Information Sciences 466 (2018) 170–188

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A hybrid differential evolution algorithm for mixed-variable optimization problems

Ying Lin[a], Yu Liu[b], Wei-Neng Chen[c,*], Jun Zhang[d]

[a] Department of Psychology, Sun Yat-sen University, Guangzhou, China
[b] Department of Computer Science, Sun Yat-sen University, Guangzhou, China
[c] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[d] Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, Guangzhou, China

ARTICLE INFO

Article history:
Received 27 November 2016
Revised 9 July 2018
Accepted 12 July 2018
Available online 21 July 2018

Keywords:
Mixed-variable optimization (MVOP)
Differential evolution (DE)
Set theory

ABSTRACT

Mixed-variable optimization problems (MVOPs) that involve continuous and discrete decision variables widely exist in industrial and scientific domains. However, how to solve MVOPs efficiently remains an open issue because the fact that continuous and discrete variables present different spatial distribution features posts a great challenge to algorithmic design. In this paper, a hybrid differential evolution (DE) framework is proposed for MVOPs. The proposed framework, namely DE_{MV}, hybridizes the original DE and the set-based DE for evolving continuous and discrete variables, respectively. The two DEs are selected for hybridization because algorithmic analysis and experimental studies show that they share the same search mechanism. The compatibility and consistency of the two DEs is the key for enabling DE_{MV} to coevolve different types of decision variables efficiently. Experiments are conducted on a set of artificial MVOPs converted from continuous benchmark functions and real-world engineering problems with mixed variables. Experimental results and comparisons with other representative algorithms show that DE_{MV} is effective and efficient.

1. Introduction

Given an optimization problem, the quantities that need to be determined in order to achieve the optimum are known as decision variables. In optimization theory, depending on whether decision variables are solely continuous or discrete, problems are categorized as continuous optimization problems (CnOPs) and discrete optimization problems (DOPs). Great research efforts have been devoted to developing efficient algorithms for CnOPs and DOPs, respectively. As continuous and discrete variables present distinct spatial distribution features, most algorithms designed for CnOPs are inefficient or even inapplicable for DOPs, and vice versa. Many real-world problems, however, are mixed-variable optimization problems (MVOPs) such that their decision variables are partially continuous and partially discrete. Therefore, there is a strong desire for developing an efficient optimization algorithm that can handle continuous and discrete variables simultaneously.

Evolutionary algorithms (EAs) are a new family of optimization algorithms inspired from the evolutionary procedure or intelligent behaviors of organisms [3]. Different from traditional optimization algorithms, EAs do not rely on the mathematical properties (e.g., continuity or differentiability) of optimization problems for finding the optimal or near-optimal

* Corresponding author.
E-mail address: cwnraul634@aliyun.com (W.-N. Chen).

https://doi.org/10.1016/j.ins.2018.07.035
0020-0255/

solution(s). Such a feature makes EAs especially suitable for optimization problems whose mathematical properties are undesired or unclear. As a result, EAs have been applied to optimization problems in different domains and obtained promising results [15,18,22,23,36,41,43,49,54]. However, like traditional optimization algorithms, most EAs are designed specifically for CnOPs or DOPs. For example, without extension, differential evolution (DE) [39] and particle swarm optimization (PSO) [24] are only applicable to CnOPs, while ant colony optimization (ACO) [12] is only applicable to DOPs. Developing efficient EAs for MVOPs thus becomes a research topic of theoretical and practical importance. In the literature, the few EAs applicable to MVOPs can be classified into the following three categories.

The most distinct feature of the first category of EAs for MVOPs lies in discretization. That is, the search ranges of continuous variables are discretized so that they can be optimized in the same way as discrete variables. A representative is the binary-coded genetic algorithm (GA) [30], in which the value of each continuous variable is encoded as a binary string. By doing so, the evolutionary operators for discrete variables can also operate on continuous variables. However, since the precision of continuous variables is restricted by the length of binary strings, binary-coded GA suffers from limited solution accuracy.

Compared with the first category, the second category of EAs address MVOPs in an opposite way [16,44]. More specifically, these algorithms relax discrete variables into continuous ones and optimize them with evolutionary operators designed for CnOPs. The continuous value of each discrete variable is rounded off before solution evaluation. However, the relaxation method is not always effective. Discrete variables can be further divided into two types: ordinal and categorical. The set of available values for an ordinal variable have a clear order. On the opposite, there is no ordering relation among the available values of a categorical variable. The relaxation method usually works for ordinal variables, but not for categorical ones, because the available values of a categorical variable are not numerically related and applying arithmetic operators on them does not make any sense. As such, the second category of EAs are restricted to MVOPs with continuous and ordinal variables.

As for the third category of EAs, different evolutionary operators are employed to handle continuous and discrete variables separately [9]–[11,13,27,46]. The evolutionary operators do not necessarily come from the same algorithm. New operators can also be introduced to handle the type(s) of decision variables that cannot be handled originally. Besides, a local search procedure is often embedded into these algorithms for improving the search efficiency. Although the third category of EAs no longer have limitation on the types of decision variables, there is no guarantee that the search mechanisms behind the evolutionary operators for different types of decision variables are consistent and compatible. Such inconsistency and incompatibility hinder the coevolution of continuous and discrete variables. Therefore, the efficiency of these algorithms remains unsatisfying.

From the above, it can be known that using specialized evolutionary operators to optimize continuous and discrete variables separately is so far the most promising way for solving MVOPs. The key to improve the overall search efficiency lies in the consistency and compatibility of the search mechanisms behind different evolutionary operators. Based on this idea, this paper proposes a hybrid DE framework for MVOPs (DE_{MV}), which hybridizes the original DE [39] for CnOPs and the set-based DE [29] for DOPs. The two DEs are selected for hybridization because they share the same search mechanism, as will be shown by the algorithmic analysis and the experimental studies in this paper. In DE_{MV}, the continuous variables of an MVOP are encoded, initialized, and evolved through the original DE, whilst the discrete variables are encoded, initialized, and evolved through the set-based DE. The separate evolutionary procedures of continuous and discrete variables merge at a common selection operator. Then the new population generated from selection repeats the above evolution cycle until the termination criterion is met. The proposed DE_{MV} is applied to solve artificial MVOPs converted from continuous benchmark functions and real-world problems with mixed variables. Experimental results and comparisons with representative algorithms show that DE_{MV} is effective and efficient.

The remainder of this paper is organized as follows. Section 2 formally defines MVOPs. Section 3 briefly reviews the original DE for CnOPs and the set-based DE for DOPs, based on which the proposed DE_{MV} is introduced. Experimental results of DE_{MV} and comparisons with representative algorithms are presented in Section 4. Section 5 offers a series of additional experiments to show that the set-based DE has the same search mechanism as the original DE and investigate the effects of the evolutionary operators and the parameters on DE_{MV}. Section 6 draws a conclusion to the whole paper.

2. Definition of MVOPs

In general, an optimization problem can be defined by a triple (S, f, Ω), where S is the search space, f: $S \rightarrow \mathrm{R}$ is the objective function, and Ω is the set of constraints. Given that f has D decision variables, $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$ is a solution to the problem if each decision variable x_i $(i = 1, 2, \ldots, D)$ is in the legal range defined by S. If $\boldsymbol{x}$ satisfies all the constraints in Ω, it is a feasible solution. The goal of the optimization problem is to find a feasible solution $\boldsymbol{x}^*$ from the search space S such that no other solutions can achieve an objective value better than $f(\boldsymbol{x}^*)$.

In some cases, all the decision variables are defined in the continuous domain. The search space S is a continuous space of D dimensions. The optimization problem is thus a continuous optimization problem (CnOP). In some other cases, the decision variables can be of other types, e.g., a binary number, an integer, an element selected from a finite candidate set, or even a set. Then the optimization problem is a discrete optimization problem (DOP, also known as combinatorial optimization problems). Variables in DOPs can be further divided into two types, ordinal and categorical. An ordinal variable is a variable whose values have an explicit order. For example, a variable that represents the age of examinees is an ordinal

variable because it is defined on a set of integers with a clear order. In contrast, a categorical variable is a variable whose values do not have an explicit order. For example, in a binary knapsack problem (BKP), each variable is a categorical variable as it uses one or zero to indicate whether or not to select the related item into the knapsack. Other than pure CnOPs and DOPs, some problems have continuous and discrete decision variables. Such problems are categorized as mixed-variables optimization problems (MVOPs).

Different types of variables have different characteristics and need to be handled with different optimization methods. Ordinal variables can be relaxed into continuous variables and optimized with continuous optimization algorithms. However, for categorical variables, only discrete optimization algorithms specialized in DOPs are applicable. So far, most optimization algorithms are designed for pure CnOPs or pure DOPs. Their effectiveness and efficiency are significantly impaired on MVOPs.

3. Hybrid DE for MVOPs

In this section, the continuous DE and the set-based DE are reviewed at first. Then the proposed algorithm DE_{MV} that hybridizes the above two DEs for MVOPs is introduced in detail. The reasons for hybridizing the above two DEs are also discussed.

3.1. DE for CnOPs

Differential evolution (DE), proposed by Storn and Price in 1990s, is a simple yet efficient meta-heuristic designed for global optimization of CnOPs [39]. As a population-based stochastic search method, DE uses evolutionary operators (i.e., mutation, crossover, and selection) at each generation to evolve the population towards the optimum. In the following subsections, the classic DE and its notable variants are reviewed briefly.

3.1.1. Classic DE

Let X denote the population of DE, in which each individual $\boldsymbol{x}_i = (x_i^1, x_i^2, ..., x_i^D)$ denotes a solution to the problem, x_i^j is the j-th decision variable, $i = 1, 2, ..., NP$, $j = 1, 2, ..., D$, NP is the population size, and D is the number of decision variables. The population X is initialized by randomly sampling the search space S. Then it enters the evolutionary cycle. In each generation, the following three operators are used to evolve X:

- *Mutation:* For each individual $\boldsymbol{x}_i$, a mutant vector $\boldsymbol{v}_i = (v_i^1, v_i^2, ..., v_i^D)$ is generated as

$$v_i^j = x_{r1}^j + F \cdot (x_{r2}^j - x_{r3}^j) \ , j = 1, 2, \ldots, D, \tag{1}$$

where $r1, r2, r3 \in \{1, 2, ..., NP\}$ are the indices of three randomly selected individuals and $F \in [0,1]$ is the predefined scale factor. Note that $r1$, $r2$, and $r3$ must be mutually exclusive. $\boldsymbol{x}_{r1}$ is often referred to as the base vector.

- *Crossover:* The *crossover* operator is meant for increasing the diversity of the perturbed solution. For each individual $\boldsymbol{x}_i$, a trial vector $\boldsymbol{u}_i = (u_i^1, u_i^2, \ldots u_i^D)$ is generated by combining $\boldsymbol{x}_i$ with the mutant vector $\boldsymbol{v}_i$:

$$u_i^j = \begin{cases} v_i^j, & \text{if } r < CR \text{ or } j = j_{\text{rand}} \\ x_i^j, & \text{otherwise} \end{cases}, j = 1, 2, \ldots, D, \tag{2}$$

where $r \in (0,1)$ is a random number, $j_{\text{rand}} \in \{1, 2, ..., D\}$ is a randomly selected dimension, and $CR \in [0,1]$ is the crossover probability that controls the extent to which $\boldsymbol{u}_i$ learns from $\boldsymbol{v}_i$. If $\boldsymbol{u}_i$ turns out to be an infeasible solution, repair strategies can be adopted to convert it into a feasible one. Then $\boldsymbol{u}_i$ is evaluated with the objective function f.

- *Selection:* For every pair of $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$, the selection operator chooses the one with a better objective value to form the population of the next generation, i.e.,

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{u}_i, & \text{if } f(\boldsymbol{u}_i) \succ f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i, & \text{otherwise} \end{cases}, i = 1, 2, \ldots, NP, \tag{3}$$

where '$\succ$' indicates a better relation. By doing so, the selection operator guarantees that the better solution will enter the next generation.

The classic DE iterates the above three operators until the predefined termination criterion (e.g., the maximum number of function evaluations) is satisfied. Then the best individual in the population, i.e., the best-so-far solution $\boldsymbol{x}_{\text{bsf}}$, is reported as the result. The pseudo-code of the classic DE is shown in Algorithm 1.

3.1.2. Notable variants of DE

Many DE variants have been proposed (for surveys please refer to [2,7,8,14], and [34]). To distinguish DE variants that use different evolutionary operators, the following notations are introduced:

$$DE/a/b/c. \tag{4}$$

Algorithm 1 x_{bsf} = Classic_DE (*NP*, *F*, *CR*).

Input:
NP: population size
F: scale factor
CR: crossover probability
Output:
x_{bsf}: best-so-far solution

BEGIN
Initialize a population $X=\{x_1, x_2,..., x_{NP}\}$ by randomly sampling the search space *S* for *NP* times
WHILE the termination criterion is not met
  FOR each individual x_i ($i=1, 2,..., NP$)
    Mutation: generate a mutant vector v_i according to Eq. (1)
    Crossover: generate a trial vector u_i by combining v_i and x_i according to Eq. (2)
  END FOR
  FOR each individual x_i ($i=1, 2,..., NP$)
    Selection: update x_i with u_i according to Eq. (3)
  END FOR
END WHILE
Return the best individual x_{bsf} in the population as the result
END

Traveling Salesman Problem (TSP)

An undirected graph $G=(V, A)$ is given, where V is the set of nodes and A is the set of arcs. $D=|V|$ is the number of nodes. Each arc $(j,k)\in A$ is assigned a length d_{jk}. The goal of TSP is to find a Hamiltonian circuit with the minimum length.

Binary Knapsack Problem (BKP)

A set of D items and a knapsack with a weight limit b are given. Each item j has a value v_j and a weight w_j ($j=1, 2, \ldots, D$). The goal of BKP is to find a subset of items such that the total value is maximized and the total weight is less than or equal to the weight limit b.

Fig. 1. Definitions of TSP and BKP.

Specifically, *a* indicates the base vector in the mutation operator, which can be "rand", "best", or "current-to-best", etc. "rand" means that the base vector is randomly chosen from the population. "best" means that the base vector is the best individual in the current population. "current-to-best" means that the base vector is an arithmetic combination of the best individual and the individual currently being mutated. *b* is the number of differential vectors used in mutation. Each differential vector is obtained by subtracting one solution from the other. *c* denotes the strategy of crossover, which can be "binomial" or "exponential". "binomial", or for short "bin", is the crossover method used in the classic DE [i.e., Eq. (2)]. The "exponential" crossover is performed as follows:

$$u_i^j = \begin{cases} v_i^j, & \text{if } j \in ST \text{ and } r < CR \\ x_i^j, & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, D, \tag{5}$$

where $ST \subset \{1, 2,..., D\}$ is a set of consecutive dimensions.

Following the above notations, the aforementioned classic DE is written as DE/rand/1/bin. Other notable variants include DE/rand/2/bin, DE/best/1/exp, and DE/current-to-best/1/exp, etc. Besides using new evolutionary operators, other variants enhance the performance of DE in CnOPs through adaptation strategies [14,41,53], hybridization techniques [19,46–48,50], sampling mechanisms based on statistics evolution [20,31–33], and parallel or distributed computational models [6,17,45,49,52]. Considering the simplicity and the relatively robust performance of DE/rand/1/bin, it is adopted in this paper to exemplify the idea and the procedure of the proposed DE_{MV}. Note that other DE variants can serve as an alternative of DE/rand/1/bin to seek further enhancement on the performance of DE_{MV}.

3.2. Set-based DE for DOPs

DE has been found a promising optimization algorithm since its proposal. However, as the operators in DE are all defined in the continuous space, it is difficult to extend DE for DOPs, especially DOPs involving categorical decision variables. Although several discrete versions of DE have been proposed [4,35,38,42,49], they were designed for specific problems and cannot be fully generalized to common scenarios. Following the idea of redefining arithmetic operators in the discrete space through the set theory [5,21,28,51], Liu et al. [29] proposed a set-based DE for DOPs. Except for using the set theory to redefine the solution representation and the evolutionary operators, the set-based DE shares the evolutionary strategy with the original DE. Experimental studies have showed that the set-based DE can achieve satisfying results on different DOPs [29]. Encouraged by this, the mixed-variable DE proposed in this paper employs the set-based DE for optimization of discrete variables. A brief review of the set-based DE is given below. To facilitate understanding, the traveling salesman problem

(TSP) and the binary knapsack problem (BKP), as typical permutation-based and subset-based DOPs, are taken as examples. Detailed definitions of TSP and BKP are given in Fig. 1.

Given a DOP with D decision variables, the set-based DE maps the search space to a universal set $\boldsymbol{E}$ of D dimensions. Each dimension E^j is a subset of $\boldsymbol{E}$ that contains all the candidate elements for the j-th variable, i.e., $\boldsymbol{E}=E^1 \cup E^2 \cup \ldots \cup E^D$. Each individual is represented as $\boldsymbol{x}_i = x_i^1 \cup x_i^2 \cup \ldots \cup x_i^D$ ($i=1, 2,\ldots, NP$), where each dimension x_i^j is a subset of E^j that contains one or more elements for composing a complete solution. For example, in TSP, the universal set $\boldsymbol{E}$ is the arc set $\boldsymbol{A}$, and each dimension E^j is the set of arcs connecting with node j. Each dimension x_i^j of an individual $\boldsymbol{x}_i$ is a subset of E^j that contains exactly two arcs. In BKP, each dimension of the universal set $\boldsymbol{E}$ is defined as $E^j=\{(j,0), (j,1)\}$, where $(j,1)$ denotes the decision that selects item j and $(j,0)$ denotes the opposite. Each dimension x_i^j of an individual $\boldsymbol{x}_i$ is a subset of E^j that contains a single element, i.e., $x_i^j = \{(j,0)\}$ or $x_i^j = \{(j,1)\}$. Notably, the number of values in x_i^j depends on the considered problem. As a general method, the set-based DE can handle DOPs regardless of the number of elements that each variable takes from the universal set to compose a solution.

Based on the above representation scheme, the set-based DE initializes the population X as follows: for each dimension x_i^j of each individual $\boldsymbol{x}_i$, select a subset from the jth dimension E^j of the universal set $\boldsymbol{E}$, $i=1, 2,\ldots, NP$, $j=1, 2,\ldots, D$. To avoid generating infeasible solutions, problem-specific selection rules or repair strategies can be applied. After initialization, the population is evolved iteratively through the following three evolutionary operators redefined based on the set theory. Since the operators operate in the same way on all the individuals, the individual index i is omitted in the descriptions to facilitate illustration.

- *Mutation:* Similar to the classic DE, the mutation operator in the set-based DE generates a mutant set by adding one or more scaled differential sets to a base set. Each differential set is obtained from subtraction between two individuals. To operate in the discrete space, the related arithmetic operators are redefined as below.

  (1) '$\boldsymbol{x}_1 - \boldsymbol{x}_2$': As the set-based DE defines each dimension of an individual as a crisp set, the subtraction operator between two individuals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is equivalent to performing set subtraction on each dimension. That is,

  $$x_1^j - x_2^j = \left\{ e \middle| e \in x_1^j \text{ and } e \notin x_2^j \right\}, j = 1, 2, \ldots, D. \tag{6}$$

  The set subtraction generates a crisp set containing one or more elements that differ x_1^j from x_2^j. If x_1^j and x_2^j have the same element(s), the resulting set is empty. The subtraction operator in the set-based DE thus plays the same role as in the original DE, i.e., finding out the difference from one individual to the other.

  (2) '$F \times \boldsymbol{d}$': Let $\boldsymbol{d}$ be the differential set. The multiplication operator between the scale factor F and $\boldsymbol{d}$ is performed as

  $$F \times d^j = \begin{cases} d^j, \text{if } r < F \\ \emptyset, \text{otherwise} \end{cases}, j = 1, 2, \ldots, D, \tag{7}$$

  where $r\in(0,1)$ is a random number and d^j is the jth dimension of $\boldsymbol{d}$. As such, the scale factor F in the set-based DE has similar effect as in the original DE, i.e., controls the extent to which the difference between two individuals is taken into account.

  (3) '$\hat{\boldsymbol{d}}_1 + \hat{\boldsymbol{d}}_2$': Some continuous DE variants require several differential vectors. The set-based DE also allows such a possibility. Let $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$ be two scaled differential sets obtained from different pairs of individuals. The summation operator between $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$ is equivalent to performing set union on each dimension. That is,

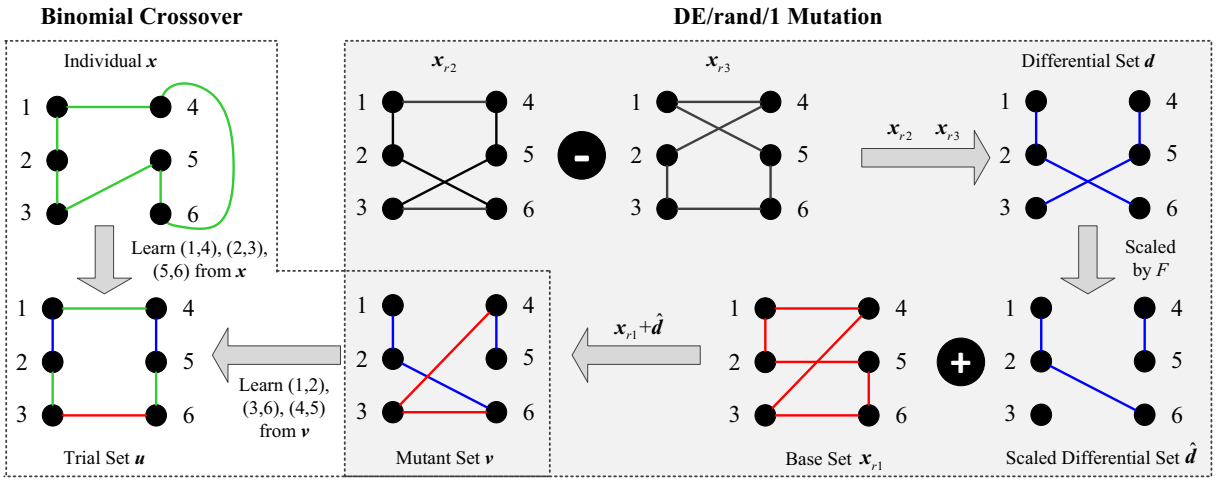
  $$\hat{d}_1^j + \hat{d}_2^j = \{e | e \in \hat{d}_1^j \text{ or } e \in \hat{d}_2^j\}, j = 1, 2, \ldots, D. \tag{8}$$

  (4) '$\boldsymbol{x} + \hat{\boldsymbol{d}}$': In the mutation operator of the original DE, adding a nonzero differential vector to the base vector leads to a new solution. The set-based DE follows the above idea to define the summation operator between the base set $\boldsymbol{x}$ and the scaled differential set $\hat{\boldsymbol{d}}$. More specifically,

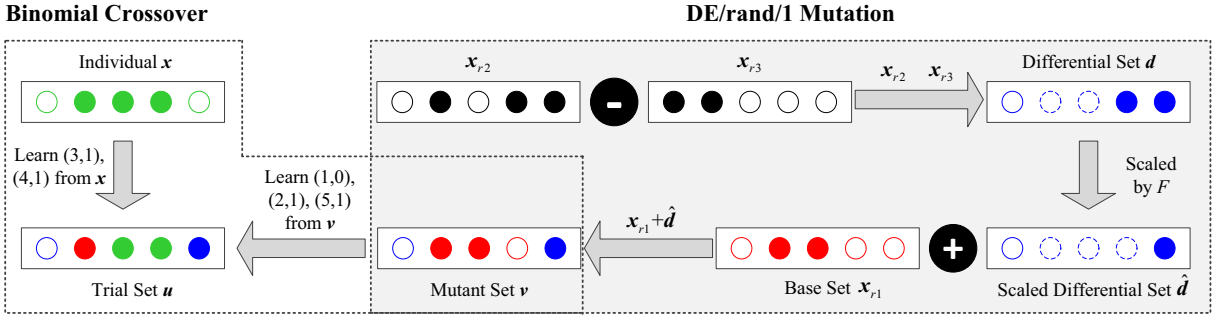
  $$x^j + \hat{d}^j = \begin{cases} x^j, & \text{if } \hat{d}^j = \emptyset \\ \hat{d}^j, & \text{otherwise} \end{cases}, j = 1, 2, \ldots, D. \tag{9}$$

  Based on the above redefined operators, a mutant set $\boldsymbol{v}=\{v^1, v^2,\ldots, v^D\}$ can be obtained for each individual. Note that $\boldsymbol{v}$ is not necessarily a feasible or even a complete solution, but each dimension v^j remains to be a subset of E^j. Taking TSP and BKP as examples, Fig. 2 illustrates the DE/rand/1 mutation operator in the set-based DE.

- *Crossover:* As in the original DE, the crossover operator in the set-based DE combines an individual $\boldsymbol{x}$ and its mutant set $\boldsymbol{v}$ to generate a trial set $\boldsymbol{u}$. Take the binomial crossover as an example. The pseudo-code of the crossover procedure is given in Algorithm 2. As shown in lines 05–09, for each dimension u^j of the trial set $\boldsymbol{u}$ ($j=1, 2,\ldots, D$), if a random number $r\in[0,1]$ is smaller than the crossover probability CR or j is the selected dimension j_{rand}, u^j learns from the corresponding dimension v^j of the mutant set $\boldsymbol{v}$; otherwise, u^j learns from the corresponding dimension x^j of the current individual $\boldsymbol{x}$. When learning from a set $\boldsymbol{s}$ ($\boldsymbol{s}$ can be either v^j or x^j), the elements that satisfy the problem constraints are first selected to form a candidate set $\boldsymbol{c}$ (line 11). Then the elements in $\boldsymbol{c}$ are selected into u^j one by one until $\boldsymbol{c}$ is empty or u^j is completed (e.g., u^j has enough elements) (lines 12–15). For example, in TSP, u^j is completed if exactly two edges connecting with node j have been selected; in BKP, u^j is completed if a single decision regarding the selection of item j, either $(j,0)$ or $(j,1)$, has been selected. Note that once an element is selected, the candidate set $\boldsymbol{c}$ must be updated



(a) TSP (a complete graph with 6 nodes)



(b) BKP (a total of 5 items)

Fig. 2. Examples of the set-based mutation and crossover operators in (a) TSP and (b) BKP. The TSP instance is a complete undirected graph with six nodes. The BKP instance has five items. The solid spot in (b) indicates selection of the corresponding item, whilst the hollow spot indicates the opposite. The dash circle indicates that the corresponding dimension is an empty set.

according to the problem constraints, because some elements that used to satisfy the constraints may become illegal after u^j changes. If u^j is still incomplete after taking all the elements in $\boldsymbol{c}$, the crossover procedure continues to select elements from the jth dimension E^j of the universal set $\boldsymbol{E}$ (lines 16–20). The selection process from E^j is similar to that from $\boldsymbol{s}$. Fig. 2 exemplifies the binomial crossover in TSP and BKP. From the above, it is clear that CR in the set-based DE plays the same role as in the original DE, that is, controls the probability that each dimension learns from the mutant set. In the case of exponential crossover, the only modification in the above procedure is to change the condition that controls when $\boldsymbol{u}$ learns from $\boldsymbol{v}$ (line 05). After crossover, it is possible that the resulting trial set is an unfeasible solution. In this case, problem-specific repair strategies can be performed.

- *Selection:* The selection operator in the set-based DE is the same as the one in the original DE. If the trial set $\boldsymbol{u}$ achieves a better objective value than $\boldsymbol{x}$, $\boldsymbol{u}$ replaces $\boldsymbol{x}$ and enters the population of the next generation. Otherwise, $\boldsymbol{x}$ remains unchanged.

As the original DE, the set-based DE iterates the above evolutionary operators until the termination criterion is met. Then the best individual $\boldsymbol{x}_{\text{bsf}}$ is reported as the result.

3.3. The proposed DE_{MV} for MVOPs

The set-based DE described above successfully extends the original DE to the discrete search space. More importantly, the set-based DE shares a similar search mechanism with the original DE. The roles of the two control parameters and the three evolutionary operators in the two DEs are consistent. Such a consistency will be further validated by additional experiments in Section 5.1. The fact that the original DE and the set-based DE share the same search mechanism makes it natural and promising to combine them for solving MVOPs. Based on the above idea, this paper proposes a hybrid DE framework, namely DE_{MV}, that hybridizes the original DE and the set-based DE for MVOPs. The pseudo-code of DE_{MV} is shown in Algorithm 3.

Algorithm 2 $\boldsymbol{u}$ = Binomial_Crossover($\boldsymbol{x}$, $\boldsymbol{v}$).

Input:
x: an individual
v: a mutant set
Output:
u: a trial set

BEGIN
01 $u \leftarrow \emptyset$
02 $j_{\text{rand}} \leftarrow$ a random number in {1, 2,..., D}
03 **FOR** each dimension $j = 1, 2, \ldots, D$
04 Generate a random number $r \in [0,1]$
05 **IF** $r < CR$ or $j = j_{\text{rand}}$
06 $u^j = \text{Learn}(v^j)$
07 **ELSE**
08 $u^j = \text{Learn}(x^j)$
09 **END IF**
10 **END FOR**
END
Function $u^j = \text{Learn}(\boldsymbol{s})$

BEGIN
11 $\boldsymbol{c} \leftarrow \{e \mid e \in \boldsymbol{s}$ and e satisfies all the problem constraints$\}$
12 **WHILE** u^j is complete and $\boldsymbol{c} \neq \emptyset$
13 Select an element from $\boldsymbol{c}$ and add it to u^j
14 Update $\boldsymbol{c}$ according to the problem constraints
15 **END WHILE**
16 $\boldsymbol{c} \leftarrow \{e \mid e \in E^j$ and e satisfies all the problem constraints$\}$
17 **WHILE** u^j is incomplete
18 Select an element from $\boldsymbol{c}$ and add it to u^j
19 Update $\boldsymbol{c}$ according to the problem constraints
20 **END WHILE**
END

As it can be seen, the decision variables of an MVOP are first classified into two parts: continuous and discrete. The indices of the continuous variables are recorded in a set I_C, whilst the indices of the discrete variables are recorded in a set I_D (lines 01–02). Then the initial population is generated by applying the initialization strategies of the original DE and the set-based DE to the variables indexed by I_C and I_D, respectively (lines 03–06). More specifically, let S be the search space of the continuous variables indexed by I_C and $\boldsymbol{E}$ be the universal set of elements available for the discrete variables indexed by I_D. Each individual $\boldsymbol{x}_i$ can be initialized by sampling each dimension of S and $\boldsymbol{E}$ separately ($i = 1, 2, \ldots, NP$). To avoid generating infeasible solutions, problem-specific sampling rules or repair strategies can be designed and performed.

After initialization, DE_{MV} enters the evolution cycle that involves mutation, crossover, and selection (lines 07–23). In each generation of the evolution cycle, for each individual $\boldsymbol{x}_i$ ($i = 1, 2, \ldots, NP$), the mutation and crossover operators of the original DE and the set-based DE are employed to evolve the variables indexed by I_C (lines 09–12) and I_D (lines 13–16), respectively. By combining the D trial variables generated, a complete trial solution $\boldsymbol{u}_i$ can be obtained (line 17). After evaluating all the trial solutions with the objective function f, the selection operator, which is the same in the two DEs, is performed to compare $\boldsymbol{u}_i$ with $\boldsymbol{x}_i$ (lines 20–22). The one with a better objective value will enter the population of the next generation. The evolution cycle repeats until the termination criterion is met and the best solution $\boldsymbol{x}_{\text{bsf}}$ is reported as the result (line 24). As described above, the optimization model of DE_{MV} is simple yet efficient, as will be testified by the experimental studies in Section 4.

As pointed out in Section 2, ordinal variables, although categorized as discrete variables, can be relaxed into continuous variables and handled by continuous optimization algorithms. On the other hand, they can be handled as categorical variables by discrete optimization algorithms if the order among available values is not taken into account. As such, both continuous and discrete optimization algorithms can handle ordinal variables. It remains an open issue how mixed-variable optimization algorithms, like DE_{MV}, should handle ordinal variables so as to achieve better performance. Few studies have elaborated on this issue. In Section 4, this paper studies this issue by designing a series of experiments to examine the performances of DE_{MV} with ordinal variables treated differently. The results are expected to contribute to our understandings of the above issue.

4. Experimental studies of DE_{MV}

In this section, the proposed DE_{MV} is applied to artificial MVOPs converted from continuous benchmark functions and real-world engineering problems with mixed variables. The results of DE_{MV} are compared with representative algorithms to further validate its effectiveness and efficiency. In the last subsection, an investigation is conducted on how to handle ordinal variables and categorical variables so that mixed-variable optimization algorithms can achieve better performances.

Algorithm 3 $\boldsymbol{x}_{bsf}$ = DE_{MV} (*NP*, *F*, *CR*).

Input:
NP: population size
F: scale factor
CR: crossover probability
Output:
$\boldsymbol{x}_{bsf}$: best-so-far solution

BEGIN
$I_C \leftarrow$ Indices of continuous variables
$I_D \leftarrow$ Indices of discrete variables
FOR each individual $\boldsymbol{x}_i$ ($i = 1, 2, \ldots, NP$)
  Initialize each variable $x_i^j (j \in I_C)$ using the initialization strategy of the original DE
  Initialize each variable $x_i^k (k \in I_D)$ using the initialization strategy of the set-based DE
END FOR
WHILE the termination criterion is not met
  FOR each individual $\boldsymbol{x}_i$ ($i = 1, 2, \ldots, NP$)
    FOR each index $j \in I_C$
      Generate a mutated variable v_i^j by the mutation operator of the original DE
      Generate a trial variable u_i^j by the crossover operator of the original DE
    END FOR
    FOR each index $k \in I_D$
      Generate a mutated variable v_i^k by the mutation operator of the set-based DE
      Generate a trial variable u_i^k by the crossover operator of the set-based DE
    END FOR
    Combine $u_i^j (j \in I_C)$ and $u_i^k (k \in I_D)$ into a complete trial solution $\boldsymbol{u}_i$
    Evaluate the objective value of $\boldsymbol{u}_i$ with f
  END FOR
  FOR each individual $\boldsymbol{x}_i$ ($i = 1, 2, \ldots, NP$)
    Update $\boldsymbol{x}_i$ with $\boldsymbol{u}_i$ according to Eq. (3)
  END FOR
END WHILE
Return the best individual $\boldsymbol{x}_{bsf}$ in the population as the result
END

4.1. Experiments on artificial MVOPs

4.1.1. Benchmark generation

The MVOPs used in this part are developed based on five unimodal functions and eight multimodal functions in the CEC2005 benchmark (i.e., F1–F6, F8–F14) [40]. The CEC2005 benchmark is chosen to facilitate comparisons with the representative algorithm ACO_{MV} [27], which also employed functions transformed from this benchmark in the experimental studies. The dimension of these thirteen functions are all set at ten (i.e., $D = 10$). To transform these continuous functions into MVOPs, the following method is employed.

A parameter $\rho \in (0,1)$ is introduced to control the proportion of discrete variables. According to the ratio, $\lceil \rho D \rceil$ variables are randomly selected to be discretized, whilst the other variables remain continuous. For each discretized variable, a set V of N values is generated. Among them, $(N-1)$ values are obtained by evenly sampling the search range of the variable, and the other one is the variable value on which the function achieves the optimum. Then the discretized variable is determined to be an ordinal or categorical variable. As an ordinal variable, V is the set of available values. Since values in the search range are naturally ordered, the values in V also have a clear order. As a categorical variable, the set of available values are set as $\{1, 2, \ldots, N\}$. Each integer in the set is uniquely related to a value randomly selected from V. As such, the set of available values is mapped to V without inheriting the ordering relation.

Using the above method, the thirteen benchmark functions in CEC2005 are converted to MVOPs with $\rho = 0.5$ and $N = 100$. In order to fully investigate the performance of the proposed DE_{MV}, three groups of MVOPs are derived. The first group of MVOPs involve continuous and ordinal variables. The second group of MVOPs involve continuous and categorical variables. The third group involve continuous, ordinal, and categorical variables, with the ratio of ordinal variables to categorical variables set as 2:3.

4.1.2. Experimental results and comparisons

The performance of the proposed DE_{MV} is compared with four representative algorithms for MVOPs: GA, ACO-DE, GA-DE, and ACO_{MV}. Among them, GA is the only EA that can handle MVOPs inherently. Both of ACO-DE and GA-DE use the classic DE for optimizing continuous variables and relaxed ordinal variables, but use ACO and GA for optimizing categorical variables, respectively. ACO_{MV} is an ACO-based mixed-variable optimization algorithm recently proposed by Liao et al. [27]. For each group of artificial MVOPs generated as above, the five algorithms are run for 100 independent times with the termination criterion of each run set as 50,000 function evaluations. Each algorithm reports the average distance to the optimal value of each function for comparison. A smaller distance indicates a better result.

Table 1
Results on artificial MVOPs with continuous & ordinal variables.

	GA	GA-DE	ACO-DE	ACO_{MV} [30]	DE_{MV}
F1	$2.129{,}719 \times 10^3$	$2.488{,}945 \times 10^2$	$5.497{,}622 \times 10^2$	$9.744{,}312 \times 10^{-2}$	0*
F2	$3.763{,}099 \times 10^3$	$3.004{,}915 \times 10^3$	$7.396{,}319 \times 10^2$	$1.588{,}953 \times 10^3$	3.334133*
F3	$2.087{,}768 \times 10^7$	$1.290{,}702 \times 10^6$	$5.666{,}851 \times 10^6$	$2.247{,}988 \times 10^6$	$2.947{,}024 \times 10^6$
F4	$4.473{,}846 \times 10^3$	$2.715{,}662 \times 10^3$	$9.425{,}363 \times 10^2$	$2.404{,}751 \times 10^3$	27.57903*
F5	$2.036{,}312 \times 10^4$	$1.794{,}904 \times 10^4$	$1.747{,}089 \times 10^4$	$1.217{,}914 \times 10^4$	$1.644{,}161 \times 10^4$
F6	$3.912{,}368 \times 10^7$	$9.577{,}014 \times 10^6$	$1.090{,}644 \times 10^6$	$8.529{,}926 \times 10^2$	56.01207*
F8	20.42519	20.37457	20.39647	20.38740	20.39388
F9	$1.323{,}208 \times 10^2$	28.88046	12.49311	73.46556	0*
F10	$2.033{,}022 \times 10^2$	67.36962	44.46091	$1.109{,}059 \times 10^2$	34.23669*
F11	9.229571	8.980393	8.729905	9.341289	8.341259*
F12	$2.167{,}877 \times 10^4$	$2.122{,}085 \times 10^3$	$1.583{,}455 \times 10^3$	$3.974{,}974 \times 10^3$	$2.121{,}968 \times 10^3$
F13	$2.431{,}886 \times 10^5$	$5.495{,}240 \times 10^5$	$2.662{,}483 \times 10^5$	$7.936{,}339 \times 10^5$	$7.279{,}254 \times 10^3$*
F14	3.983160	3.087209	3.319469	3.547523	3.676111

'*': The one-sided unpaired *t*-test finds the result significantly better than the others at the 0.05 level.

Table 2
Results on artificial MVOPs with continuous & categorical variables.

	GA	GA-DE	ACO-DE	ACO_{MV} [30]	DE_{MV}
F1	$2.096{,}523 \times 10^3$	24.03985	$5.384{,}242 \times 10^2$	$4.883{,}960 \times 10^2$	0*
F2	$3.619{,}453 \times 10^3$	$7.609{,}642 \times 10^2$	$1.607{,}069 \times 10^3$	$1.532{,}606 \times 10^3$	9.676140*
F3	$2.018{,}334 \times 10^7$	$1.410{,}724 \times 10^7$	$2.301{,}115 \times 10^7$	$1.419{,}493 \times 10^7$	$3.760{,}209 \times 10^6$*
F4	$4.381{,}144 \times 10^3$	$1.242{,}256 \times 10^3$	$1.964{,}541 \times 10^3$	$2.247{,}379 \times 10^3$	30.60446*
F5	$2.032{,}865 \times 10^4$	$1.743{,}671 \times 10^4$	$1.884{,}070 \times 10^4$	$1.339{,}680 \times 10^4$	$1.664{,}457 \times 10^4$
F6	$3.783{,}632 \times 10^7$	$6.171{,}272 \times 10^6$	$2.540{,}028 \times 10^6$	$6.906{,}911 \times 10^4$	$1.504{,}137 \times 10^2$*
F8	20.40223	20.39540	20.38825	20.38704	20.39968
F9	$1.337{,}236 \times 10^2$	1.698452	41.85008	79.24410	0*
F10	$2.093{,}780 \times 10^2$	72.33775	80.94039	$1.309{,}296 \times 10^2$	26.80897*
F11	9.336072	6.573612	9.501334	9.269823	8.452727
F12	$2.196{,}520 \times 10^4$	$2.485{,}339 \times 10^3$	$4.366{,}560 \times 10^4$	$6.633{,}967 \times 10^3$	$1.182{,}902 \times 10^3$*
F13	$2.403{,}978 \times 10^5$	$9.672{,}221 \times 10^3$	$3.394{,}445 \times 10^5$	$8.923{,}127 \times 10^4$	2.232409*
F14	3.999396	3.157789	3.376402	3.560220	3.713238

'*': The one-sided unpaired *t*-test finds the result significantly better than the others at the 0.05 level.

Table 3
Results on artificial MVOPs with continuous, ordinal, & categorical variables.

	GA	GA-DE	ACO-DE	ACO_{MV} [30]	DE_{MV}
F1	$2.157{,}078 \times 10^3$	79.75323	$5.703{,}749 \times 10^2$	15.32780	0*
F2	$3.804{,}272 \times 10^3$	$1.580{,}947 \times 10^3$	$1.916{,}640 \times 10^3$	$1.629{,}001 \times 10^3$	6.774124*
F3	$1.874{,}950 \times 10^7$	$1.470{,}268 \times 10^7$	$1.792{,}228 \times 10^6$	$9.386{,}997 \times 10^6$	$3.836{,}965 \times 10^6$*
F4	$4.597{,}231 \times 10^3$	$1.408{,}526 \times 10^3$	$2.246{,}263 \times 10^3$	$2.818{,}896 \times 10^3$	28.43589*
F5	$2.033{,}781 \times 10^4$	$1.751{,}794 \times 10^4$	$1.841{,}407 \times 10^4$	$1.309{,}186 \times 10^4$	$1.654{,}840 \times 10^4$
F6	$3.767{,}955 \times 10^7$	$5.028{,}674 \times 10^6$	$4.427{,}467 \times 10^6$	$3.980{,}150 \times 10^2$	$1.534{,}953 \times 10^2$*
F8	20.39802	20.39686	20.40656.	20.41086	20.41750
F9	$1.336{,}560 \times 10^2$	6.105148	43.13704	74.36113	0*
F10	$2.007{,}039 \times 10^2$	58.72323	94.14427	$1.072{,}743 \times 10^2$	31.57889*
F11	9.320379	8.959359	9.233052	9.412767	8.340853*
F12	$2.191{,}524 \times 10^4$	$1.993{,}342 \times 10^3$	$3.791{,}224 \times 10^4$	$5.527{,}256 \times 10^3$	$1.255{,}688 \times 10^3$*
F13	$2.210{,}256 \times 10^5$	$5.506{,}656 \times 10^4$	$1.115{,}804 \times 10^6$	$2.864{,}778 \times 10^5$	2.193270*
F14	3.986557	3.119762	3.357229	3.551345	3.720625

'*': The one-sided unpaired *t*-test finds the result significantly better than the others at the 0.05 level.

Tables 1–3 tabulate the results of the five algorithms on the three groups of MVOPs. The results of DE_{MV} are shown in **bold** if they are better than the other four algorithms. The one-sided unpaired *t*-test is performed to examine the significance of the advantage of DE_{MV}. An asterisk ('*') is attached to the results of DE_{MV} if significant advantage is detected at the 95% confidence level. From the tables, it can be observed that except on a few functions, DE_{MV} achieves significantly better results than the others. More specifically, DE_{MV} performs the best on eight, nine, and ten out of the thirteen functions in the first, second, and third groups, respectively.

Take F9 as an example. The convergence curves of DE_{MV} and ACO_{MV} are plotted in Fig. 3 for comparison of their search speed. The convergence curves on the other functions are similar. ACO_{MV} is chosen for comparison because of the following reasons: (1) neither DE nor ACO is originally applicable to MVOPs; (2) DE and ACO have been found promising on CnOPs and DOPs, respectively, which makes it attractive to extend their search mechanisms to MVOPs; (3) according to the summary

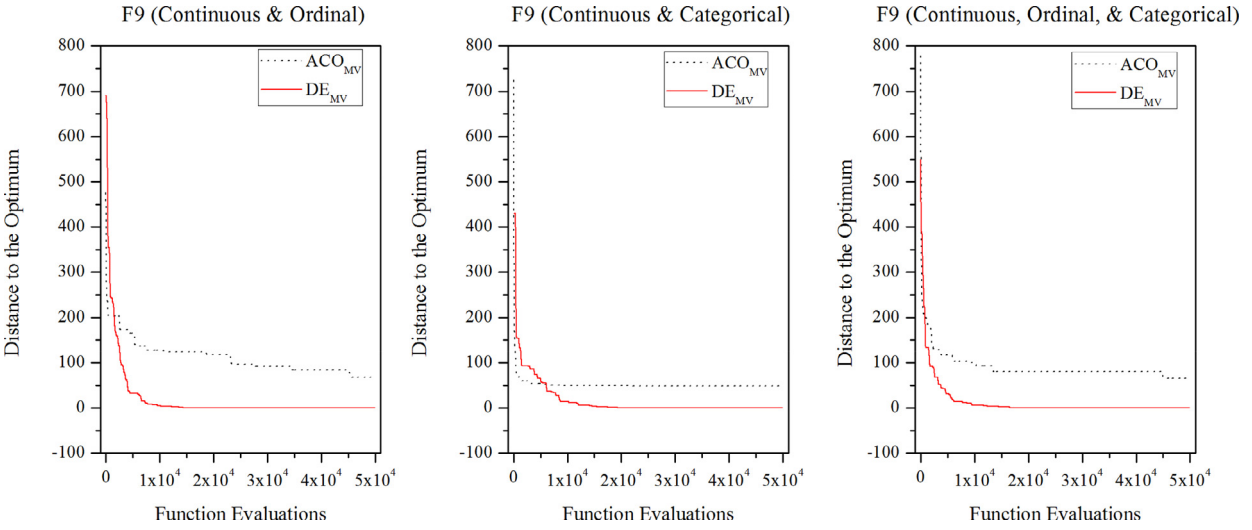


Fig. 3. Convergence curves of DEMV and ACOMV on F9 in the three groups.

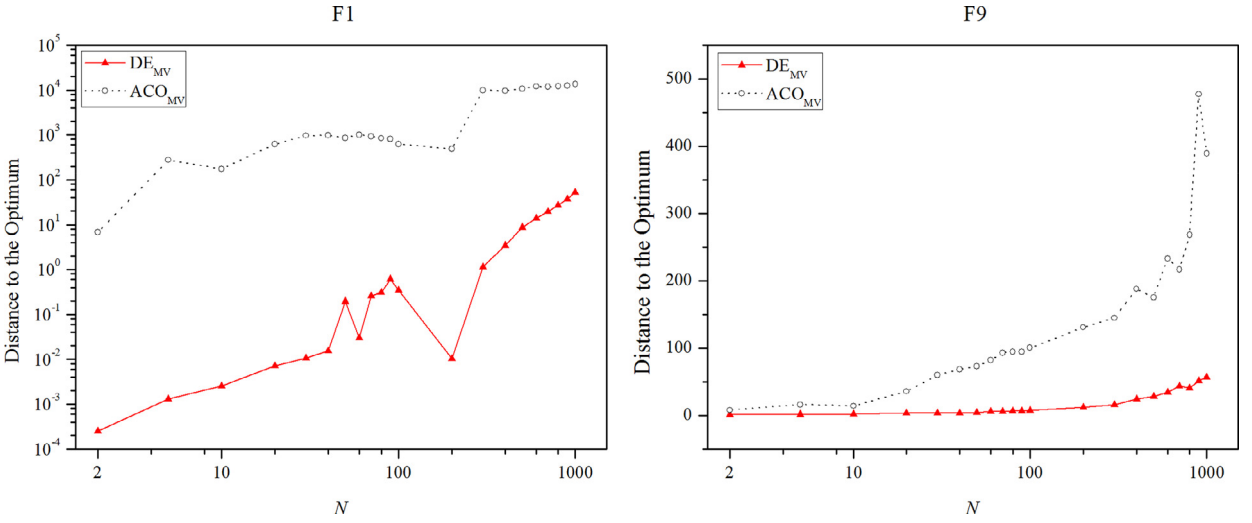


Fig. 4. The tendency of DEMV and ACOMV on F1 and F9 when N increases from 2 to 1000.

in Section 1, both DE_{MV} and ACO_{MV} belong to the third category of EAs for MVOPs. As seen from Fig. 3, DE_{MV} converges faster than ACO_{MV} and reaches a smaller function value.

It is clear that an MVOP becomes more difficult when the number of available values (i.e., the parameter N) for each discrete variable increases. In order to examine the performance of DE_{MV} with respect to different values of N, DE_{MV} is tested on F1 and F9 with N increasing from 2 to 1000. The average results obtained by DE_{MV} in 100 independent runs are plotted in Fig. 4 in comparison with ACO_{MV}. From the figure, it can be observed that although the performance of DE_{MV} gradually declines as N increases, the decline rate is slower than ACO_{MV} and it can still achieve better results than ACO_{MV}.

4.2. Experiments on real-world problems

In order to further examine the proposed DE_{MV}, it is applied to three real-world engineering problems with mixed variables. The categorical variables in these problems have small sets of available values. However, there are a number of complex constraints in each problem, which makes it challenging to find the optimal solution while satisfying all the constraints. In the following paragraphs, the definitions and the results of DE_{MV} on the three problems are given one by one.

4.2.1. Pressure vessel design problem - Case D (PVD-D)

The pressure vessel design (PVD) problem was first introduced in [18] and developed four different variations. Among them, case D is the most difficult one because of its extended search space [27]. A pressure vessel comprises a cylindrical

Table 4
Definition of PVD-D.

Objective function		min $f(T_s, T_h, R, L) = 0.6224T_sRL + 1.7781T_hR^2 + 3.1611T_s^2 + 19.84T_s^2R$
Constraints	g1	$-T_s + 0.0193R \le 0$
	g2	$-T_h + 0.009543R \le 0$
	g3	$-\pi R^2L - 4\pi R^3/3 + 750.1728 \le 0$
	g4	$L - 240 \le 0$
	g5	$0 \le T_s \le 100$
	g6	$0 \le T_h \le 100$
	g7	$10 \le R \le 200$
	g8	$10 \le L \le 200$

Table 5
Experimental results on PVD-D.

Methods	f_{Best}	f_{Mean}	f_{Worst}	Sd	FEs
CS[16]	6059.714	6447.736	6495.347	502.693	–
CLPSO[17]	6059.7143	6066.0311	–	12.2718	60,000
LCA[24]	6059.714353	6064.921613	6090.649839	10.4428332	50,000
IACO[25]	6059.7258	6081.7812	6150.1289	67.2418	–
ACO_{MV}[30]	6059.7143	6065.7923	6089.9893	12.2	30,000
TLBO[40]	6059.714335	6059.71434	–	–	10,000
DE_{MV}	6059.13	6059.15	6059.27	0.030652	22,000

'-': The information is not available in the source literature.

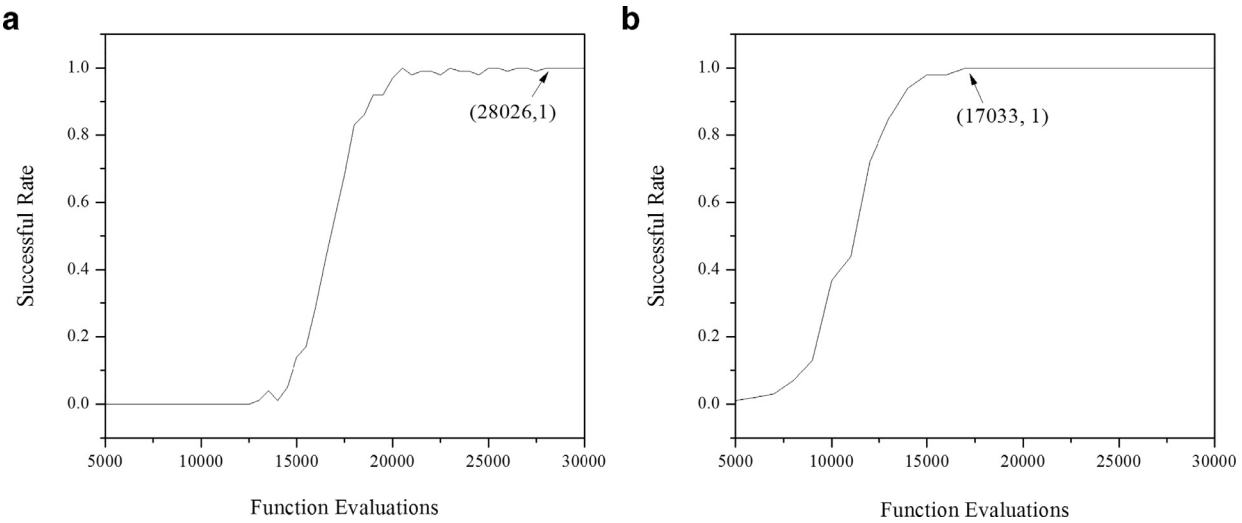


Fig. 5. The curve of DE_{MV}'s successful rate of finding a feasible solution for (a) PVD-D and (b) WBD-B along the number of function evaluations.

body and two hemispherical heads. The optimization objective is to minimize the manufacturing cost by adjusting four parameters: thickness of the pressure vessel T_s, thickness of the head T_h, inner radius of the cylinder body R, and length of the main cylinder L. Among them, R and L are continuous variables, while T_s and T_h are ordinal variables whose values must be the multiples of 0.0625 inches. The formal definitions of the objective function and the constraints are given in Table 4.

A number of methods have been proposed for solving the above problem [15,16,22,23,27,37]. Their results and the results of DE_{MV} are tabulated in Table 5, where FEs is the number of function evaluations used by each algorithm. As it can be seen, the results of DE_{MV} are the best and the most stable among the seven algorithms. Moreover, DE_{MV} can reach the currently known best 6059.13. It is also observed that the successful rate of finding a feasible solution increases when DE_{MV} is allowed to run for a longer time. This tendency is depicted in Fig. 5(a). As it can be seen, DEMV can find a feasible solution with 100% successful rate when the number of function evaluations exceeds 28,026.

4.2.2. Thermal insulation systems design problem (TISD)

A thermal insulation system is used to control the heat flow from a hot surface to a cold one. Several heat intercepts are inserted into the system, each of which maintains a fixed temperature. The space between every pair of intercepts is filled with insulators. The heat flow can be minimized by optimizing the following parameters: number of intercepts n, location x_i and temperature T_i of each intercept ($i = 1, 2, \ldots, n$), and type of insulators I_j filled in the space between every pair of

Table 6
Definition of TISD.

Objective function		$\min f(\boldsymbol{x}, \boldsymbol{T}) = \sum_{i=1}^{n} P_i = AC_i\left(\frac{T_{\text{hot}}}{T_{\text{cold}}} - 1\right)\left(\frac{1}{\Delta x_i}\int_{T_i}^{T_{i+1}} kdT - \frac{1}{\Delta x_{i-1}}\int_{T_{i-1}}^{T_i} kdT\right)$ where $A = 1,\ L] = 1,\ T_{\text{cold}} = 4.2,\ T_{\text{hot}} = 300,\ C = \begin{cases} 2.5, & \text{if } T \geq 71K \\ 4, & \text{if } 4.2K < T < 71K, \text{ and the material set is} \\ 5, & \text{if } T \leq 4.2K \end{cases}$ {Teflon(T), Nylon(N), epoxy-fiberglass (in plane cloth) (F), epoxy-fiberglass (in normal cloth) (E), stainless steel (S), aluminum (A), low-carbon steel (L)}.
Constraints	g1	$\sum_{i=1}^{n} \Delta x_i = 1$ and $\Delta x_i \geq 0 (i = 1, 2, ..., n)$
	g2	$T_{\text{cold}} \leq T_1 \leq T_2 \leq \ldots \leq T_{n-1} \leq T_n \leq T_{\text{hot}}$

Table 7
Experimental results on TISD.

Number of intercepts (n)	FEs	MVP[1]	MIDC-DE[28]	ACO_{MV}[30]	DE_{MV}
		Best Mean Worst	Best Mean Worst	Best Mean Worst	Best Mean Worst
7	10,000	27.04 28.53 31.09	28.39 32.45 47.31	25.93 27.79 32.10	25.54 26.77 28.10
10	10,000	26.17 28.13 31.72	29.49 36.57 50.47	25.30 26.52 31.94	24.68 26.35 28.03
20	10,000	26.66 29.41 36.34	38.47 61.12 127.3	25.19 26.41 31.00	24.53 25.97 27.30
30	20,000	27.01 28.56 33.48	36.06 53.95 5485.38	25.35 26.27 32.08	24.38 25.68 28.12
50	100,000	26.69 28.63 32.23	33.65 40.39 68.39	25.42 27.27 31.08	25.05 26.71 30.02

intercepts ($j = 1, 2, \ldots, n-1$). The type of insulators is chosen from a set of available types. The location and the temperature of the intercept are continuous. One of the difficulties of this problem is that the number of decision variables also needs to be optimized. Following the existing methods [1,25], we assign several values to n and focus on optimizing the other variables (continuous and categorical) with n fixed. Detailed definitions of the objective function and the constraints of this problem are listed in Table 6.

Table 7 tabulates the results of DE_{MV} as well as other three representative algorithms under different values of n. The three algorithms in comparison are MVP [1], MIDC-DE [26] and ACO_{MV} [27]. With the same number of function evaluations, it is shown that the proposed DE_{MV} obtains the best results under all the settings of n.

4.2.3. Welded beam design problem - Case B (WBD-B)

A rectangular beam is designed to be welded as a cantilever beam to load a certain weight. The objective is to minimize the cost for constructing the beam. In case A, there are four parameters that influence the cost: thickness of the weld x_1, length of the weld x_2, width of the beam x_3, and thickness of the beam x_4. Among them, x_2 is a continuous variable, while x_1, x_3, x_4 are ordinal variables whose values must be the multiples of 0.0625 inches. In case B, there are two additional parameters: beam material x_5 and welded joint configuration x_6 (two-side or four-side), both of which are categorical variables. Therefore, WBD-B is a problem involving all three types of variables. Definitions of the objective function and the constraints of WBD-B are given in Table 8.

Only a few methods have been proposed for WBD-B, including GeneAS [9], PSOA [10], CLPSO [16], ACO_{MV} [27], RSPSO [46]. All these algorithms are compared with the proposed DE_{MV}, as shown in Table 9. As it can be seen, although DE_{MV} needs a larger number of function evaluations than PSOA and RSPSO, it achieves the best results among all the algorithms. Moreover, the standard deviation of DE_{MV} is the smallest, indicating that it has better stability. The tendency curve in Fig. 5(b) shows that DE_{MV} can always find a feasible solution when the number of function evaluations exceeds 170,333.

4.3. Further discussions

This subsection focuses on investigating the following two issues: (1) how to optimize ordinal variables so that a mixed-variable optimization algorithm achieves better performance; (2) whether or not treating categorical variables as ordinal variables worsens the performance of a mixed-variable optimization algorithm. Experiments are conducted by applying

Table 8
Definition of WBD-B.

Objective function		$\min f(\boldsymbol{x}) = (1+c_1)x_1^2x_2 + c_2x_3x_4(14+x_2)$
Constraints	g1	$\tau(x) - \tau_{\max} \le 0$
	g2	$\sigma(x) - S \le 0$
	g3	$x_1 - x_4 \le 0$
	g4	$0.10471x_1^2 + 0.04811x_3x_4(14+x_2) - 5 \le 0$
	g5	$0.125 - x_1 \le 0$
	g6	$\delta(x) - \delta_{\max} \le 0$
	g7	$P - P_c(x) \le 0$
	g8	$0.1 \le x_1,\ x_4 \le 2.0$
	g9	$0.1 \le x_2,\ x_4 \le 10.0$

where

$L = 14,\ P = 600,\ \delta_{\max} = 0.25,\ \tau_{\max} = 0.577S,$

$\tau(\vec{x}) = \sqrt{(\tau')^2 + \tau'\tau''\frac{x_2}{R} + (\tau'')^2},\ \tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J},\ M = P(L+\frac{x_2}{2}),\ R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2},$

$J = \begin{cases} 2\{\sqrt{2}x_1x_2[\frac{x_2^2}{12} + (\frac{x_1+x_3}{2})^2]\}, & \text{if } x_6 = \text{two} - \text{side} \\ 2[\sqrt{2}x_1(\frac{x_1+x_2+x_3}{12})^2], & \text{if } x_6 = \text{four} - \text{side} \end{cases},$

$\sigma(\boldsymbol{x}) = \frac{6PL}{x_4x_3^2},\ \delta(\boldsymbol{x}) = \frac{4PL^3}{Ex_3^4x_4},\ P_c(\vec{x}) = \frac{4.013E\sqrt{x_3^2x_4^6/36}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}})$

$x_5 \in \{\text{Steel (S), Cast iron (C), Aluminum (A), Brass (B)}\},$

$S = \begin{cases} 3\times10^4, & \text{if } x_5 = \text{S} \\ 8\times10^3, & \text{if } x_5 = \text{C} \\ 5\times10^3, & \text{if } x_5 = \text{A} \\ 8\times10^3, & \text{if } x_5 = \text{B} \end{cases},\ E = \begin{cases} 3\times10^7, & \text{if } x_5 = \text{S} \\ 1.4\times10^7, & \text{if } x_5 = \text{C} \\ 10^7, & \text{if } x_5 = \text{A} \\ 1.6\times10^7, & \text{if } x_5 = \text{B} \end{cases},\ G = \begin{cases} 1.2\times10^7, & \text{if } x_5 = \text{S} \\ 6\times10^6, & \text{if } x_5 = \text{C} \\ 4\times10^6, & \text{if } x_5 = \text{A} \\ 6\times10^6, & \text{if } x_5 = \text{B} \end{cases},$

$c_1 = \begin{cases} 0.1047, & \text{if } x_5 = \text{S} \\ 0.0489, & \text{if } x_5 = \text{C} \\ 0.5235, & \text{if } x_5 = \text{A} \\ 0.5584, & \text{if } x_5 = \text{B} \end{cases}, \text{ and } c_2 = \begin{cases} 0.0481, & \text{if } x_5 = \text{S} \\ 0.0244, & \text{if } x_5 = \text{C} \\ 0.2405, & \text{if } x_5 = \text{A} \\ 0.2566, & \text{if } x_5 = \text{B} \end{cases}$

Table 9
Experimental results of WBD-B.

Methods	f_{Best}	f_{Mean}	f_{Worst}	Sd	FEs
GeneAS [9]	1.9422	–	–	–	–
PSOA[11]	1.7631	1.7631	–	0.00	6570
CLPSO[17]	1.5809	1.7405	–	0.2109	60,000
ACO_{MV}[30]	1.58089	1.59089	1.75083	0.027764	20,000
RSPSO[51]	1.9421	1.9515	1.9986	0.021412	15,000
DE_{MV}	**1.58089**	**1.58091**	**1.58102**	**9.7477 × 10^{-5}**	21,000

'–': The information is not available in the source literature.

DE_{MV} and its two variants to the artificial MVOPs generated in Section 4.1. The two variants, namely DE_{MV}/O and DE_{MV}/C, are set as below:

- **DE_{MV}/O:** The ordinal variables in an MVOP are relaxed into continuous variables and optimized through the original DE in DE_{MV}. The other parts of DE_{MV} remain unchanged.
- **DE_{MV}/C:** A random order is forced onto the available values of each categorical variable in an MVOP. Then the categorical variables are treated as ordinal variables and optimized through the original DE in DE_{MV}. The other parts of DE_{MV} remain unchanged.

To make the investigation more comprehensive, the parameter N (i.e., the number of available values for each discrete variable) varies from 10 to 1000. All the parameters of DE_{MV}, DE_{MV}/O, and DE_{MV}/C are set identical for the sake of fair comparison. The average results of the three algorithms over 100 independent runs are reported in Table 10. The numbers in **bold** are the best results among the three algorithms. The one-sided unpaired t-test is performed to examine the significance of the difference between the best result and the other two results. If the best result is significantly better at the 95% confidence level, an asterisk ('*') is attached. If the best result is only significantly better than the worst result at the 0.05 level, a dagger ('†') is attached.

From Table 10, it can be observed that DE_{MV} significantly outperforms DE_{MV}/O on 15 out of the 42 functions, whilst DE_{MV}/O performs significantly better than DE_{MV} on 16 functions. In fact, on most functions, the results of DE_{MV} and DE_{MV}/O have the same order of magnitude. Such a phenomenon implies that relaxing ordinal variables does not necessarily bring poor performance. Overall, it is shown that whether to relax the ordinal variables into continuous variables during optimization does not have significant influence on the results. On the other hand, DE_{MV}/C performs the worst among the three algorithms in most cases. In fact, it only achieves the best results on six out of the 42 functions. Moreover, the advantage of DE_{MV}/C is minor (insignificant on four functions). This phenomenon implies that it may be unsuitable to convert categorical variables into ordinal ones. Taken together, it suggests that the performance of a mixed-variable optimization algorithm

Table 10
Results of DE_{MV}, DE_{MV}/O, and DE_{MV}/C on artificial MVOPs.

N	Functions	DE_{MV}/O	DE_{MV}/C	DE_{MV}
10	F1	$8.946{,}350 \times 10^{-4}$	$2.419{,}294 \times 10^{3}$	**$3.588{,}254 \times 10^{-4}$***†
	F2	$8.080{,}389 \times 10^{2}$	$1.380{,}392 \times 10^{3}$	**$6.901{,}143 \times 10^{2}$***†
	F3	**$6.830{,}528 \times 10^{6}$***†	$1.106{,}220 \times 10^{7}$	$7.891{,}363 \times 10^{6}$
	F4	$1.328{,}781 \times 10^{3}$	$2.066{,}404 \times 10^{3}$	**$1.216{,}473 \times 10^{3}$***†
	F5	**$4.706{,}069 \times 10^{3}$**†	$8.274{,}307 \times 10^{3}$	$4.710{,}860 \times 10^{3}$
	F6	8.408570	$1.595{,}879 \times 10^{8}$	**7.958114***†
	F8	20.51661	**20.50978**	20.51689
	F9	**1.731765***†	2.994249	1.999061
	F10	29.72783	**28.34231***†	29.28293
	F11	**9.931982**†	9.932822	9.998626
	F12	**$2.864{,}086 \times 10^{3}$***†	$4.526{,}280 \times 10^{3}$	$2.911{,}310 \times 10^{3}$
	F13	**2.264092***†	2.776398	2.519634
	F14	**3.965334**†	3.972255	3.985575
100	F1	0.7032478	15.19386	**0.3417352***†
	F2	$1.040{,}592 \times 10^{3}$	$1.412{,}184 \times 10^{3}$	**$9.048{,}929 \times 10^{2}$***†
	F3	$1.517{,}019 \times 10^{7}$	**$1.081{,}837 \times 10^{7}$***†	$1.382{,}826 \times 10^{7}$
	F4	**$1.639{,}445 \times 10^{3}$**†	$1.994{,}356 \times 10^{3}$	$1.644{,}759 \times 10^{3}$
	F5	$1.686{,}211 \times 10^{4}$	**$1.670{,}047 \times 10^{4}$**	$1.678{,}525 \times 10^{4}$
	F6	$7.411{,}809 \times 10^{2}$	$1.218{,}282 \times 10^{4}$	**$4.657{,}547 \times 10^{2}$***†
	F8	20.52371	**20.50819**	20.52895
	F9	**6.522907***†	56.30631	7.165795
	F10	**50.63038***†	$1.207{,}705 \times 10^{2}$	56.52157
	F11	9.992939	10.10783	**9.924335***†
	F12	**$4.835{,}052 \times 10^{3}$***†	$9.479{,}160 \times 10^{3}$	$5.548{,}231 \times 10^{3}$
	F13	**7.935829***†	$6.689{,}221 \times 10^{4}$	$9.369{,}480 \times 10^{2}$
	F14	**4.034995**	4.058661	4.058261
1000	F1	60.90802	$7.242{,}637 \times 10^{2}$	**41.15776***†
	F2	**$1.067{,}692 \times 10^{3}$**†	$1.062{,}291 \times 10^{4}$	$1.469{,}394 \times 10^{3}$
	F3	$9.052{,}137 \times 10^{7}$	$1.062{,}025 \times 10^{8}$	**$8.435{,}498 \times 10^{7}$***†
	F4	**$1.921{,}777 \times 10^{3}$***†	$1.724{,}100 \times 10^{4}$	$2.262{,}626 \times 10^{3}$
	F5	$3.615{,}111 \times 10^{4}$	$3.938{,}914 \times 10^{4}$	**$3.586{,}535 \times 10^{4}$***†
	F6	$2.685{,}504 \times 10^{5}$	$1.534{,}413 \times 10^{7}$	**$1.872{,}081 \times 10^{5}$***†
	F8	20.51010	**20.50702**	20.51498
	F9	**50.66213***†	$2.344{,}541 \times 10^{2}$	69.42306
	F10	**$1.482{,}174 \times 10^{2}$***†	$5.183{,}787 \times 10^{2}$	$1.535{,}253 \times 10^{2}$
	F11	10.03274	10.08959	**9.913198***†
	F12	**$4.617{,}848 \times 10^{3}$***†	$1.028{,}499 \times 10^{4}$	$5.246{,}665 \times 10^{3}$
	F13	$8.582{,}874 \times 10^{6}$	$6.362{,}986 \times 10^{8}$	**$4.682{,}693 \times 10^{6}$***†
	F14	4.347009	4.389818	**4.333485***†

'*': The one-sided unpaired *t*-test finds the best result significantly better than the second-best result.
'†': The one-sided unpaired *t*-test finds the best result significantly better than the worst result.

probably depends on whether it has an efficient search mechanism to handle continuous (including ordinal) and categorical variables simultaneously.

5. Additional experiments and discussions on DE_{MV}

The additional experiments in this section have two purposes. Firstly, we aim to provide experimental evidence for the foundation of the proposed DE_{MV}, that is, it hybridizes the original DE and the set-based DE for MVOPs due to their consistency and compatibility. To this end, using TSP as an example, the effects of the evolutionary operators (mutation and crossover) and the control parameters (F and CR) in set-based DE are analyzed and compared with those in the original DE. Secondly, we perform an investigation on the settings of operators and parameters of DE_{MV} on MVOP benchmark generated in Section 4.1. The results are expected to not only provide further evidence for the foundation of DE_{MV}, but also offer reference for its practical use.

5.1. Effects of operators and parameters in set-based DE

Using a TSP instance kroa100 as an example, this subsection investigates the effects of operators and parameters in the set-based DE.

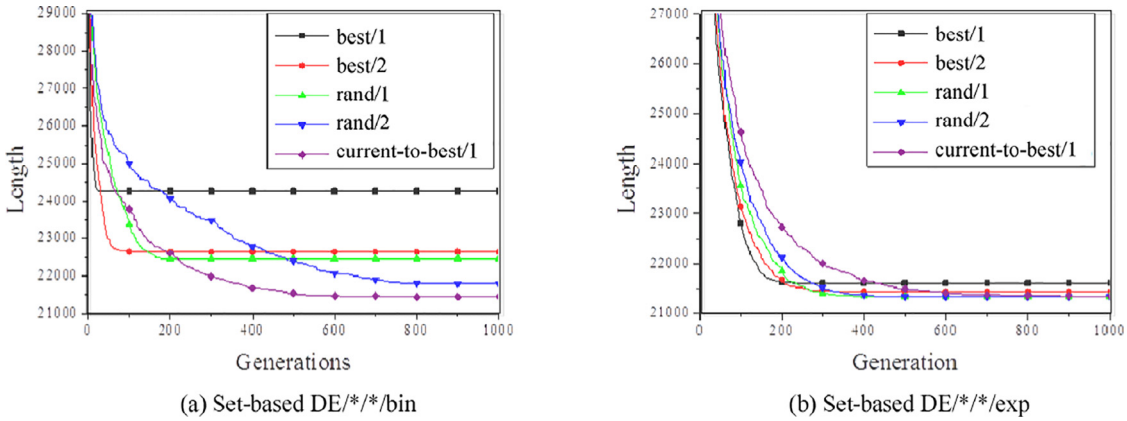


Fig. 6. Convergence curves of the set-based DE using different mutation and crossover operators on kroa100, where '*' can be any mutation operator tested.

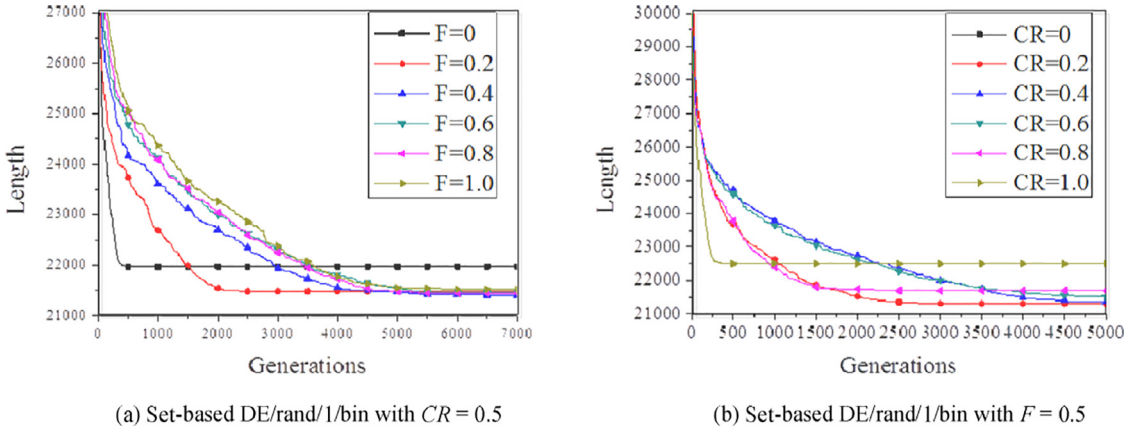


Fig. 7. Convergence curves of set-based DE/rand/1/bin on kroa100 with different settings of F and CR. Note that the convergence curve with $F = 0.5$ and $CR = 0$ is not shown in (b) because the result is too poor to be presented in the same scale with the others.

5.1.1. Effects of mutation and crossover operators

The main evolutionary operators in DE are the crossover and mutation operators. To investigate their effects, a series of set-based DE variants that use different mutation and crossover operators are derived. Five mutation operators (namely "best/1", "best/2";, "rand/1", "rand/2", "current-to-best/1") and two crossover operators ("bin" and "exp") are tested, resulting in a total of ten set-based DE variants. With F and CR both set as 0.5, each of the ten variants is run for 100 independent times. Their convergence curves are plotted in Fig. 6.

From Fig. 6, it can be observed that no matter which crossover operator is used, the variants whose mutation operators use the best individual as the base set converge at a faster speed, but they are also more likely to suffer from early convergence. The variants whose mutation operators use a random individual or the current individual as the base set converge at a slower speed, but they show a stronger exploration ability and achieve better results. Comparing Fig. 6(a) with (b), it can be seen that the variants using exponential crossover generally outperform the variants using binominal crossover, no matter which mutation operator is used.

5.1.2. Effects of F and CR

The parameters F and CR have significant influence on the performance of original DE. The scale factor F determines the relative distance of the mutant vector from the base vector, and the crossover probability CR controls the fraction of the trial vector learnt from the mutant vector. According to the design of set-based DE, F and CR are supposed to play similar roles. For validation, this part takes the set-based DE/rand/1/bin (corresponding to the classic DE for CnOPs) as an example and investigates the effects of F and CR. The set-based DE/rand/1/bin is tested on kroa100 with different settings of F and CR for 100 independent times. Fig. 7(a) and (b) plot the average convergence curves of set-based DE/rand/1/bin with one parameter (F or CR) fixed at a moderate value, i.e., 0.5, and the other parameter (CR or F) increasing from 0 to 1 with a step of 0.2.

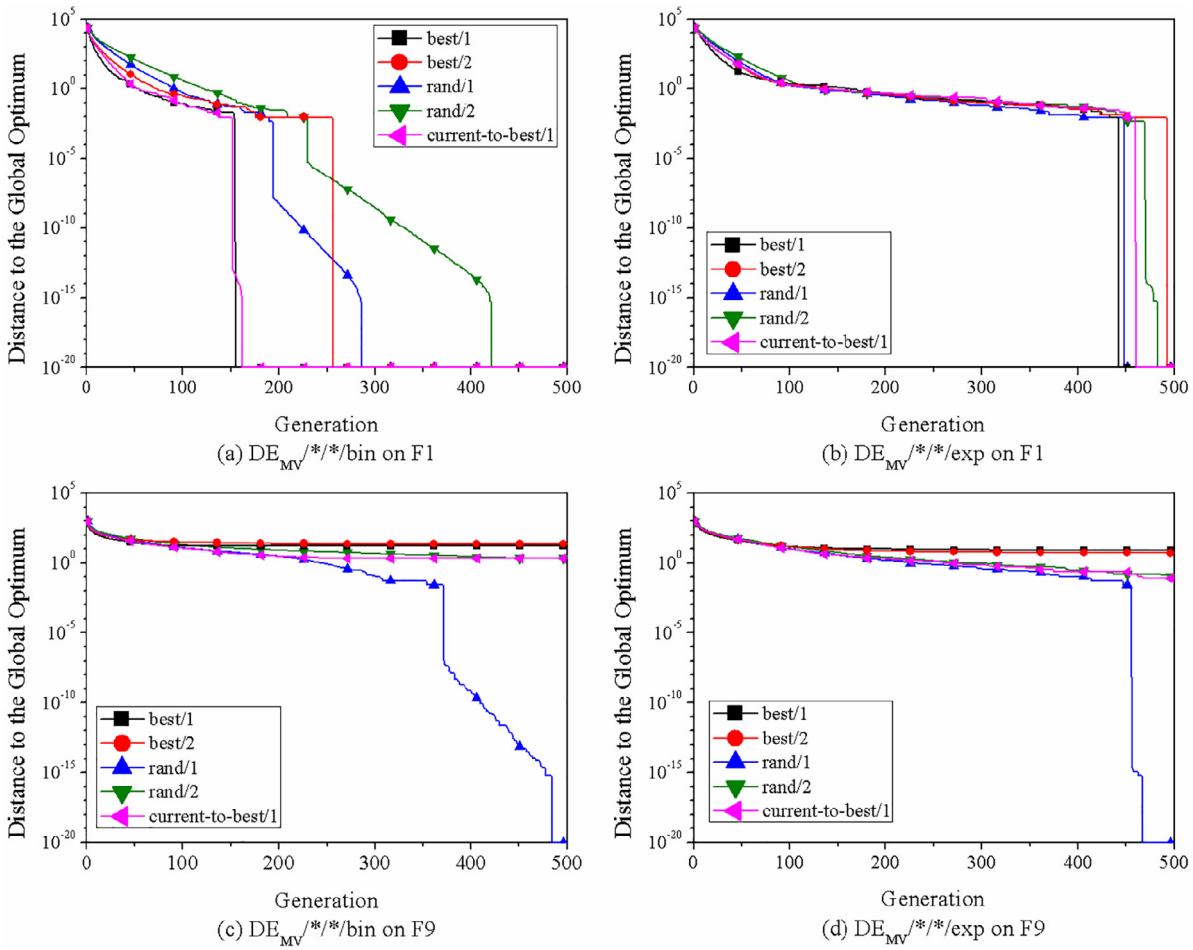


Fig. 8. Convergence curves of DE_{MV} using different mutation and crossover operators on MVOP benchmark F1 and F9, where '*' can be any mutation operator tested.

Fig. 7 shows that *F* and *CR* both have significant influence on the performance of set-based DE. In Fig. 7(a), the slope of the convergence curve becomes gentler as *F* increases. Meanwhile, the set-based DE is less likely to suffer from early convergence. Fig. 7(b) shows that the increase of *CR* generally leads to a faster convergence speed of the set-based DE, but it also raises the risk of early convergence.

The above results about the effects of operators and parameters are not surprising, since consistent relations have already been observed in the original DE for CnOPs. The consistency, however, does support our hypothesis that the set-based DE and the original DE share a similar search mechanism. It is such compatibility and consistency that lays the foundation for hybridizing them to solve MVOPs cooperatively.

5.2. Effects of operators and parameters in DE_{MV}

After validating that the original DE for CnOPs and the set-based DE for DOPs share a similar search mechanism, this subsection further investigates the effects of operators and parameters in DE_{MV}. The investigation takes the unimodal function F1 and the multimodal function F9 in the third group of MVOP benchmark as examples. As illustrated in Section 4.1, both of F1 and F9 have continuous, ordinal, and categorical variables, and the number of available values for each discrete variable (either ordinal or categorical) is set as $N = 100$.

5.2.1. Effects of mutation and crossover operators

Similar with the investigation in Section 5.1.1, ten variants of DE_{MV} are derived by combining the five mutation operators (namely "best/1", "best/2", "rand/1", "rand/2", "current-to-best/1") and the two crossover operators ("bin" and "exp"). With *F* and *CR* both set as 0.5, the convergence curves of the ten variants averaged over 100 independent runs on F1 and F9 are plotted in Fig. 8. Notably, the vertical axis represents the logarithm of the distance to the global optimum. Since the loga-

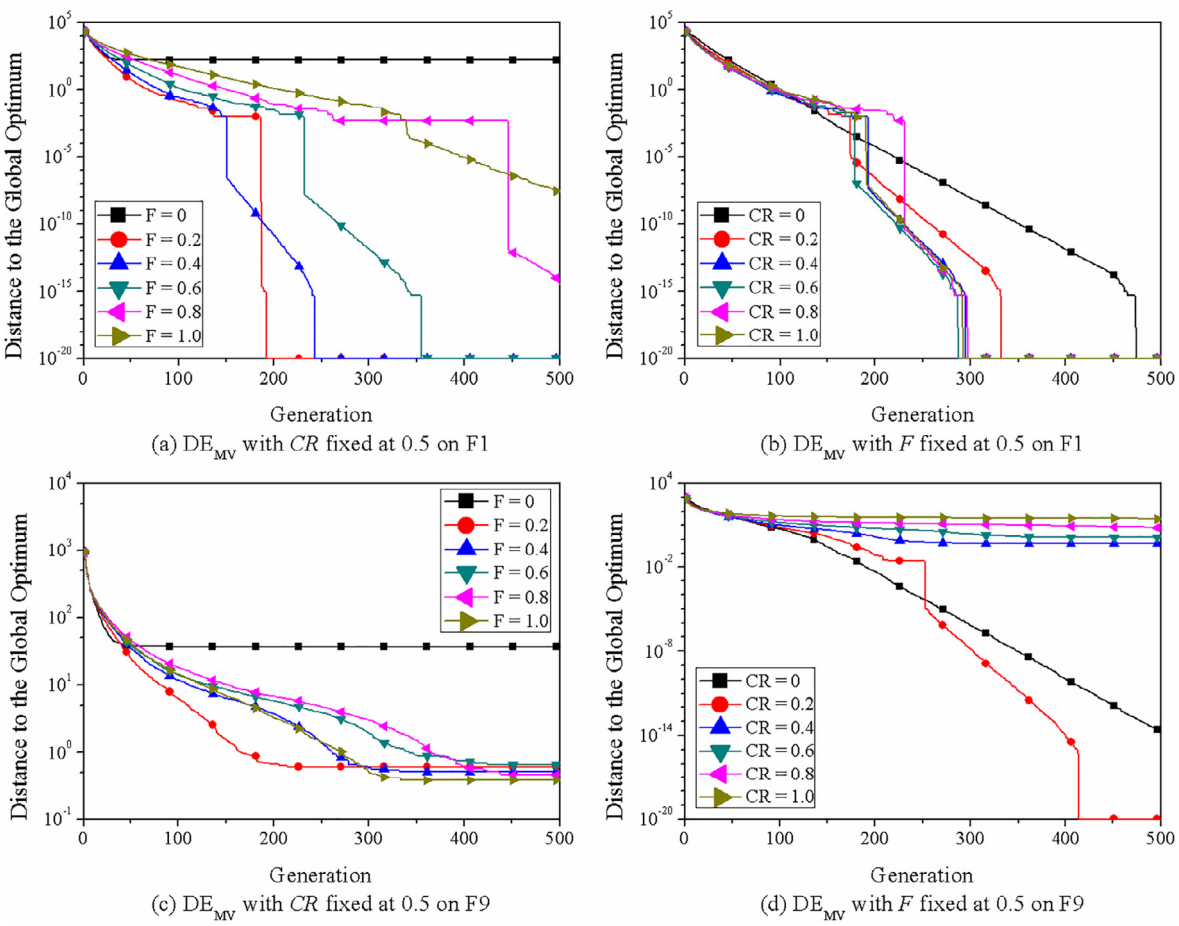


Fig. 9. Convergence curves of DE_{MV}/rand/1/bin with different settings of F and CR on MVOP benchmark F1 and F9 with continuous, ordinal, and categorical variables.

rithm of zero is not defined, a small constant (10^{-20}) is added to the results. Therefore, a convergence curve that descends to the level of 10^{-20} means that the corresponding algorithm achieves the global optimum of the function.

As can be observed from Fig. 8, no matter which crossover operator is used, DE_{MV} variants whose mutation operators involve the best individual (i.e., "best/1", "best/2", and "current-to-best") generally converge faster than the others. This helps DE_{MV} to achieve better performance on unimodal F1 [Fig. 8(a) and (b)], but it leads to easier trap in local optima on multimodal problems like F9 [Fig. 8(c) and (d)]. As for the crossover operator, by comparing Fig. 8(c) and (d), it is found that using the exponential crossover makes DE_{MV} achieve better results on multimodal F9 than using the binomial crossover [Fig. 8(c) and (d)]. However, it may slow down the convergence of DE_{MV} on unimodal functions like F1 [Fig. 8(a) and (b)].

5.2.2. Effects of F and CR

As in Section 5.1.2, we take DE_{MV}/rand/1/bin as an example and investigate the effects of F and CR, respectively. During the investigation, one of the two parameters is varied from zero to one at a step of 0.2, whilst the other one is fixed at 0.5. The average convergence curves of DE_{MV} over 100 independent runs on F1 and F9 are plotted in Fig. 9, where the logarithm scale is applied to the vertical axis in the same way as in the previous subsection.

Fig. 9(a) and (c) show that the convergence curves on F1 and F9 both descend in a gentler manner as F increases, implying a negative correlation between F and the convergence speed. Meanwhile, large F helps relieve the problem of early convergence on multimodal functions, as indicated by better results achieved on F9 when F rises [Fig. 9(c)]. With respect to CR, the convergence curves on F1 and F9 [Fig. 9(b) and (d)] both show that the convergence speed is positively correlated with CR. However, large CR also increases the risk of getting trapped in local optima, as indicated by poorer results on F9 when CR rises.

Comparing the above results with those obtained from the investigation on the set-based DE, it is clear that the main findings are consistent and compactible, which suggests that the effects of operators and parameters in DE_{MV} are similar with the set-based DE and thus with the original DE. Such similarity not only confirms our hypothesis that the set-based

DE and the original DE share the same search mechanism, but also implies that the empirical rules for choosing operators and parameters of the original DE might serve as good reference for DE_{MV} in applications.

6. Conclusion

This paper proposes a novel DE algorithm (DE_{MV}) that hybridizes the original DE and the set-based DE for solving MVOPs. In DE_{MV}, the original DE is responsible for optimizing continuous decision variables, whilst the set-based DE is for discrete decision variables. The rationale behind such hybridization lies in the hypothesis that the original DE and the set-based DE share the same search mechanism and thus are highly compatible. The hypothesis is validated not only by analyzing the consistency in algorithmic design, but also by experimental studies that illustrate the identical effects of evolutionary operators and control parameters. The proposed DE_{MV} is applied to thirteen artificial MVOPs converted from benchmark functions of CEC 2005 and three real-world engineering problems with mixed variables. Comparisons with representative mixed-variable optimization algorithms show that the proposed DE_{MV} can achieve the best performance in most cases. This paper also investigates the issue about how to deal with ordinal variables so that the algorithm can achieve better performance. Experiments show that relaxing ordinal variables into continuous ones does not necessarily have significant influence on the results.

Acknowledgment

This paper is supported by the Natural Science Foundation of China (NSFC) Nos. 61622206, 61772569, and 61332002, the Guangzhou Pearl River New Star of Science and Technology Project No. 201506010002, and the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing.

References

[1] M.A. Abramson, Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm, Optim. Eng. 5 (2) (2004) 157–177.
[2] R.D. Al-Dabbagh, F. Neri, N. Idris, M.S. Baba, Algorithm design issues in adaptive differential evolution: review and taxonomy, Swarm Evolut. Comput. (2018) press.
[3] T. Bäck, U. Hammel, H.-P. Schwefel, Evolutionary computation: comments on the history and current state, IEEE Trans. Evolut. Comput. 1 (1) (1997) 3–17.
[4] A. Banitalebi, M.I.A. Aziz, Z.A. Aziz, A self-adaptive binary differential evolution algorithm for large scale binary optimization problems, Inf. Sci. 367–368 (1) (2016) 487–511.
[5] W.-N. Chen, J. Zhang, H.S.-H. Chung, et al., A novel set-based particle swarm optimization method for discrete optimization problems, IEEE Trans. Evolut. Comput. 14 (2) (2010) 278–300.
[6] Z. Chen, X. Jiang, J. Li, S. Li, L. Wang, PDECO: parallel differential evolution for cluster optimization, J. Comput. Chem. 34 (12) (2013) 1046–1059.
[7] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evolut. Comput. 15 (1) (2011) 4–31.
[8] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution: an updated survey, Swarm Evolut. Comput. 27 (2016) 1–30.
[9] K. Deb, M. Goyal, A combined genetic adaptive search (GeneAS) for engineering design, Comput. Sci. Inf. 26 (1996) 30–45.
[10] K. Deb, M. Goyal, A flexible optimization procedure for mechanical component design based on genetic adaptive search, Mech. Des. 120 (2) (1998) 162–164.
[11] G.G. Dimopoulos, Mixed-variable engineering optimization based on evolutionary and social metaphors, Comput. Methods Appl. Mech. Eng. 196 (4–6) (2007) 803–817.
[12] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evolut. Comput. 1 (1) (1997) 53–66.
[13] L. dos Santos Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, Expert Syst. Appl. 37 (2) (2010) 1676–1683.
[14] E.N. Dragoi, V. Dafinescu, Parameter control and hybridization techniques in differential evolution: a survey, Artif. Intell. Rev. 45 (4) (2016) 447–470.
[15] A.H. Gandomi, X.S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (1) (2013) 17–35.
[16] L. Gao, A. Hailu, Comprehensive learning particle swarm optimizer for constrained mixed-variable optimization problems, Comput. Intell. Syst. 3 (6) (2010) 832–842.
[17] Y.-F. Ge, et al., Distributed differential evolution based on adaptive mergence and split for large-scale optimization, IEEE Trans. Cybern. 48 (7) (2017) 2166–2180.
[18] M. Ghasemi, M.M. Ghanbarian, S. Ghavidel, S. Rahmani, E.M. Moghaddam, Modified teaching learning algorithm and double differential evolution algorithm for optimal reactive power dispatch problem: a comparative study, Inf. Sci. 278 (10) (2014) 231–249.
[19] Y.-J. Gong, J. Zhang, Y.-C. Zhou, Learning multimodal parameters: a bare-bones niching differential evolution approach, IEEE Trans. Neural Netw. Learn. Syst. 29 (7) (2017) 2944–2959.
[20] S. Iliya, F. Neri, Towards artificial speech therapy: a neural system for impaired speech segmentation, Int. J. Neural Syst. 26 (6) (2016) #1650023.
[21] Y.-H. Jia, et al., A dynamic logistic dispatching system with set-based particle swarm optimization, IEEE Trans. Syst. Men. Cybern. Syst. (2018) in press.
[22] A.H. Kashan, An efficient algorithm for constrained global optimization and application to mechanical engineering design: league championship algorithm, Comput. Des. 43 (12) (2011).
[23] A. Kaveh, S. Talatahari, An improved ant colony optimization for constrained engineering design problems, Eng. Comput. 27 (1) (2010) 155–182.
[24] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[25] M. Kokkolaras, C. Audet, J. Dennis Jr, Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system, Optim. Eng. 2 (1) (2001) 5–29.
[26] J. Lampinen, I. Zelinka, Mixed integer-discrete-continuous optimization by differential evolution, in: Proceedings of the Fifth International Conference on Soft Computing, 1999, pp. 77–81.
[27] T. Liao, K. Socha, M. de Oca, et al., Ant colony optimization for mixed-variable optimization problems, IEEE Trans. Evolut. Comput. 18 (4) (2014) 503–518.
[28] Y. Lin, Y.-J. Gong, Z.-H. Zhan, J. Zhang, A discrete multiobjective particle swarm optimizer for automated assembly of parallel cognitive diagnosis tests, IEEE Trans. Cybern. (2018) in press.

[29] Y. Liu, W.-N. Chen, Z.-H. Zhan, et al., A set-based discrete differential evolution algorithm, in: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2013, pp. 1347–1352.
[30] Z. Michalewicz, Genetic Algorithms + Data Structure = Evolution Programs, Springer, 1992.
[31] E. Mininno, F. Neri, F. Cupertino, D. Naso, Compact differential evolution, IEEE Trans. Evolut. Comput. 15 (1) (2011) 32–54.
[32] F. Neri, G. Iacca, E. Mininno, Disturbed exploitation compact differential evolution for limited memory optimization problems, Inf. Sci. 181 (12) (2011) 2469–2487.
[33] F. Neri, E. Mininno, Memetic compact differential evolution for cartesian robot control, IEEE Comput. Intell. Mag. 5 (2) (2010) 54–65.
[34] F. Neri, V. Tirronen, Recent advances in differential evolution: a review and experimental analysis, Artif. Intell. Rev. 3 (1) (2010) 61–106.
[35] Q.-K. Pan, L. Wang, L. Gao, W.D. Li, An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers, Inf. Sci. 181 (3) (2011) 668–685.
[36] S. Praharaj, S. Azarm, Two-level non-linear mixed discrete continuous optimization-based design: an application to printed circuit board assemblies, Adv. Des. Autom. 1 (44) (1992) 307–321.
[37] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput. Des. 43 (3) (2011) 303–315.
[38] V. Santucci, M. Baioletti, A. Milani, Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion, IEEE Trans. Evolut. Comput. 20 (5) (2016) 682–694.
[39] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[40] P.N. Suganthan, N. Hansen, J.J. Liang, et al., Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, KanGAL Report, 2005.
[41] R. Tanabe, A. Fukunaga, Evaluating the performance of SHADE on CEC 2013 benchmark problems, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2013), 2013, pp. 1952–1959.
[42] M.F. Tasgetiren, P.N. Suganthan, Q.K. Pan, An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem, Appl. Math. Comput. 215 (9) (2010) 3356–3368.
[43] A. Trivedi, D. Srinivasan, S. Biswas, T. Reindl, A genetic algorithm – differential evolution based hybrid framework: case study on unit commitment scheduling problem, Inf. Sci. 354 (1) (2016) 275–300.
[44] N. Turkkan, Discrete optimization of structures using a floating point genetic algorithm, in: Proceedings of the Annual Conference of the Canadian Society for Civil Engineering, Moncton, N.B., Canada, 2003, pp. 4–7.
[45] H. Wang, S. Rahnamayan, Z. Wu, Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems, J. Parallel Distrib. Comput. 73 (1) (2013) 62–73.
[46] J. Wang, Z. Yin, A ranking selection-based particle swarm optimizer for engineering design optimization problems, Struct. Multidiscip. Optim. 37 (2008) 131–147.
[47] Z.-J. Wang, et al., Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems, IEEE Trans. Evolut. Comput. (2018) in press.
[48] B. Xin, J. Chen, J. Zhang, H. Fang, Z.H. Peng, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy, IEEE Trans. Syst. Man Cybern. Part C 42 (5) (2012) 744–767.
[49] Y. Xue, Y. Zhuang, T. Ni, et al., Self-adaptive learning based discrete differential evolution algorithm for solving CJWTA problem, Syst. Eng. Electron. 25 (1) (2014) 59–68.
[50] W.-J. Yu, J.-Y. Ji, Y.-J. Gong, Q. Yang, J. Zhang, A tri-objective differential evolution approach for multimodal optimization, Inf. Sci. 423 (2018) 1–23.
[51] X. Yu, et al., Set-based discrete particle swarm optimization based on decomposition for permutation-based multiobjective combinatorial optimization problems, IEEE Trans. Cybern. 48 (7) (2017) 2139–2153.
[52] Z.H. Zhan, et al., Cloudde: a heterogeneous differential evolution algorithm and its distributed cloud version, IEEE Trans. Parallel Distrib. Syst. 28 (3) (2017) 704–716.
[53] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evolut. Comput. 13 (5) (2009) 945–958.
[54] G. Zhang, H. Rong, F. Neri, M.J. Perez-Jimenez, An optimization spiking neural p system for approximately solving combinatorial optimization problems, Int. J. Neural Syst. 24 (5) (2014) #1440006.