany, Springer-verlag, In Search of Solutions
69. Mezura-Montes E, Coello CAC (2005) A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Trans Evol Comput 9(1):1–17
70. Dong N, Wang Y (2014) A memetic differential evolution algorithm based on dynamic preference for constrained optimization problems. J Appl Math, Article ID 606019, p 15. doi:10.1155/2014/606019
71. Lampinen J, Zelinka I (1999) Mixed integer-discrete-continuous optimization by differential evolution, Part 1: the optimization method. In: Ošmera P (ed.) (1999). Proceedings of Mendel 99, 5th international Mendel conference on soft computing, Brno, Czech Republic
72. Omran MGH, Engelbrecht AP (2007) Differential evolution for integer programming problems, IEEE congress on evolutionary computation, pp. 2237–2242
73. Li Y, Gen M (1996) Nonlinear mixed integer programming problems using genetic algorithm and penalty function. IEEE Int Conf Syst Man Cybernet 4:2677–2682
74. Lu H, Chen W (2008) Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. J Global Optim 41:427–445
75. Conley W (1984) Computer optimization techniques. Petrocelli Books, Princeton