Noname manuscript No.
(will be inserted by the editor)

Adaptation of Operators and Continuous Control Parameters in Differential Evolution for Constrained Optimization

Saber Elsayed[1*], Ruhul Sarker[1], Carlos Coello Coello[2] and Tapabrata Ray[2]

Received:01/2016 / Revised:06/2016, 11/2016,5/2017 / Accepted:xxx

Abstract A large number of differential evolution algorithms has been proposed in recent years, many of which have been used to solve mainly unconstrained problems. However, similar to other evolutionary algorithms, their performances are highly dependent on their search operators and control parameters. Although many investigations have been conducted to ensure appropriate choices of these operators and parameters, the task is recognized as tedious. In this research, a differential evolution algorithm, which includes a new mechanism for automatically selecting the best combinations of parameters (amplification factor, crossover rate, and population size) as well as search operators, is developed. Instead of choosing discrete values for the amplification factor and crossover rate from a given set of values, this study adaptively selects them from some given continuous ranges and, furthermore, proposes a new methodology for handling constraints. The performance of the algorithm is assessed using a well-known set of constrained problems, with the experimental results demonstrating that it is superior to state-of-the-art algorithms.

1 Introduction

Many real-world decision processes involve solving optimization problems, that is, finding the best solutions in their feasible spaces which, as they are bounded by constraint functions, are recognized as constrained optimization problems (COPs). In a COP, it is necessary to optimize an objective function subject to satisfying a set of constraints. A COP may contain different types of variables, such as real, integer and discrete, and may have equality and/or inequality constraints. These objective and constraint functions can be linear or nonlinear, continuous or discontinuous and uni-modal or multi-modal. The feasible region of such a problem can be either a tiny or a significant portion of the search space and, moreover, either a single bounded region or a collection of multiple disjointed regions or, as happens in some practical problems, even unbounded. The optimal solution may exist on either the boundary or in the interior of the feasible region. Also, high dimensionality due to large numbers of variables and constraints may add further complexity to solving COPs [16]. These different characteristics have made COPs a challenging research area in the optimization domain. In this paper, single-objective COPs are considered, with a COP formally expressed as:

$$\text{minimize } f(\overrightarrow{x})$$

$$\text{subject to: } g_k(\overrightarrow{x}) \leq 0,\ k = 1, 2, \ldots, K$$
$$h_e(\overrightarrow{x}) = 0,\ e = 1, 2, \ldots, E$$

$$\underline{x}_j \leq x_j \leq \overline{x}_j,\ j = 1, 2, \ldots, D \quad (1)$$

where $\overrightarrow{x} = [x_1, x_2, ..., x_D]$ is a vector with D decision variables, $f(\overrightarrow{x})$ the objective function, $g_k(\overrightarrow{x})$ the k^{th} inequality constraint, $h_e(\overrightarrow{x})$ the e^{th} equality constraint and each x_j has a lower limit ($\underline{x}_j$) and an upper limit ($\overline{x}_j$).

*Corresponding author

[1]The authors are with the School of Engineering and Information Technology, University of New South Wales at Canberra, Canberra 2600, Australia, emails: s.elsayed, r.sarker, t.ray@adfa.edu.au;
[2]The author is with the Depto. de Computación, CINVESTAV-IPN, Mexico, email:ccoello@cs.cinvestav.mx

Depending on the complexity of a problem, researchers and practitioners choose either conventional mathematical programming approaches [5] or computational intelligence (CI) methods to solve it. From the many CI methods currently available, genetic algorithms (GAs) [14], differential evolution (DE) [48], evolution strategies (ES) [24] and particle swarm optimization (PSO) [27] are among the most popular.

DE has been successfully used to solve COPs [22, 63, 36, 21, 4, 55, 66, 64, 26]. However, like any other EA, the performance of a DE algorithm depends on the choice of its search operators and control parameters, which normally requires a tedious trial-and-error approach to select the best ones [40, 60]. Furthermore, one search operator and/or set of parameters well suited for a certain set of problems may not work well for another [65]. In addition, even if a set of parameters works well during the early stages of the evolutionary process, it may not perform well in later ones and vice versa. The idea of parameter adaptation was introduced a few decades ago in the context of GA [23] and ES [44] as a way to deal with these problems. Due to its importance for improving the performance of DE, many research studies have proposed different ways of managing dynamic changes of its control parameters [8, 25, 40, 42]. To the best of our knowledge, only a few algorithms reported in the literature have adapted all three control parameters (amplification factor (F), crossover (Cr) and population size (PS)) and DE operators to solve mainly unconstrained problems [54, 7, 52].

In this research, a DE algorithm with automatic selection of operators and parameters for solving COPs is introduced. In it, for each DE control parameter (F and Cr), instead of selecting a value from a set of discrete ones, as assumed in [42] and commonly assumed in other research [30, 3], a range of real values is considered because choosing a combination of parameters from given discrete sets may not ensure the best possible performance of the algorithm. The process begins by assigning different random real values from predefined F and Cr ranges for the individuals in a population which means that each individual will have an independent set of F and Cr values that we recognize as a combination. The success rate of each combination of parameters (the ratio of the number of successful offspring generated by a combination of parameters and the number of times the same combination was used) is recorded for a certain number of generations (a *cycle*) and the better-performing ones are applied for a number of subsequent generations with the number of combinations reduced in subsequent cycles, based on the success rate. Also, the proposed method dynamically adapts PS, in which a pool of PS values is used, with each PS used for a cycle. After all PS values are tested, the one with the best average performance is fixed for a predefined number of generations.

At the beginning of each cycle, the success rates of the current combinations are set to zero and, after a predefined number of cycles, the whole process is restarted from a random assignment. This work is also different from that in [42] which proposes: (1) a set of DE operators from which the algorithm automatically selects the best-performing one during the evolutionary process; (2) a new constraint handling technique whereby the algorithm starts with a subset of the constraints based on the level of constraints violation and gradually considers all of them whereas, in [42], all constraints are handled in any stage of the evolutionary process (more details are provided later); and (3) a different way of adapting PS. Although the algorithm could be designed in a simpler way, in this design : (i) a reduction in the number of combinations is done while the gradual handling of the constraint complexity helps to reduce the computational burden significantly; and (ii) the use of multi-operators and the restart process ensures diversity and better search opportunities as the evolutionary process progresses.

The performance of the proposed algorithm was tested on a well-known set of 36 constrained test problems [32] (18 with 10 dimensions (10D) and 18 with 30D) with different mathematical properties. The results indicated that the proposed algorithm was able to reduce the computational time by 13.29% and 23.06% for the 10D and 30D, respectively compared with the algorithm reported in [42]. Also, the quality of solutions and statistical test results showed its superiority to seven state-of-the-art algorithms.

The rest of this paper is organized as follows: an overview of DE and its operators and parameters is provided in Section 2; the proposed algorithm is illustrated in Section 3; and the experimental results and conclusions are discussed in Sections 4 and 5, respectively.

2 Differential Evolution

DE is a population-based stochastic algorithm for global optimization [47]. In it, initially, a population of random D-dimensional vectors is generated, i.e., $X = \{\overrightarrow{x}_1, \overrightarrow{x}_2, ... \overrightarrow{x}_{PS}\}$ which are randomly chosen to generate a mutant population of size PS, i.e., $V = \{\overrightarrow{v}_1, \overrightarrow{v}_2, ... \overrightarrow{v}_{PS}\}$. Then, each solution ($\overrightarrow{x}_z$), in the current population, is recombined with its corresponding mutant vector ($\overrightarrow{v}$)$_z$ to generate a trial vector ($\overrightarrow{u}_z$), where $z = 1, 2, ...PS$, and every $\overrightarrow{x}_z$ and its corresponding $\overrightarrow{u}_z$ pair-wise compared, with the winning vectors

becoming the parent population (X) in the next generation [39, 46]. Below is a brief description of each step in DE.

- **Initialization:** each individual in the population is represented as a D-dimensional vector in which each variable is generated within its boundaries:

$$x_{z,j} = \underline{x}_j + rand_j(0,1) \times (\overline{x}_j - \underline{x}_j) \forall j = 1, 2, ...D \quad (2)$$

where $rand_j(0,1)$ is a uniform random number within $[0,1]$. Note that DE was initially proposed to solve continuous domain problems and then adapted to solve other types of optimization problems [39].
- **Mutation:** DE generates new solutions that are perturbations of the current ones in which, in its simplest form (DE/rand/1), a mutant vector ($\overrightarrow{v}_z$) is generated by adding a scaled difference between two random solution vectors to a third one (equation 3).

$$\overrightarrow{v}_z = \overrightarrow{x}_{r_1} + F \times (\overrightarrow{x}_{r_2} - \overrightarrow{x}_{r_3}) \quad (3)$$

where $\overrightarrow{x}_{r_1}$, $\overrightarrow{x}_{r_2}$ and $\overrightarrow{x}_{r_3}$ are distinct solution vectors in the current population and none similar to $\overrightarrow{x}_z$, F a positive real number that controls the rate at which the population evolves [39] and $\overrightarrow{x}_{r_1}$ is also called the *base vector*.
There are many variants of this operator, such as DE/best/1 [48], DE/rand-to-best/1 [40] and DE/current-to-best [71]. For more details, readers are referred to [13].
- **Crossover:** there are two well-known crossover schemes, binomial and exponential. The former (sometimes called a uniform or discrete crossover [39]) is conducted on every $j \in [1, D]$ with a predefined crossover probability. In particular, for each j, a uniform random number ($rand_j(0,1)$) is generated. If its value is less than Cr, the trial value ($\overrightarrow{u}_{z,j}$) is copied from the corresponding value from the mutant vector ($\overrightarrow{v}_{z,j}$), otherwise it is copied from the parent vector ($\overrightarrow{x}_{z,j}$) that is

$$u_{z,j} = \begin{cases} v_{z,j} & if\ (rand_j(0,1) \leq cr \text{ or } j = j_{rand}) \\ x_{z,j} & \text{otherwise} \end{cases} \quad (4)$$

$j_{rand} \in 1, 2, ..., D$ is a randomly integer index which ensures that $\overrightarrow{u_z}$ obtains at least one component from $\overrightarrow{v_z}$.
On the other hand, an exponential crossover is similar to a two-point crossover in which the first cut point (l) is randomly selected from a range $[1, D]$ and the second is determined such that L components are copied from $\overrightarrow{v}_z$ [68] as:

$$u_{z,j} = \begin{cases} v_{z,j} \ \forall j = \langle l \rangle_D, \langle l+1 \rangle_D, ..., \langle l+L-1 \rangle_D \\ x_{z,j} \qquad\qquad \forall j \in [1, D] \end{cases} \quad (5)$$

where $\langle l \rangle_D$ denotes a modulo function with a modulus of D and $L \in [1, D]$.
- **Selection:** DE uses a simple one-to-one survivor selection in which, at generation (t), a trial vector ($\overrightarrow{u}_{z,t}$) competes against the target/parent vector ($\overrightarrow{x}_{z,t}$), and the better, in terms of the fitness value and/or constraint violation, is considered as a vector in the new population in the next generation ($t+1$).

2.1 Operators and parameters: a brief review

DE has two main operators (mutation and crossover) and three basic parameters (F, Cr, and PS). These three parameters may be set by using deterministic, adaptive or self-adaptive techniques. In addition, the specialized DE algorithms may use additional operators and parameters, such as constraint handling operator and parameters to control self-adaptive techniques. So, in this subsection, a brief review of recent studies on the adaptation of DE operators and parameters is provided.

2.1.1 DE operators

Over the last two decades, a great deal of work has been undertaken on dealing with DE search operators. However, a general conclusion accepted among researchers is that no single DE operator is suitable for all optimization problems. Consequently, researchers have been motivated to propose multi-operator DE and multi-method frameworks; for instance, the multi-method-based framework introduced in [59] a few years ago in which the authors proposed using five optimization methods (the covariance matrix adaptation (CMA) ES, GA, PSO, DE and parental-centric recombination operator (PCX)) in a single framework, and a self-adaptive strategy for automatically controlling the number of new individuals to be generated by each algorithm. On a set of real-parameter unconstrained problems, this method showed its superiority over many others. This idea was also adopted using a single optimizer, such as DE [40] and GA [16]. Below are some examples related to DE.

Tasgetiren and Suganthan [53] proposed a multi-population DE algorithm for solving real-parameter COPs in which each sub-population conducted its

search in parallel with a regrouping schedule every predefined number of generations. They applied two mutation operators (DE/rand/1 and DE/best/1) with a fixed probability of 0.5 to each, and the algorithm produced encouraging results. Qin et al. [40] proposed a self-adaptive DE algorithm (SaDE) in which the choice of two control parameters, F and Cr, was not required to be pre-specified. In it, four mutation strategies were used and each individual in the population assigned to one of them based on a given probability. Then, the selection probability of each operator was updated based on its success and failure rates during previous generations (a learning period). The algorithm was tested on a set of unconstrained problems, and showed good performance.

Caraffini et al. [10] proposed a super-fit multi-adaptive DE for solving unconstrained problems in which four DE operators with equal probability were used. Then, based on the normalized relative fitness improvement and normalized distance to the best individual, each operator's probability was updated. In addition, F was generated using a Cauchy distribution and Cr based on a normal distribution, with both parameters adapted during the evolutionary process and CMA-ES used as a local search engine.

Elsayed et al. [16] proposed a general framework that divided the population into four sub-populations, each of which used a combination of search operators. During the evolutionary process, the sub-population sizes were adaptively varied based on the success of each operator calculated according to changes in fitness values, constraint violations and feasibility ratios. The algorithm performed well on a set of constrained problems and then the authors extended and improved it in [17, 18]. Elsayed et al. [19] proposed two novel DE variants, each of which utilized the strengths of multiple mutation and crossover operators to solve 60 constrained problems, obtaining results superior to those from state-of-the-art algorithms.

Zamuda and Brest [69] introduced an algorithm that employed two mutation strategies in jDE [8], with the population size adaptively reduced during the evolutionary process based on their earlier technique proposed in [7]. The algorithm was tested on 22 real-world applications and performed better than two other algorithms. It was then extended by embedding a self-adaptation mechanism for parameter control [9][70], in it, whereby the population was divided into sub-populations to apply more DE strategies and a population diversity mechanism. The mutation strategies used depended on the population size which was reduced as the number of function evaluations increased.

Following their original work proposed in [56], Tvrdíkand and Poláková [57] introduced a DE algorithm for solving a set of COPs. In it, with a predefined probability, one set of control parameters was selected from 12 available sets and, during the evolutionary process, the probability of selecting each set was updated based on its success rate in the previous steps. The algorithm was evaluated using a set of unconstrained problems and showed to have competitive performance [58].

Wang et al. [61] introduced a composite DE algorithm (CoDE) in which, in each generation, a trial vector was generated by randomly combining three DE variants with three control parameter settings. The algorithm performed well on a set of unconstrained test problems. Mallipeddi et al. [33, 30] proposed a framework that used a mix of mutation strategies and discrete control parameters in DE to solve unconstrained problems. In it, a pool of different mutation strategies, along with a pool of values for each control parameter, coexisted during the entire evolutionary process and competed to produce new individuals. Also, the authors proposed using a mix of four CHTs based on a DE algorithm (ECHT-DE) [31, 30] to solve COPs in which four populations were initialized, each using a different CHT. In addition, a mix of mutation strategies and F and Cr values, along with two mutation strategies, were employed and the algorithm [30] ranked second in the CEC2010 competition for solving COPs. However, our proposed algorithm is different from that in [30] in that it uses: (1) a single population instead of four sub-populations; (2) continuous values of F and Cr instead of only discrete values; (3) a different mechanism for selecting combinations of parameters and operators and reducing the number of combinations; and (4) a new mechanism for handling constraints.

In [38], the authors proposed a self-adaptive competitive variant of DE, with opposite-based optimization and an adaptively controlled random search used to enhance the search of the feasible region. They also studied the effects of different mutation operators and search strategies. This algorithm showed competitive performance in comparison with other state-of-the-art algorithms. However, many questions regarding its design and selection of mutation operators, as well as the enhanced techniques used to find feasible solutions, still remain open. An improved adaptive DE (ACDE) algorithm, in which a mechanism was used to reduce the population size, and four mutation strategies with values of F and Cr using a Cauchy distribution, was introduced in [11].

Wang et al. [62] introduced a new replacement mechanism to reduce the greediness of the feasibil-

ity rule. The algorithm used two mutation operators (DE/rand-to-best/1 and DE/current-to-rand/1), each of which was applied with a probability of 0.5, with the binomial crossover was only used with the first mutation. Also, a mutation strategy was used when all the individuals in the population are infeasible. The algorithm was tested on two sets of constrained problems and showed competitive, if not better, performance in comparison to other algorithms.

2.1.2 DE parameters

As previously mentioned, because the selection of DE parameters plays a pivotal role in its success, many researchers have proposed techniques for adapting them.

In [48], it was suggested that PS be within [5D-20D] and F be set to a value of 0.5 . In another research study [41], setting $F \in [0.4 - 0.95]$ was recommended. Self-adapting F using DE operators was introduced in [1] and then modified in [16]. In [40], F was randomly generated using an independent normal distribution, with its mean initially set to a value of 0.5 and its standard deviation fixed at 0.1. Then, Cr was re-generated after a predefined number of generations.

Liu and Lampinen [28] proposed to self-adaptively control F and Cr using the concept of fuzzy logic. Brest et al. [8] introduced a self-adaptation method for F and Cr in which each individual in the population was assigned a different combination of their values. Zhang et al. [71] proposed an adaptive DE (JADE) in which, at each generation, Cr_z was independently generated using $N(\overline{Cr}, Cr_\sigma = 0.1)$ with $\overline{Cr}$ initially set to a value of 0.5 and then it was dynamically updated. Similarly, F_z was generated according to a Cauchy distribution with a location parameter ($\overline{F}$), the initial value of which was 0.5, and a scaling parameter of 1 and, at the end of each generation, $\overline{F}$ was updated. As an improved version of JADE, Tanabe and Fukunaga [51, 50] proposed a success-history-based adaptive DE (SHADE) in which, instead of generating new control parameter settings based on some distribution around a single pair of parameters ($\overline{Cr}, \overline{F}$), historical memories ($M_{Cr}, M_F$) which stored sets of values of Cr and F, respectively, were used. As it performed well in earlier generations, it generated new pairs of Cr and F by directly sampling the parameter space close to one of the stored pairs. This algorithm was tested on the CEC2013 unconstrained problems and performed better than other state-of-the-art algorithms. It was then improved by adapting the population size [52] by setting the initial population size to a large value and then linearly reducing it to a small one at the end of the evolutionary process. It was tested on a set of unconstrained problems and was found to have a superior performance with respect to many other algorithms.

Mallipeddi et al. [29] introduced a parallel populations-based DE algorithm, with the number of function evaluations (FEs) assigned to each population self-adapted based on the number of better individuals obtained in the previous generation. In [67], an adaptive DE algorithm with the 'lbest/1' mutation strategy, based on the greedy DE/best/1 strategy and a two-level adaptive parameter control scheme, was introduced. However, in it, the population was mutated under the guidance of multiple locally best individuals. It was analyzed using a set of unconstrained problems which showed its capability to outperform state-of-the-art DE variants for different kinds of optimization problems although its complexity was high. Sarker et al. [42] proposed a DE algorithm that used a mechanism for dynamically selecting the best-performing combinations of the parameters Cr and F for a problem during the course of a single run. Its performance was judged based on its capability to solve three well-known sets of optimization test problems (two constrained and one unconstrained), with the results demonstrating that it was superior to other state-of-the-art algorithms.

It is worthy to mention that there are other methods introduced in the literature to find the best configuration of parameters and/or optimization algorithms [6, 20]. For instance, F-Race and iterated F-Race [6] inspired from racing algorithms in machine learning. Their main idea is to evaluate a given set of candidate configurations iteratively on a stream of instances, and when there is enough statistical evidence against some candidate configurations, these are eliminated, and the race continues only with the surviving ones. The F-Race and iterated F-Race were used for offline configuration of parameterized algorithms.

3 DE with Automatic Adaptation of Operators and Control Parameters

In this section, the algorithm proposed in this study (DE-AOPS) is defined and the constraint-handling technique developed is described.

3.1 DE-AOPS

This research aims to find the most appropriate parameters and search operators during an optimization process. Firstly, the range of each of two DE parameters (F and Cr) is divided into a number of segments (with each segment representing a small range of continuous values). The upper and lower bounds of the segments

are given as sets of discrete values represented as F_{set} and Cr_{set}. For each individual in the evolutionary process, one real value for F and one for Cr are randomly assigned from each segment so that individuals have parameter values from different combinations of segments. The success rate of each combination of segments is recorded for a certain number of generations, and the better-performing combinations applied for a number of subsequent generations (known as a cycle). Depending on the success rate (rank) of each combination, the number of combinations is reduced in subsequent cycles. At the beginning of each cycle, the success rates of the current combinations are set to zero and, after a pre-defined number of cycles, the whole process is repeated, starting from random assignments of parameter values. At the same time, the algorithm emphasizes the best-performing operator during the evolutionary process, with multiple operators considered through similar steps to those for the parameters. Also, PS plays a major role in the success of a DE algorithm [29]. However, one size may be good for one problem but may not suit another, i.e., a small value of PS can lead to a fast convergence rate in solving one problem but may lead the algorithm to suffer in another type of problems. Therefore, the DE-AOPS dynamically determines the most suitable PS among a set of values. Details of the steps in the algorithm are discussed below.

3.1.1 Continuous scheme for parameters

In many adaptive DE algorithms [42, 33], a set of discrete values between 0 and 1 is considered for both F_{set} and Cr_{set}; for example, $F_{set} = \{0.8, 0.9, 1.0\}$ means that the value of F is either 0.8, 0.9 or 1.0. In this case, any real values within a given range or any two consecutive discrete values, i.e., between 0.8 and 0.9 and/or between 0.9 and 1.0, are not considered. This was the motivation for proposing a real parameter adaptation process in this paper, with the expectation that it may provide better performance. This research considers a range of parameters defined by their upper ($\overline{S}$) and lower ($\underline{S}$) bounds from which any real value can be selected. Also, instead of using a single continuous range, multiple disjointed segments may be considered. For convenience of implementing the adaptive process, one continuous range is divided into multiple segments; for example, a range from 0.8 to 1.0 is split into two segments, such that $0.8 \leq F_1 < 0.9$ and $0.9 \leq F_2 < 1.0$, i.e., $F_1 = \underline{S} + rand \times (\overline{S} - \underline{S})$, where $\underline{S} = 0.8$, $\overline{S} = 0.9$ and $rand$ is a uniform random number within $[0.0, 1.0]$. Similarly, F_2 can be calculated considering $\underline{S} = 0.9$ and $\overline{S} = 1.0$.

3.1.2 Multiple DE operators

As previously discussed, it has been proven that the relative performance of a DE operator may vary during the evolutionary process, that is, one operator may work well in the early (or some) stages of the search process and perform poorly in later (or other) stages or vice versa [16]. Also, a DE operator may work well for a specific problem but badly for another. This encourages the use of multiple DE operators, bearing in mind that more emphasis should be placed on the better-performing one in each stage of the evolutionary process, as described below.

3.1.3 Description of algorithm

The main steps in DE-AOPS are presented in Algorithm 1.

Initially, three sets of the control parameters are defined as F_{set}, Cr_{set} and PS_{set}, where F_{set} and Cr_{set} contain nf and ncr discrete values to represent the segment bounds, respectively (where each segment is a range of continuous values) while $PS_{set} = \{PS_1, PS_2, ..., PS_{nps}\}$ is a set of discrete values. A set of different operators ($SO_{set} = \{SO_1, SO_2, ..., SO_{nso}\}$) is also considered, and nf, ncr, nps and nso refer to the cardinality of the sets of F, Cr, PS and SO, respectively. Note that the total number of combinations (NoC_{total}) for F and Cr is equal to ($nf \times ncr$) and PS_i is assumed to be larger than $PS_{i-1} \forall i = nps, nps-1, ..., 2$. Also, a population of size PS_{nps-1} is a subset of that population with a size of PS_{nps} and so on.

In lines 3 to 6 in Algorithm 1, PS_{nps} random vectors are generated. Then, each is evaluated, the number of current fitness evaluations (cfe) is increased by 2 because evaluating the constraints is counted as one fitness evaluation [1], and the population is sorted from best to worst based on the fitness function and constraint violation values.

As cfe is less than the maximum number of fitness evaluations (cfe_{max}), each individual in the population ($\overrightarrow{x_z}$) is assigned random F and Cr segments (F_z and Cr_z, respectively). Then, F_z and Cr_z are converted to values within their segment ranges, as described in section 3.1.1. At the same time, one search operator, from the SO_{set}, is assigned to each individual. To clarify, each combination of F and Cr is assigned to at least a single individual and all search operators are strictly assigned to the same number of individuals (lines 8 to 10 in Algorithm 1).

[1] This was a condition in the CEC2010 competition, which is used in this paper to assess the performance of DE-AOPS

Algorithm 1 General framework of DE-AOPS

1: $PS_{set} \leftarrow PS_1, PS_2, ..., PS_{nps}$; $F_{set} \leftarrow F_1, F_2, ..., F_{nf}$; $Cr_{set} \leftarrow Cr_1, Cr_2, ..., Cr_{ncr}$; $SO_{set} \leftarrow SO_1, SO_2, ..., SO_{nso}$; $t \leftarrow 1$; and set con_{ss};
2: $i \leftarrow nps$; $PS \leftarrow PS_i$; $period \leftarrow 0$; $iter \leftarrow 1$; $cfe \leftarrow 0$;
3: Generate an initial random population. The variables of each individual ($\overrightarrow{x_z}$) must be within their boundaries;
4: Calculate the fitness value and constraint violation of ($\overrightarrow{x_z}$);
5: $cfe \leftarrow 2 \times PS$ as evaluating the constraints is counted;
6: Sort the whole population, based on the superiorty of feasible solutions method [15].
7: **while** $cfe < cfe_{max}$ **do**
8: Each individual is assigned a random combination of parameter segments F and Cr;
9: Convert discrete segments of F and Cr to continuous values (sub-section 3.1.1).
10: Randomly assign each search operator to the same number of individuals;
11: Update con_{ss} if required.
12: Evolve all individuals as described in Algorithm 2 considering con_{ss} constraints (subsection 3.1.4).
13: **if** $(period\%CS) = 0$ and $(period < \eta \times CS)$ **then**
14: Select the best half combinations to be used in the subsequent cycle;
15: Select the best half SO to be used in the subsequent cycle;
16: $com_{y,suc} \leftarrow 0$; $SO_{p,suc} \leftarrow 0$;
17: **end if**
18: **if** $(period\%CS) = 0$ and $(i > 0)$ **then**
19: Calculate $Rank_{ps_i}$ based equation (8);
20: **if** $1 < i \leq nps$ **then**
21: Archive the worst $ps_i - ps_{i-1}$ individuals ;
22: $i \leftarrow i - 1$;
23: $PS \leftarrow PS_i$;
24: **end if**
25: **end if**
26: **if** $(period = nps \times CS)$ and $(i = 0)$ **then**
27: $PS \leftarrow$ the one with the best $Rank_{ps_i}$;
28: Use individuals from the archive as required, i.e., if the best $PS > PS_1$;
29: **end if**
30: **if** $(period = \eta \times CS)$ or no success was recorded for all combinations **then**
31: Reset all parameters to their initial values (step 1 and 2);
32: Retrieve the remaining individuals from the archive, and hence clear the archive.
33: **end if**
34: $t \leftarrow t + 1$; and go to step 7;
35: **end while**

Algorithm 2 Evolving Individuals

1: **for** $z = 1 : PS$ **do**
2: Generate a new individual ($\overrightarrow{u_z}$) using its assigned F_z, Cr_z and SO_z;
3: Calculate the constraints violation $\Theta(\overrightarrow{u_z})$;
4: **if** $\Theta(\overrightarrow{u_z}) > 0$ // the individual is infeasible **then**
5: $cfe \leftarrow cfe + 1$;
6: Fitness value ($fit(\overrightarrow{u_z})$) $\leftarrow fit(\overrightarrow{x_z})$;
7: **else if** $\Theta(\overrightarrow{u_z}) = 0$ // the individual is feasible **then**
8: Calculate the fitness value ($fit(\overrightarrow{u_z})$);
9: $cfe \leftarrow cfe + 2$;
10: **end if**
11: **if** $\overrightarrow{u_z}$ is better than $\overrightarrow{x_z}$ **then**
12: $\overrightarrow{u_z}$ is survived to the next generation; $com_{y,suc} \leftarrow com_{y,suc} + 1$; $SO_{p,suc} \leftarrow SO_{p,suc} + 1$;
13: **end if**
14: $period \leftarrow period + 1$;
15: Update and sort the new population.
16: **end for**

For each $\overrightarrow{x_z}$, a new offspring ($\overrightarrow{u_z}$) is generated using its assigned combination of parameters and operators, and evaluated based on the objective function and subset of constraints (con_{ss}). Note that con_{ss} is periodically updated (line 11) and is described in section 3.1.4. If $\overrightarrow{u_z}$ is better than $\overrightarrow{x_z}$, it survives to the next generation and the success of the corresponding combination ($com_{y,suc}$) is increased by 1, i.e., $com_{y,suc} = com_{y,suc} + 1$, where $y = 1, 2, ..., NoC_{total}$. The success of a search operator ($SO_{p,suc}$) is also incremented by 1, where $p = 1, 2, ..., nso$. It is worth mentioning that, to reduce the number of fitness evaluations, if $\overrightarrow{u_z}$ is infeasible, the objective value is not calculated and, as it takes the fitness value of its parent, cfe is only increased by 1. On the other hand, if $\overrightarrow{u_z}$ individual is feasible, as its fitness value is calculated, cfe is increased by 2 (line 12 in Algorithm 1).

As shown in lines 13 to 17 in Algorithm 1, the above-mentioned processes are repeated for a predefined number of CS generations, after which the number of combinations is reduced by half, i.e., the better combinations of F and Cr are preserved based on their ranks. The ranking of any combination ($Rank_y$) is calculated using equation (6), where NI_y is the number of individuals updated by a combination (y). The higher the value of $Rank_y$, the better-performing the combination.

$$Rank_y = \frac{com_{y,suc}}{NI_y} \tag{6}$$

Similarly, the number of search operators is reduced by half, but if nso becomes equal to 1, no reduction occurs. This is decided using equation (7), where NI_{op} is the number of individuals updated by the search operator (SO_{op})

$$Rank_p = \frac{SO_{p,suc}}{NI_{op}} \tag{7}$$

Note that at the end of each cycle, the ranks of all combinations and operators are reset to zero.

The process of changing the population sizes is as follows (lines 18 to 25 in Algorithm 1): at the end of every CS generations, the better-performing PS_{nps-1} individuals are kept in the population while the remaining ones ($PS_{nps} - PS_{nps-1}$) are transferred to an archive/memory. The ranking of PS_{nps} is computed as the average performance per individual in the popula-

tion for all parameter combinations over CS generations as:

$$Rank_{ps_i} = \frac{\sum_1^{CS} \sum_1^{NoC_{total}} com_{y,suc}}{PS_i \times CS} \quad (8)$$

This process continues to calculate the ranking of each population size. After $CS \times nps$ generations, PS is fixed to the size with the best ranking, and the individuals stored in the archive/memory can be used if required, i.e., if the best $PS > PS_1$ (lines 26 to 29 in Algorithm 1).

At every $\eta \times CS$ generations (the selection of η is based on [42]), all the parameters are reset to their initial values. The individuals stored in the archive/memory are retrieved to set PS to its initialized value (PS_{nps}), and hence the archive/memory is cleared (lines 30 to 33 in Algorithm 1).

To elucidate, a case with $nps = 3$ (75, 50 and 25 individuals), $nso = 4$, $CS = 20$ and $NoC_{total} = 50$ is considered. Initially, PS is set to a value of 75, and then this population of individuals will evolve for 20 generations with 50 combinations and 4 search operators, then the best 50 individuals will evolve for 20 more generations with the best 25 combinations and best 2 search operators, as determined based on their success in the previous 20 generations, and then the population size of 25 will evolve for 20 generations with 13 combinations and the best search operator. Finally, the selected population size (75, 50 or 25) will be fixed, and the number of combinations will be reduced by half. Once the number of generations is $\eta \times CS = 120$, all the parameters are reset to their initial values, i.e., $nps = 3$, $nso = 4$, $CS = 20$, and $NoC_{total} = 64$. This process continues until a stopping criterion is met.

3.1.4 New cumulative constraints handling method

It is well known that an increase in the number of constraints in any problem will make it more complex and, therefore, any EA will take a higher computational effort than for a simpler problem to reach an optimal solution. Motivated by this, a new way of handling constraints is proposed. It begins by ordering the constraints according to the sum of the constraint violations of all the individuals in the initial population from the most violated to least violated constraint or vice versa. The evolutionary process starts with a subset of the constraints (con_{ss}), instead of all, for a predefined number of generations and, as a new subset is then added to the current one, the technique tries to reach the feasible region of both the previous and new subsets of constraints. This process continues until all the constraints are incorporated, and the final feasible region is reached.

The selection process between any offspring and its parent follows one of three scenarios [15]: (1) for two feasible candidates, the fittest one (according to the fitness function) is selected; (2) a feasible point is always better than an infeasible one; and (3) for two infeasible solutions, the one with a smaller sum of constraint violations (Θ) is chosen, where Θ of an individual ($\overrightarrow{x_z}$) is calculated based on equation (9).

$$\Theta_z = \sum_{k=1}^{K} max(0, g_k(\overrightarrow{x_z}) - \delta_k) + \sum_{e=1}^{E} max(0, |h_e(\overrightarrow{x_z})| - \epsilon_e) \quad (9)$$

where $g_k(\overrightarrow{x_z})$ is the k^{th} inequality constraint and $h_e(\overrightarrow{x_z})$ the e^{th} equality constraint. Note that, as some inequality constraints may be difficult, instead of setting the right-hand side of any g_k to zero, a large value (δ) is used at the beginning and then reduced to be zero. This also applies to any equality constraint (h_e), whereby ϵ_e is initialized with a large value and then reduced to 0.0001. Setting the initial value of ϵ is problem dependent, as demonstrated in [34, 36, 45].

The proposed CHT can be better illustrated using the following two-dimensional problem.

$$\text{minimize } f = \sum_{j=1}^{2} x_j^2$$

Subject to

$$g_1 = \prod_{i=1}^{2} x_j \leq 0$$

$$g_2 = \sum_{j=1}^{2} x_j \leq 0$$

$$h_1 = g_3 = \sum_{j=1}^{2} x_j sin(4\sqrt{|x_j|}) = 0$$

$$-10 \leq x_j \leq 10 \forall j = 1, 2 \quad (10)$$

Figure 1a shows a two-dimensional plot of the objective function and all constraints, and Figure 1b shows plots of all the constraints and the contour plot of the objective function (f) and the optimal solution ($f(\overrightarrow{x}^*) = 0$), where $\overrightarrow{x}^* = \{0, 0\}$; 20 random points

are generated per instance (see Figure 1c). Then, the average violation of each constraint of 20 solutions ($\Theta_i = \frac{\sum_{z=1}^{20} \Theta_{i,z}}{20} \forall i = 1,2,3$) is calculated. Based on this example, $\Theta_1 = 13.05$, $\Theta_2 = 2.6$ and $\Theta_3 = 5.56$. If we are interested in dealing with the most violated constraints first and $con_{ss} = 1$, g_1 comes first, followed by g_3 and g_1. Therefore, all 20 points will be evolved considering only g_1 for a predefined number of generations (Figure 1d) which shows that all points satisfy g_1 but may violate the other two constraints. Then, the algorithm will evolve all points considering both g_1 and g_3 up to a predefined number of generations (Figure 1e). Finally, all three constraints are considered for solving the problem until an overall stopping criterion is met and, as is clear in Figure 1f all points converged to the optimum.

We like to mention here that the proposed CHT shares a similarity with the behavioral memory (BM) method proposed by Schoenauer and Xanthakis [43], in which both handle a subset of constraints at a time. However, there are significant differences too as discussed below.

In the BM method, a random population is evolved until a given percentage of the population is feasible considering only the 1st constraint. Then, the population is evolved until a threshold proportion of the population satisfies the following constraint (say g-th). During this phase, the individuals that are not feasible for any of the 1^{st} to $(g-1)^{th}$ constraint disappear from the selection process. After considering all the constraints, the algorithm starts the search process to optimize the objective function. On the other hand, in each phase of our proposed method, a constraint is added to the problem, i.e., in the 2^{nd} phase, the population is evolved considering both the 1^{st} and 2^{nd} constraints, and no individual is removed from the population. Also, the proposed method does not wait until the last phase to apply the search process for optimizing the objective function, i.e., it optimizes the objective function given the set of constraints in a particular phase. Another important difference is that the BM method requires a linear order of all constraints that are processed in turn without any preferences [35]. However, in our method, the constraints are ordered based on their complexity, such as the average level of constraints violation. These differences make our method more flexible, in which a subset constraints may be considered (instead of one at a time) at each phase.

4 Experimental Results

In this section, the computational results obtained by DE-AOPS for the set of CEC2010 constrained problems are presented and analyzed [32]. All algorithms were run 25 times for each test problem, with the stopping criterion run for up to 200,000 and 600,000 FEs for the 10D and 30D problems, respectively, using an algorithm coded in Matlab R2012b[2].

Regarding the parameter values used in this study,

- $SO_{set} = \{\text{DE}_1, \text{DE}_2\}$, where
  1. DE_1: DE/φ-best/1/bin [42]

$$u_{z,j} = \begin{cases} x_{\phi,j} + F_z.(x_{r_1,j} - x_{r_2,j}) \\ \quad if(rand \leq cr_z \text{ or } j = j_{rand}) \\ x_{z,j} \qquad \text{otherwise} \end{cases} \tag{11}$$

  2. DE_2: DE/current-to-ϕbest with archive/1/bin [71]

$$u_{z,j} = \begin{cases} x_{z,j} + F_z.(x_{\phi,j} - x_{z,j} + x_{r_1,j} - \widetilde{x}_{r_3,j}) \\ \quad if(rand \leq cr_z \text{ or } j = j_{rand}) \\ x_{z,j} \qquad \text{otherwise} \end{cases} \tag{12}$$

  where $\varphi = 0.5$, as suggested in [42], $\phi = 0.1$ [71], $r_1 \neq r_2 \neq r_3 \neq z$ are random integer numbers, $\widetilde{x}_{r_2,j}$ randomly chosen from $PS \cup AR$, i.e., the union of PS and the archive AR, which is different from that in Algorithm 1. Initially, the archive was empty, then parent vectors which failed in the selection process were added to it and, once its size exceeded a threshold, $1.4PS$ randomly selected elements were deleted to make space for the newly inserted ones [71]. The reason for using DE_1 was to obtain a balance between diversity and intensification as described in [42], whereas DE_2 had a high convergence rate.
- F_{set} =$\{F_5 \in [0.5-0.6[, F_6 \in [0.6-0.7[, F_7 \in [0.7-0.8[, F_8 \in [0.8-0.9[, F_9 \in [0.9,1[\}$. This means that $nf = 5$.
- Cr_{set} =$\{Cr_2 \in [0.2-0.3[\ , Cr_3 \in [0.3-0.4[, Cr_4 \in [0.4-0.5[, Cr_5 \in [0.5-0.6[, Cr_6 \in [0.6-0.7[, Cr_7 \in [0.7-0.8[, Cr_8 \in [0.8-0.9[, Cr_9 \in [0.9,1[\}$. This means that $ncr = 8$. Therefore, $NoC_{total} = 5 \times 8 = 40$ combinations.
- nps was set to a value of 4 and the minimum PS to a value of 60 with each sub-sequent size increased by 5 for the 10D problems i.e., PS_{set} =$\{60, 65, 70, 75\}$ and by 20 for the 30D problems, i.e. PS_{set} =$\{60, 80, 100, 120\}$, to maintain diversity for higher-dimensional problems. Note that nps should be less than η.

[2] The source code is available upon request

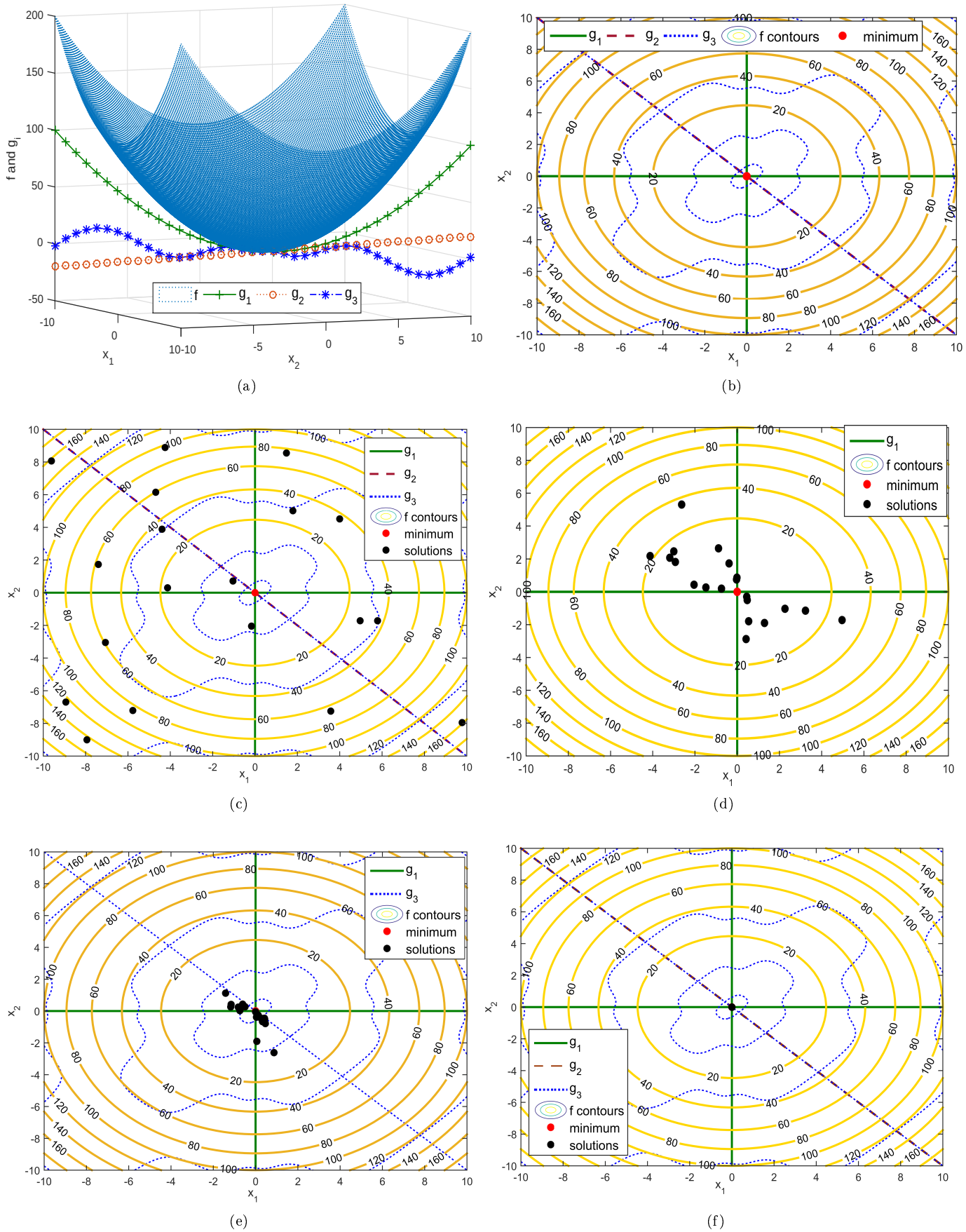

Fig. 1 Possible movements of a population of individuals based on the proposed constraint handling technique

Table 1 Comparison summary of var_1 against var_2 for 30D problems with more than one constraint (values in rows 1 and 2 refer to numbers of test problems for which var_1 better, similar and worse than var_2 based on best and average fitness values obtained, respectively)

Algorithms	Criteria	Better	Similar	Worse
var_1 vs var_2	Best fitness values	0	9	3
	Average fitness values	0	8	4

Table 2 Comparison summary of var_2 against var_3 for 30D problems with more than one constraint (values in rows 1 and 2 refer to numbers of test problems for which var_2 better, similar and worse than var_3 based on best and average fitness values obtained, respectively)

Algorithms	Criteria	Better	Similar	Worse
var_2 vs var_3	Best fitness values	2	9	1
	Average fitness values	4	8	0

- $CS = 25$ generations, $\eta = \frac{log(NoC_{total})}{log(2)} \simeq 6$ [42].
- $con_{ss} = 50\%$ of the number of constraints.

4.1 Effect of proposed CHT

The components of the proposed CHT were analyzed in order to answer the following questions: (1) should the most violated subset of constraints be handled first or vice versa; (2) does the proposed CHT add any benefits to the algorithm; and (3) what is the best subset size to consider?

To answer the first question, two variants of DE-AOPS were run. In the first variant var_1, the most violated constraints and, in var_2, the easiest ones were considered first, with $con_{ss} = 50\%$ in both. Also, both variants were tested on the 30D test problems with more than one constraint, that is, 12 problems. A comparison summary is presented Table 1. Although, from the results obtained, there was not a great deal of difference between the variants, there was a slight bias towards var_2. As it was noticed that var_2 was able to reduce the average FEs to obtain optimal solutions with a tolerance of 0.0001 by 0.25%, it was selected.

To demonstrate the benefit of the proposed method, var_2 was compared with the same algorithm without the proposed CHT, i.e., $con_{ss} = 100\%$, which was called var_3, for solving the same 12 problems, that is, those with more than one constraint. From the comparison summary shown in Table 2, it was found that var_2 was better, especially for the average results obtained, and was able to reduce the average FEs by 0.25%.

Another variant was run by changing con_{ss} to 25% (var_4) and its results were compared with those for var_2, on only test problems with more than 2 constraints, that is, problems C02, C04, C13, C14, C15, C16 and C17. From the comparison summary shown in Table 3, var_2 was found to be superior to var_4 in terms of the quality of solutions obtained although var_4 was 25% faster. It is worth mentioning that C02 was the only problem for which var_4 obtained better mean results. As we seek good results, var_2 was selected and used as the proposed DE-AOPS in this paper.

Table 3 Comparison summary of var_2against $var_4$for 30D problems with more than two constraints (7 problems) (values in rows 1 and 2 refer to numbers of test problems for which $var_2$was better, similar and worse than var_4based on best and average fitness values obtained, respectively).

Algorithms	Criteria	Better	Similar	Worse
var_2 vs var_4	Best fitness values	3	4	0
	Average fitness values	3	3	1

4.2 DE-AOPS VS DE-DPS

In this section, the proposed algorithm is compared with a successful adaptive DE (DE-DPS) with discrete values [42]. Detailed computational results are shown in Appendix A a summary is presented in Table 4.

Considering the best solutions obtained for the 10D test problems, DE-AOPS was better than DE-DPS for only one problem while both algorithms were able to achieve optimal solutions for the remaining 17 problems. Based on the average results, DE-AOPS was better than DE-DPS for 6 problems, and both were similar for 10 problems while DE-AOPS was inferior for C02 and C08.

In reference to the best results obtained for the 30D test problems, those from DE-AOPS were better for 13 test problems, and those from both algorithms were the same for the other 5. Based on the average solutions, DE-AOPS performed best for 14 test problems and was able to obtain the same results as DE-DPS for 2 more while being inferior to DE-DPS for other 2 problems (C02 and C13). The reason for these inferior solutions was the large population sizes considered, as described in Section IV-C. From these results, it was noted that DE-AOPS had the capability to improve results for most test problems, especially the higher-dimensional ones.

Also, the Wilcoxon signed rank test [12] was used to statistically compare the algorithms. Using a significance level of $p = 5\%$, one of three symbols (+, −, and ≈) was assigned, where + means that the 1^{st} algorithm was statistically superior to the 2^{nd}, − that the 1^{st} algorithm was statistically inferior to the 2^{nd} and ≈ that there was no significant difference between the two algorithms. The results in Table 4 show that there was no significant difference between those from the algorithms

Table 4 Comparison summary of DE-AOPS against DE-DPS (values in rows 1 and 2 refer to numbers of test problems for which DE-AOPS better, similar and worse than DE-DPS based on best and average fitness values obtained, respectively with p a probability used to take decision based on Wilcoxon test)

Algorithms	Criteria	10D					30D				
		Better	Similar	Worse	p	Decision	Better	Similar	Worse	p	Decision
DE-AOPS vs DE-DPS	Best fitness values	1	17	0	0.317	$\approx$	13	5	0	0.002	+
	Average fitness values	6	10	2	0.612	$\approx$	14	2	2	0.056	+

for the 10D results, with $p = 10\%$, DE-AOPS was found to be significantly better than DE-DPS based for the 30D ones.

As it is well known that no solution better than the optimum can be obtained, if competing algorithms obtain the same solution (either the optimum or a solution within an acceptable tolerance limit), the only way to assess the superiority of one is to compare their computational burdens. We must mention that DE-AOPS achieved a faster convergence rate, as shown in Figure 2 in which it is clear that it was able to converge quickly to optimal solutions for C03, C07 and C14. It was noted that for all the test problems, DE-AOPS reduced the average number of fitness evaluations required to reach optimal solutions, with a tolerance value of 0.0001, by 13.29% and 23.06% for the 10D and 30D problems, respectively.

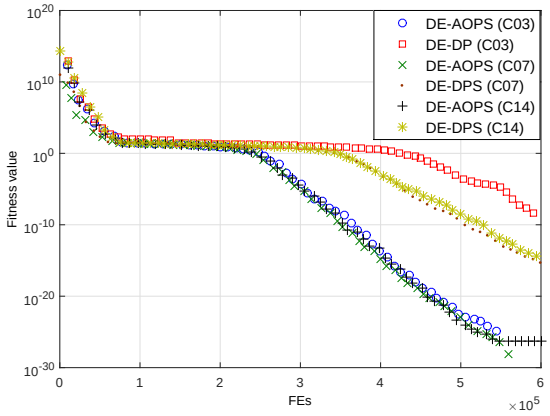

Fig. 2 Convergence plots of DE-AOPS and DE-DPS for C03, C07 and C14 with 30D (the y-axis in log scale)

4.3 Benefits of dynamic change in population sizes

In this subsection, we discuss the benefit of the proposed dynamic adaptation of PS demonstrated by running DE-AOPS with different single PS values of 60, 80, 100 and 120. All the variants were used to solve the 30D test problems, and their average results recorded and ranked from best to worst, i.e., 1 to 5, with a summary presented in Table 5.

Firstly, it was found that DE-AOPS with $PS = 60$ was the best for C02 and C13, and achieved more stable results than all the other algorithms. However, it became stuck in local solutions for the multi-modal problems C01, C03, C07 toC12 and C14 to C15. On the other hand, with $PS = 120$, it was superior to the other variants for those problems, but the worst for others, especially C02 and C13, while with $PS = 100$ it was good for some test problems, such as C03, C07, C08, C14, C16 and C18.

Regarding the feasibility ratio (FR), it was found that all the variants were able to obtain a 100% FR, except DE-AOPS with $PS = 60$ and $PS = 100$ for which both variants obtained a rate of 99.8%. In terms of the average FEs required to obtain optimal solutions, with a tolerance of 0.0001, it was found that DE-AOPS with $PS = 60$ was the best and DE-AOPS with $PS = 120$ the worst. For all the test problems, DE-AOPS with the proposed dynamic mechanism of PS was found to be the best compared to the abovementioned variants with a single PS value, as it provided a balance among all the other variants. From this analysis, one could easily justify including this dynamic mechanism of PS in the proposed algorithm.

Table 5 Ranks of DE-AOPS with $PS = 60$, $PS = 80$, $PS = 100$, $PS = 120$ and dynamic PS (proposed)

Probs.	Dynamic PS	$PS = 60$	$PS = 80$	$PS = 100$	$PS = 120$
C01	2	5	4	3	1
C02	3	1	2	4	5
C03	2	5	3	1	4
C04	1	1	1	1	1
C05	1	1	1	1	1
C06	1	1	1	1	1
C07	2	4	3	1	5
C08	2	5	4	1	3
C09	3	4	5	2	1
C10	2	5	3	4	1
C11	1	5	1	5	1
C12	1	5	2	3	4
C13	3	1	2	4	5
C14	3	5	2	1	4
C15	1	5	2	4	3
C16	1	1	1	1	1
C17	4	5	3	2	1
C18	3	4	2	1	5
Avg.	2.00	3.50	2.33	2.22	2.61
FR	100%	99.8%	100%	99.8%	100%
Avg. FEs	3.37E+05	3.07E+05	3.13E+05	3.25E+05	3.44E+05

Table 6 Ranks of DE-AOPS with nps = 3, 4, 5, 6 based on Friedman test

	nps			
	3	**4**	**5**	6
Ranking	2.86	**2.14**	**2.14**	2.86

4.4 Effect of *nps*

To evaluate the performance of each population size, nps should be less than or equal to η, as discussed in [42]. Therefore, to analyze the effect of this parameter, we ran the algorithm by setting nps =3, 4,5 and 6 and solved the 10D problems. Then, the ranking of each variant based on the average fitness value obtained was calculated by the Friedman test. From the results presented in Table 6, it was found that setting nps to 3 and 4 had best ranking.

4.5 Scaling Analysis

The relationship between the dimensionality of a problem and the average number of FEs required to obtain solutions with a tolerance limit of 0.0001 was derived. Three test problems (C07, C09 andC15) were selected, the mathematical properties of which are shown in Table 7, with their optimal solutions at $f(\overrightarrow{x^*}) = 0$. All the problems were solved using different dimensions, i.e., $D = 5, 10, 15, 20, 25$ and 30 variables. For each D, the algorithm was run 51, times and the average FEs were recorded. It is worth mentioning that only up to 30 decision variables were used due to data availability. Note that to be consistent with the selection of PS_{set} discussed in Section 4, it was assumed that the minimum PS should be 60 and each subsequent PS_i increased by 5 for 5D and 10D (i.e., PS_{set} ={60, 65, 70, 75}), 10 for 15D (i.e., PS_{set} ={60, 70, 80, 80}), 15 for $20D$ (i.e., PS_{set} ={60, 75, 90, 105}) and 20 for $D > 20$ (i.e., PS_{set} ={60, 80, 100, 120}).

Figure **3** shows the average FEs for each dimension. Also, the quadratic regression equations [37] were fitted to help approximate the average FEs required for every D. Hence, the quadratic regression equation for C07 was $FEs = 193.61D^2 + 3480.1D + 10062$ and the coefficient of determination [37] 97.9% (a larger value, with a maximum absolute value of 1, means that the curve is highly fitted to expect future values). For C09, $FEs = 165.74D^2 + 4828.6D + 5240.4$ and the coefficient of determination was equal to 98.1% and, that for C15, $FEs = 219.85D + 3051.9D + 22856$, and the coefficient of determination 97.6%. These encouraging results demonstrated that the algorithm was efficient and scaled well with an increasing number of dimensions.

4.6 Discussion

In this subsection, we discuss the different parameter segment combinations (F and Cr) during the evolutionary process that could lead to good performances which will guide researchers and practitioners in choosing appropriate parameter sets for their problems.

For the best run of each problem, the best segment combination, at every level of combination reduction was recorded. B the x-axis represents FEs and the y-axis is scaled from 2 to 10, where each value represents a segment of F and Cr, as described in Section 4. A summary of the preferred values for Cr and F is presented in Table 8. Generally speaking, there was no single value that was good for all the test problems which was consistent with the motivation for our work.

4.7 Comparison with state-of-the-art algorithms

DE-AOPS is compared with: (1) εDEag [49] (the winner of the CEC2010 competition), (2) ECHT-DE [30], which used an ensemble of CHTs as well as operators and parameters; (3) an adaptive ranking mutation operator-based DE (ECHT-ARMOR-DE) [22], which is an improved version of ECHT-DE; (4) a constrained DE with non-dominated sorting mutation (MS-(μ+λ)-CDE) [63]; (5) a binary real-coded GA (BRGA) [2]; and (6) two versions of SHADE with a linear reduction mechanism (LSHADE), as it has shown its superiority to other algorithms for solving unconstrained problems [52], that is (i) LSHADE with the adaptive penalty CHT (LSHADE-AP), i.e., converting a COP to an unconstrained one; and (ii) LSHADE with the superiority of feasible solutions CHT (LSHADE-SFS). In the second version of LSHADE, the amount of fitness improvement represented in equation 9 in [52] was modified to consider constraint violations so that $\Delta fv_z = |max(0, f(\overrightarrow{x}_{z,iter}) - f(\overrightarrow{u}_{z,iter}))| + |max(0, \Theta(\overrightarrow{x}_{z,iter}) - \Theta(\overrightarrow{u}_{z,iter}))|$. Note that both versions used the same parameter settings as those reported in [52] and the source code was taken from [52] which is available online. Details of the results are presented in Appendix A.

Firstly, DE-AOPS was able to reach 100% FRs for both the 10D and 30D problems. εDEag attained a 100% FR for 35 out of the 36 test instances, while it only obtained a 12% FR for C12 with 30D, the FRs of ECHT-DE, ECHT-ARMOR-DE and MS-(μ+λ)-CDE were less than 100% for the10D and 30D problems, with no exact percentages reported in their papers, BRGA

Table 7 Mathematical properties of C07, C09 and C15

Properties	C07			C09			C15		
	Obj. Func.	Constraints		Obj. Func.	Constraints		Obj. Func.	Constraints	
		equality	inequality		equality	inequality		equality	inequality
Non-separable	√			√			√		
Separable			√		√				√
Multi-modal	√			√	√		√		√
Shifted	√			√	√		√		√
Rotated							√		
No of constraints			1		1				3

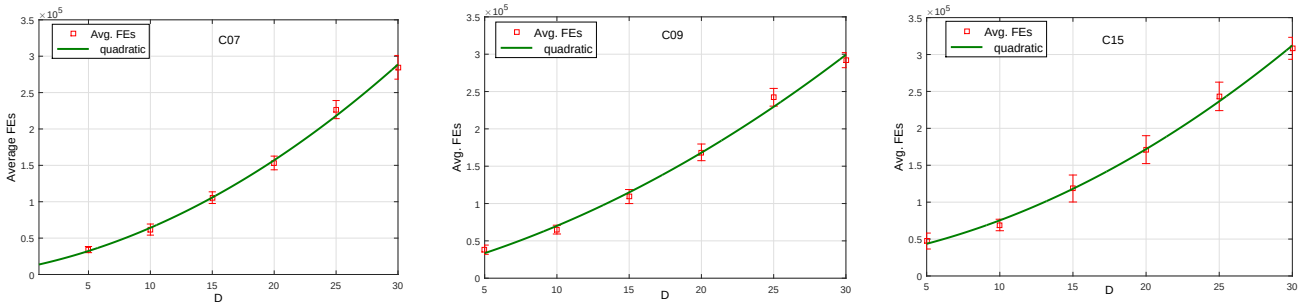

Fig. 3 Average number of function evaluations versus problem dimensions for C07, C09 and C15 using 51 runs (solid green curves represent quadratic regression fitting of data and bars represent standard deviations)

Table 8 Summary of preferred values for F and Cr based on problems' characteristics

Prob.	Characteristics		F	Cr
	Objective function	constraints		
C01and C03	multi-modal and non-separable		$[0.5-0.8]$	Early stages (small values), later stages (high values)
C07, C09, C14 and C18	multi-modal and non-separable	multi-modal separable	$[0.5-0.8]$	Early stages (small values), later stages (high values)
C02and C05	separable	multi-modal separable	$[0.5-0.8]$	$[0.2-0.8]$
C04	separable	separable and non-separable multi-modal equality	$[0.4-0.7]$	$[0.6-1.0]$
C13	separable	separable and non-separable multi-modal inequality	$[0.6-1.0]$	Early stages (small values), later stages (high values)
C17	non-separable unimodal	separable and non-separable	$[0.5-1.0]$	$[0.8-1.0]$
C16	non-separable multi-modal	separable and non-separable constraint	$[0.4-0.7]$	Small values
C06, C08, C10, C11, C12 and C13	rotated functions		$[0.4-0.7]$	Early stages (small values), later stages (high values)

achieved 89% and 81% FRs for the 10D and 30D problems, respectively, and LSHADE-SFS and LSHADE-AP 78% and 77.5% FRs, respectively, for the 10D problems, and 76% and 77%, respectively, for the 30D ones.

A summary of the quality of solutions obtained is provided in Table 9. For the 10D problems, DE-AOPS was superior to all algorithms for the majority of them based on the average results and performed much better than the other algorithms for the 30D test problems.

Based on the statistical analysis, DE-AOPS was found to be superior to ECHT-DE, ECHT-ARMOR-DE and MS-(μ+λ)-CDE considering the average results for the 10D test problems, and better than BRGA, LSHADE-SFS and LSHADE-AP based on both the best and average results. For the 30D test problems, DE-AOPS was significantly better than all the other algorithms.

5 Conclusions

For solving constrained and unconstrained optimization problems, DE has shown good performance in comparison with other EAs. However, as the selection of appropriate search operators and control parameters has been known to be a tedious task, a considerable number of DE algorithms that used an ensemble of search operators and/or adaptive or self-adaptive control parameters has been introduced. In the approach in this study, an adaptation of two search operators, as well as three parameters, was proposed. Multiple search operators and continuous values of the control parameters, in which three different sets of parameter values were initialized for F, Cr, and PS, were used. For a defined number of generations, each individual in the population was assigned to a random combination and the normalized success for each combination recorded. Subsequently, the number of combinations was reduced to a predefined level at which the success counters were reset. The same was undertaken for the search operators, whereby the algorithm emphasized the well-performing operators during the search process. Also, a new methodology for handling constraints, in which the constraints were divided into subsets based on each constraint violation, was intro-

Table 9 A comparison summary of DE-AOPS against εDEag, ECHT-DE, ECHT-ARMOR-DE, BRGA and APM-ES. The values in rows 1 and 2 refer to the number of test problems in which DE-AOPS is better, similar and worse than the second algorithm in the 1st column, based on the best and average fitness values obtained, respectively. p is a probability that is used to make a decision based on Wilcoxon's test

Algorithms	Criteria	10D					30D				
		Better	Similar	Worse	p	Decision	Better	Similar	Worse	p	Decision
DE-AOPS vs. εDEag	Best fitness values	5	12	1	0.345	$\approx$	17	1	0	0.000	+
	Average fitness values	9	7	2	0.286	$\approx$	17	0	1	0.003	+
DE-AOPS vs. ECHT-DE	Best fitness values	2	16	0	0.18	$\approx$	7	9	2	0.021	+
	Average fitness values	12	3	3	0.009	+	14	1	3	0.002	+
DE-AOPS vs. ECHT-ARMOR-DE	Best fitness values	1	17	0	0.317	$\approx$	11	6	1	0.004	+
	Average fitness values	8	7	3	0.033	+	14	1	3	0.003	+
DE-AOPS vs. MS-(μ+λ)-CDE	Best fitness values	3	15	0	0.109	$\approx$	16	2	0	0.000	+
	Average fitness values	13	2	3	0.002	+	18	0	0	0.000	+
DE-AOPS vs. BRGA	Best fitness values	17	1	0	0.000	+	18	0	0	0.000	+
	Average fitness values	18	0	0	0.000	+	18	0	0	0.000	+
DE-AOPS vs. LSHADE-SFS	Best fitness values	5	13	0	0.00	+	11	5	2	0.005	+
	Average fitness values	8	7	3	0.021	+	13	2	3	0.002	+
DE-AOPS vs. LSHADE-AP	Best fitness values	5	13	0	0.00	+	11	5	2	0.003	+
	Average fitness values	8	7	3	0.021	+	13	2	3	0.004	+

duced. The algorithm started with a single subset and gradually combined all the subsets. The proposed algorithm was tested on the CEC2010 problems and showed superior performance to our earlier proposed algorithm in terms of solution quality. Also, it was able to reduce the average FEs by 13.29% and 23.06% for the 10D and 30D problems, respectively, and performed much better than state-of-the-art algorithms

In addition, an analysis of the effect of the proposed CHT on the performance of DE-AOPS successfully demonstrated its benefits as did the proposed methodology for its dynamic switching of population sizes with fixed PS values.

6 Compliance with Ethical Standards

- Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.
- Conflict of Interest: The authors declare that they have no conflict of interest.
- Informed consent: Informed consent was obtained from all individual participants included in the study.

References

1. Abbass HA (2002) The self-adaptive pareto differential evolution algorithm. In: IEEE Congress on Evolutionary Computation, IEEE, vol 1, pp 831–836
2. Abdul-Rahman OA, Munetomo M, Akama K (2013) An adaptive parameter binary-real coded genetic algorithm for constraint optimization problems: Performance analysis and estimation of optimal control parameters. Information Sciences 233(0):54 – 86, DOI http://dx.doi.org/10.1016/j.ins.2013.01.005
3. Asafuddoula M, Ray T, Sarker R (2011) An adaptive differential evolution algorithm and its performance on real world optimization problems. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1057–1062
4. Asafuddoula M, Ray T, Sarker R (2015) A differential evolution algorithm with constraint sequencing: An efficient approach for problems with inequality constraints. Applied Soft Computing 36:101–113
5. Bertsekas D (1999) Nonlinear Programming. Athena Scientific
6. Birattari M, Yuan Z, Balaprakash P, Stützle T (2010) F-race and iterated f-race: An overview. In: Experimental methods for the analysis of optimization algorithms, Springer, pp 311–336
7. Brest J, Maučec MS (2008) Population size reduction for the differential evolution algorithm. Applied Intelligence 29(3):228–247
8. Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10(6):646–657
9. Brest J, Boskovic B, Zamuda A, Fister I, Mezura-Montes E (2013) Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies. In: IEEE Congress on Evolutionary Computation, IEEE, pp 377–383
10. Caraffini F, Neri F, Cheng J, Zhang G, Picinali L, Iacca G, Mininno E (2013) Super-fit multicriteria adaptive differential evolution. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1678–1685
11. Choi TJ, Ahn CW (2015) An adaptive cauchy differential evolution algorithm with population size reduction and modified multiple mutation strate-

gies. In: Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems-Volume 2, Springer, pp 13–26
12. Corder GW, Foreman DI (2009) Nonparametric statistics for non-statisticians: a step-by-step approach. John Wiley & Sons
13. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. Evolutionary Computation, IEEE Transactions on 15(1):4–31
14. Davis L, et al (1991) Handbook of genetic algorithms, vol 115. Van Nostrand Reinhold New York
15. Deb K (2000) An efficient constraint handling method for genetic algorithms. Computer methods in applied mechanics and engineering 186(2):311–338
16. Elsayed SM, Sarker RA, Essam DL (2011) Multi-operator based evolutionary algorithms for solving constrained optimization problems. Computers & operations research 38(12):1877–1896
17. Elsayed SM, Sarker RA, Essam DL (2012) On an evolutionary approach for constrained optimization problem solving. Applied Soft Computing 12(10):3208–3227
18. Elsayed SM, Sarker RA, Essam DL (2013) An improved self-adaptive differential evolution algorithm for optimization problems. IEEE Transactions on Industrial Informatics 9(1):89–99
19. Elsayed SM, Sarker RA, Essam DL (2013) Self-adaptive differential evolution incorporating a heuristic mixing of operators. Computational Optimization and Applications 54(3):771–790
20. Fitzgerald T, O'Sullivan B, Malitsky Y, Tierney K (2014) Online search algorithm configuration. In: AAAI, pp 3104–3105
21. Gao WF, Yen GG, Liu SY (2015) A dual-population differential evolution with coevolution for constrained optimization. IEEE Transactions on Cybernetics 45(5):1108–1121
22. Gong W, Cai Z, Liang D (2015) Adaptive ranking mutation operator based differential evolution for constrained optimization. IEEE Transactions on Cybernetics 45(4):716–727, DOI 10.1109/TCYB.2014.2334692
23. Grefenstette J (1986) Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 16(1):122–128, DOI 10.1109/TSMC.1986.289288
24. Hansen N, Müller S, Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evolutionary Computation 11(1):1–18
25. Huang VL, Qin AK, Suganthan PN (2006) Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In: IEEE congress on evolutionary computation, pp 17–24
26. Jia G, Wang Y, Cai Z, Jin Y (2013) An improved (μ+ λ)-constrained differential evolution for constrained optimization. Information Sciences 222:302–322
27. Kennedy J (2010) Particle swarm optimization. In: Encyclopedia of Machine Learning, Springer, pp 760–766
28. Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Computing 9(6):448–462
29. Mallipeddi R, Suganthan P (2009) Differential evolution algorithm with ensemble of populations for global numerical optimization. Opsearch 46(2):184–213
30. Mallipeddi R, Suganthan PN (2010) Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies. In: Swarm, Evolutionary, and Memetic Computing, Springer, pp 71–78
31. Mallipeddi R, Suganthan PN (2010) Ensemble of constraint handling techniques. IEEE Transactions on Evolutionary Computation 14(4):561–579
32. Mallipeddi R, Suganthan PN (2010) Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. Technical Report, Nanyang Technological University, Singapore
33. Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing 11(2):1679–1696
34. Mezura Montes E, Coello Coello CA (2003) Adding a diversity mechanism to a simple evolution strategy to solve constrained optimization problems. In: IEEE Congress on Evolutionary Computation, IEEE, vol 1, pp 6–13
35. Michalewicz Z, Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. Evolutionary computation 4(1):1–32
36. Mohamed AW, Sabry HZ (2012) Constrained optimization based on modified differential evolution algorithm. Information Sciences 194:171–208
37. Montgomery DC, Peck EA, Vining GG (2012) Introduction to linear regression analysis, vol 821. John Wiley & Sons
38. Poláková R, Tvrdík J (2011) Various mutation strategies in enhanced competitive differential evolution for constrained optimization. In: IEEE Symposium on Differential Evolution, IEEE, pp 1–8

39. Price K, Storn RM, Lampinen JA (2006) Differential evolution: a practical approach to global optimization. Springer Science & Business Media
40. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13(2):398–417
41. Rönkkönen J, et al (2009) Continuous Multimodal Global Optimization with Differential Evolution-Based Methods. Lappeenranta University of Technology
42. Sarker R, Elsayed S, Ray T (2014) Differential evolution with dynamic parameters selection for optimization problems. Evolutionary Computation, IEEE Transactions on 18(5):689–707, DOI 10.1109/TEVC.2013.2281528
43. Schoenauer M, Xanthakis S (1993) Constrained ga optimization. In: ICGA, pp 573–580
44. Schwefel HP (1984) Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of organic evolution. Annals of Operations Research 1(2):165–167
45. Si C, An J, Lan T, Ußmüller T, Wang L, Wu Q (2014) On the equality constraints tolerance of constrained optimization problems. Theoretical Computer Science 551:55–65
46. Storn R (2008) Differential evolution research - trends and open questions. In: Chakraborty U (ed) Advances in Differential Evolution, Studies in Computational Intelligence, vol 143, Springer Berlin Heidelberg, pp 1–31, DOI 10.1007/978-3-540-68830-3_1
47. Storn R, Price K (1995) Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, vol 3. ICSI Berkeley
48. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11(4):341–359
49. Takahama T, Sakai S (2010) Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1–9
50. Tanabe R, Fukunaga A (2013) Evaluating the performance of shade on cec 2013 benchmark problems. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1952–1959
51. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: IEEE Congress on Evolutionary Computation, IEEE, pp 71–78
52. Tanabe R, Fukunaga A (2014) Improving the search performance of shade using linear population size reduction. In: IEEE Congress on Evolutionary Computation, pp 1658–1665, DOI 10.1109/CEC.2014.6900380
53. Tasgetiren M, Suganthan P (2006) A multi-populated differential evolution algorithm for solving constrained optimization problem. In: IEEE Congress on Evolutionary Computation, pp 33–40, DOI 10.1109/CEC.2006.1688287
54. Teo J (2006) Exploring dynamic self-adaptive populations in differential evolution. Soft Computing 10(8):673–686
55. Tsai JT (2015) Improved differential evolution algorithm for nonlinear programming and engineering design problems. Neurocomputing 148:628 – 640
56. Tvrdík J (2009) Adaptation in differential evolution: A numerical comparison. Applied Soft Computing 9(3):1149–1155
57. Tvrdík J, Polakova R (2010) Competitive differential evolution for constrained problems. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1–8
58. Tvrdík J, Polakova R (2013) Competitive differential evolution applied to cec 2013 problems. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1651–1657
59. Vrugt JA, Robinson BA (2007) Improved evolutionary optimization from genetically adaptive multimethod search. Proceedings of the National Academy of Sciences 104(3):708–711
60. Vrugt JA, Robinson BA, Hyman JM (2009) Self-adaptive multimethod search for global optimization in real-parameter spaces. IEEE Transactions on Evolutionary Computation 13(2):243–259
61. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Transactions on Evolutionary Computation 15(1):55–66
62. Wang Y, Wang BC, Li HX, Yen GG (2016) Incorporating objective function information into the feasibility rule for constrained evolutionary optimization. IEEE Transactions on Cybernetics 46(12):2938–2952
63. Wei W, Wang J, Tao M (2015) Constrained differential evolution with multiobjective sorting mutation operators for constrained optimization. Applied Soft Computing 33:207–222
64. Wei W, Zhou J, Chen F, Yuan H (2016) Constrained differential evolution using generalized opposition-based learning. Soft Computing pp 1–25

65. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1):67–82
66. Yi W, Li X, Gao L, Zhou Y, Huang J (2016) ε constrained differential evolution with pre-estimated comparison using gradient-based approximation for constrained optimization problems. Expert Systems with Applications 44:37–49
67. Yu WJ, Shen M, Chen WN, Zhan ZH, Gong YJ, Lin Y, Liu O, Zhang J (2014) Differential evolution with two-level parameter adaptation. IEEE Transactions on Cybernetics 44(7):1080–1099
68. Zaharie D (2007) A comparative analysis of crossover variants in differential evolution. Proceedings of IMCSIT pp 171–181
69. Zamuda A, Brest J (2012) Population reduction differential evolution with multiple mutation strategies in real world industry challenges. In: Rutkowski L, Korytkowski M, Scherer R, Tadeusiewicz R, Zadeh L, Zurada J (eds) Swarm and Evolutionary Computation, Springer, pp 154–161
70. Zamuda A, Brest J, Mezura-Montes E (2013) Structured population size reduction differential evolution with multiple mutation strategies on cec 2013 real parameter optimization. In: IEEE Congress on Evolutionary Computation, IEEE, pp 1925–1931
71. Zhang J, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13(5):945–958

Appendix A: Best and average fitness values (FV) achieved by DE-AOPS and 8 state-of-the-art algorithms, where "*" and "-" refer infeasible solutions, while "n/a" refers to not available results

Prob.	Algorithm	10D			30D		
		Best FV	Average FV	*Std.*	Best FV	Average FV	*Std.*
C01	**DE-AOPS**	**-7.473104E-01**	**-7.473104E-01**	2.821438E-16	**-8.218844E-01**	**-8.216488E-01**	**8.639430E-04**
	DE-DPS	**-7.473104E-01**	**-7.473104E-01**	**2.26623E-16**	**-8.21884E-01**	-8.212036E-01	1.79648E-03
	εDEag	**-7.473104E-01**	**-7.470402E-01**	1.323339E-03	-8.218255E-01	-8.208687E-01	7.103893E-04
	ECHT-DE	**-0.7473**	-0.7470	0.0014	-0.8217	-0.7994	0.0179
	ECHT-ARMOR-DE	**-7.4730E-01**	-7.4700E-0	1.4E-03	-8.1806E-01	-7.8992E-01	2.51E-02
	MS-(μ+λ)-CDE	**-7.4731E-01**	-7.4193E-01	1.1715E-02	-8.1142E-01	-7.3471E-01	5.3191E-02
	BRGA	**-7.473E-01**	-7.288E-01	1.732E-02	-7.850E-01	-6.940E-01	6.740E-02
	LSHADE-SFS	**-7.473104E-01**	**-7.473104E-01**	6.969063E-15	**-8.218844E-01**	-8.215233E-01	1.261038E-03
	LSHADE-AP	**-7.473104E-01**	-7.467701E-01	1.869859E-03	**-8.218844E-01**	-8.215695E-01	1.090028E-03
C02	**DE-AOPS**	**-2.277711E+00**	-1.867885E+00	4.897646E-01	**-2.28097E+00**	-1.940295E+00	2.702060E-01
	DE-DPS	**-2.277711E+00**	**-2.277512E+00**	**2.54035E-04**	-2.280671	**-2.244631**	5.20548E-02
	εDEag	-2.277702E+00	-2.269502E+00	2.3897790E-02	-2.169248E+00	-2.151424E+00	1.197582E-02
	ECHT-DE	**-2.2777**	**-2.2744**	0.0067	-2.2251	-1.9943	0.2099
	ECHT-ARMOR-DE	**-2.2777E+00**	-2.2770E+00	3.3E-03	-2.2607E+00	-2.1706E+00	7.36E-02
	MS-(μ+λ)-CDE	**-2.2777E+00**	-2.2527E+00	1.2755E-02	-2.1431E+00	-1.2735E+00	4.2140E-01
	BRGA	-4.028E-01	3.346E+00	1.257E+00	2.660E+00	4.230E+00	5.170E-01
	LSHADE-SFS	-2.277710E+00	-2.148228E+00	1.853388E-01	**-2.28097E+00**	**-2.277377E+00**	3.940237E-03
	LSHADE-AP	-2.273042E+00	-2.179539E+00	1.022030E-01	-2.280958E+00	-2.226907E+00	8.139753E-02
C03	**DE-AOPS**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**7.705795E-26**	**2.679942E-25**
	DE-DPS	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	1.6200E-19	1.8479E-13	4.17994E-13
	εDEag	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	2.867347E+01	2.883785E+01	8.047159E-01
	ECHT-DE	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	3.2433E-21	9.8920E+01	6.2594E+01
	ECHT-ARMOR-DE	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	2.5801E-24	2.6380E+01	7.94E+00
	MS-(μ+λ)-CDE	**0.00000E+00**	5.9090E+13	1.2249E+14	1.1164E+12	2.2083E+13	2.5059E+13
	BRGA	3.298E+09	2.187E+13	3.860E+13	4.560E+11*	2.450E+13*	2.800E+13*
	LSHADE-SFS	7.274523E+00	3.477702E+14*	2.487137E+14	1.039162E+01*	1.207354E+01*	6.990000E-01
	LSHADE-AP	5.863095E+04*	1.878530E+13*	5.530525E+13	1.059550E+02*	1.210112E+02*	1.086404E+01
C04	**DE-AOPS**	**-1.000000E-05**	**-1.000000E-05**	**0.00000E+00**	**-3.333333E-06**	**-3.333333E-06**	**7.207721E-14**
	DE-DPS	**-1.000000E-05**	**-1.000000E-05**	9.09633E-15	-3.3318E-06	-3.3123E-06	2.03926E-08
	εDEag	-9.992345E-06	-9.918452E-06	1.5467300E-07	4.698111E-03	8.162973E-03	3.067785E-03
	ECHT-DE	**-1.00000E-05**	**-1.00000E-05**	**0.00000E+00**	-3.3015E-06	-1.0257E-06	9.0135E-02
	ECHT-ARMOR-DE	**-1.00000E-05**	**-1.00000E-05**	**0.00000E+00**	-3.3326E-06	8.3713E-02	2.89E-01
	MS-(μ+λ)-CDE	**-1.0000E-05**	**-1.0000E-05**	6.7684E-13	1.7476E-02	6.8459E+00	9.9375E+00
	BRGA	5.627E-03*	3.987E+00*	7.365E+00	2.830E-01*	4.340E+00*	5.260E+00
	LSHADE-SFS	1.369123E+01*	3.462476E+01*	9.246601E+00	4.006270E+01*	4.896515E+01*	4.733187E+00
	LSHADE-AP	1.287461E+02*	5.081855E+03*	3.808307E+03	7.518004E+05*	8.316182E+06*	8.363571E+06
C05	**DE-AOPS**	**-4.836106E+02**	**-4.836106E+02**	3.480934E-13	**-4.836106E+02**	**-4.836106E+02**	**7.071077E-12**
	DE-DPS	**-4.836106E+02**	**-4.836106E+02**	1.25826E-10	**-4.836106E+02**	**-4.836106E+02**	4.42074E-06
	εDEag	**-4.836106E+02**	**-4.836106E+02**	3.89035E-13	-4.531307E+02	-4.495460E+02	2.899105E+00
	ECHT-DE	**-483.6106**	-411.4532	76.3137	-213.6844	-106.4228	167.1481
	ECHT-ARMOR-DE	**-4.8361E+02**	**-4.8361E+02**	**0.0E+00**	-4.8122E+02	-4.3335E+02	1.46E+02
	MS-(μ+λ)-CDE	**-4.8361E+02**	-4.6883E+02	4.8666E+01	-4.6313E+02	-3.5787E+02	8.3832E+01
	BRGA	-3.451E+01	1.166E+02	6.813E+01	1.240E+02	2.640E+02	7.070E+01
	LSHADE-SFS	**-4.836106E+02**	**-4.836106E+02**	3.480934E-13	**-4.836106E+02**	**-4.836106E+02**	1.778427E-08
	LSHADE-AP	**-4.836106E+02**	**-4.836106E+02**	3.480934E-13	**-4.836106E+02**	**-4.836106E+02**	2.992156E-10
C06	**DE-AOPS**	**-5.786624E+02**	**-5.786624E+02**	**1.411580E-13**	**-5.306379E+02**	**-5.306379E+02**	**3.232427E-11**
	DE-DPS	**-5.786624E+02**	**-5.78662E+02**	8.05379E-04	**-5.306379E+02**	-5.306329E+02	5.89364E-03
	εDEag	-5.786581E+02	-5.786528E+02	3.6271690E-03	-5.285750E+02	-5.279068E+02	4.748378E-01
	ECHT-DE	**-578.6624**	-562.4688	45.1479	-295.7192	-137.6152	98.8995
	ECHT-ARMOR-DE	**-5.7866E+02**	**-5.7866E+02**	4.0E-13	-5.2465E+02	-4.8931E+02	1.32E+02
	MS-(μ+λ)-CDE	**-5.7866E+02**	-5.0947E+02	1.0497E+02	-5.2182E+02	-3.2308E+02	1.2814E+02
	BRGA	9.574E+01	1.874E+02	5.703E+01	1.610E+02	2.680E+02	7.490E+01
	LSHADE-SFS	**-5.786624E+02**	-3.113090E+02	2.770278E+02	**-5.306379E+02**	-5.306196E+02	9.017545E-02
	LSHADE-AP	**-5.786624E+02**	-4.787993E+02	1.544245E+02	**-5.306379E+02**	-5.306163E+02	6.487517E-02
C07	**DE-AOPS**	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**2.204073E-27**	**4.881227E-27**
	DE-DPS	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	5.48786E-20	1.03359E-13	2.20540E-13
	εDEag	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	1.147112E-15	2.603632E-15	1.233430E-15
	ECHT-DE	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	0.1329	0.7279
	ECHT-ARMOR-DE	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	1.0789E-25	2.20E-25
	MS-(μ+λ)-CDE	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	1.5946E-01	7.9732E-01
	BRGA	2.242E-03	1.260E+01	2.752E+01	1.440E+01	7.450E+02	1.570E+03
	LSHADE-SFS	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	3.542613E-26	6.644020E-22	1.945602E-21
	LSHADE-AP	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	3.292713E-21	9.561772E-21
C08	**DE-AOPS**	**0.000000E+00**	6.590903E+00	5.122612E+00	**0.000000E+00**	**6.312850E-28**	**2.715168E-27**
	DE-DPS	**0.00000E+00**	**3.950387E+00**	**5.01904E+00**	4.575719E-13	3.447568E-09	8.86366E-09
	εDEag	**0.00000E+00**	6.727528E+00	5.560648E+00	2.518693E-14	7.831464E-14	4.855177E-14
	ECHT-DE	**0.00000E+00**	6.1566E+00	6.4527E+00	**0.00000E+00**	3.3585E+01	1.1072E+02
	ECHT-ARMOR-DE	**0.00000E+00**	7.5262E+00	5.0E+00	**0.000000E+00**	2.0101E+01	4.70E+01
	MS-(μ+λ)-CDE	**0.0000E+00**	8.0132E+00	4.6560E+00	2.4806E-15	1.0464E+03	3.1626E+03
	BRGA	8.795E-04	2.716E+02	4.555E+02	1.820E+01	1.400E+03	1.870E+03
	LSHADE-SFS	**0.000000E+00**	1.006622E+01	3.029575E+00	2.267461E-23	2.607775E+01	5.651018E+01
	LSHADE-AP	**0.000000E+00**	9.773469E+00	3.284353E+00	1.198148E-25	1.878701E+01	4.545001E+01
C09	**DE-AOPS**	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**7.257287E-27**	**1.440838E-26**
	DE-DPS	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	3.81580E-23	5.29059E-14	1.27939E-13
	εDEag	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	2.770665E-16	1.072140E+01	2.821923E+01
	ECHT-DE	**0.00000E+00**	1.4691E-01	8.0482E-01	**0.00000E+00**	4.2441E+01	1.3762E+02
	ECHT-ARMOR-DE	**0.000000E+00**	1.7633E-01	8.8E-01	**0.000000E+00**	4.6110E+00	2.31E+01
	MS-(μ+λ)-CDE	**0.00000E+00**	2.9773E+04	1.4886E+05	4.3409E+07	2.5118E+09	3.7457E+09
	BRGA	9.640E-10	7.888E+02	3.189E+03	3.600E+01	2.150E+05	4.680E+05
	LSHADE-SFS	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	9.519716E+01	4.894634E+04	7.521028E+04
	LSHADE-AP	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	1.825224E-08	4.864067E+04	5.852319E+04

APPENDIX A (CONT.)

Prob.	Algorithm	10D			30D		
		Best	Mean	*Std.*	Best	Mean	*Std.*
C10	**DE-AOPS**	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**8.491790E-27**	1.654768E-26
	DE-DPS	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	1.63778E-21	2.23928E-13	6.24058E-13
	εDEag	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	3.252002E+01	3.326175E+01	4.545577E-01
	ECHT-DE	**0.000000E+00**	1.7117E+00	7.6554E+00	**0.00000E+00**	5.3381E+01	8.8308E+01
	ECHT-ARMOR-DE	**0.000000E+00**	**0.000000E+00**	**0.00000E+00**	6.0209E-13	6.5536E+01	1.07E+02
	MS-(μ+λ)-CDE	**0.000000E+00**	4.5513E+01	2.8860E+01	4.5657E+07	1.4104E+09	1.9076E+09
	BRGA	4.306E+01	5.597E+02	7.129E+02	8.790E+02	1.600E+04	1.890E+04
	LSHADE-SFS	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	5.329364E-14	1.827317E+04	5.453344E+04
	LSHADE-AP	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	3.774930E-13	8.544700E+01	1.677114E+02
C11	**DE-AOPS**	**-1.522713E-03**	**-1.522713E-03**	**7.796085E-18**	**-3.923439E-04**	**-3.923438E-04**	**1.199812E-10**
	DE-DPS	**-1.52271E-03**	**-1.52271E-03**	**3.14275E-12**	**-3.92344E-04**	-3.923423E-04	8.77688E-10
	εDEag	**-1.52271E-03**	**-1.52271E-03**	**6.3410350E-11**	-3.268462E-04	-2.863882E-04	2.707605E-05
	ECHT-DE	**-0.0015**	-0.0044*	0.0157	-0.0004	0.0026	0.0060
	ECHT-ARMOR-DE	**-1.5227E-03**	-	4.4E-02	**-3.9234E-04**	-	5.28E-03
	MS-(μ+λ)-CDE	-8.7342E-02*	-4.9555E-03*	1.7164E-02	-3.9231E-04	2.6918E-03*	7.1179E-03
	BRGA	-9.097E-05*	-1.717E-01*	1.896E+00	-4.380E-02*	-9.570E-02*	2.440E-01
	LSHADE-SFS	-2.702077E+01*	-6.232310E+00*	1.322113E+01	-8.756476E+00*	1.268593E+00*	6.369888E+00
	LSHADE-AP	2.507472E+05*	1.545297E+07*	1.343245E+07	5.490169E+10*	2.552285E+11*	2.222209E+11
C12	**DE-AOPS**	-3.054888E+02	-6.730572E+01	1.138406E+02	-1.992635E-01	-1.992635E-01	1.012003E-08
	DE-DPS	-1.9925E-01	-1.9925E-01	4.19599E-10	**-1.992635E-01**	-1.9926E-01	2.60287E-08
	εDEag	**-5.700899E+02**	**-3.367349E+02**	**1.7821660E+02**	-1.991453E-01	3.562330E+02	2.889253E+02
	ECHT-DE	-0.1992	-171.8714*	221.0436	-0.1993	-25.1292*	136.5593
	ECHT-ARMOR-DE	-1.9925E-01	-1.9925E-01	1.6E-13	**-1.9926E-01**	-1.6076E-01	1.93E-01
	MS-(μ+λ)-CDE	-1.9925E-01	-1.9924E-01	2.5000E-06	**-1.9926E-01**	5.7980E-01	3.0953E+00
	BRGA	-1.958E-01	1.666E+01*	3.712E+01	-2.820E+02*	-7.140E+00*	6.320E+01
	LSHADE-SFS	-2.194386E+03*	1.968868E+02*	1.249431E+03	-3.601012E+03*	4.279879E+02*	2.111636E+03
	LSHADE-AP	5.502293E+06*	8.044863E+07*	1.127357E+08	4.863293E+12*	3.164231E+13*	2.085429E+13
C13	**DE-AOPS**	**-6.842937E+01**	**-6.842937E+01**	2.626767E-14	**-6.84294E+01**	-6.457963E+01	3.027647E+00
	DE-DPS	**-6.842937E+01**	**-6.842937E+01**	**0.0000000E+00**	**-6.84294E+01**	**-66.33141**	2.08493E+00
	εDEag	**-6.842937E+01**	-6.842936E+01	1.0259600E-06	-6.642473E+01	-6.535310E+01	5.733005E-01
	ECHT-DE	**-68.4294**	-65.1208	2.3750	**-68.4294**	-64.5831	1.6690
	ECHT-ARMOR-DE	**-6.8429E+01**	-6.7169E+01	2.1E+00	-6.7416E+01	-6.4646E+01	1.97E+00
	MS-(μ+λ)-CDE	**6.8429E+01**	-6.3669E+01	2.3430E+00	-6.3251E+01	-5.9597E+01	2.5966E+00
	BRGA	**-6.843E+01**	6.154E+01	2.523E+00	-6.020E+01	-5.630E+01	2.320E+00
	LSHADE-SFS	-6.842937E+01	-6.769539E+01	3.604648E-01	-6.227638E+01	-3.089694E+01*	2.650899E+01
	LSHADE-AP	-6.842937E+01	-6.765320E+01	5.239996E-01	-6.227637E+01	1.028506E+03*	2.803956E+03
C14	**DE-AOPS**	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**9.521141E-27**	**2.881915E-26**
	DE-DPS	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	8.925796E-21	5.137406E-14	1.31791E-13
	εDEag	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	5.015863E-14	3.089407E-13	5.608409E-13
	ECHT-DE	**0.00000E+00**	7.0242E+05	3.1937E+06	**0.00000E+00**	1.2368E+05	6.7736E+05
	ECHT-ARMOR-DE	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	1.5809E-27	6.6135E+02	2.47E+03
	MS-(μ+λ)-CDE	**0.00000E+00**	9.5337E+01	4.0097E+02	4.8313E-16	6.7881E+05	3.3878E+06
	BRGA	5.004E+00	2.537E+06	6.030E+06	2.480E+01	3.130E+10	1.530E+11
	LSHADE-SFS	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	2.423696E-26	4.834862E-17	2.395498E-16
	LSHADE-AP	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	3.013299E-25	3.225427E-16	1.591360E-15
C15	**DE-AOPS**	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**9.452131E-28**	**2.809169E-27**
	DE-DPS	**0.000000E+00**	5.438912E-26	2.19329E-25	6.336258E-20	1.957805E-13	5.22687E-13
	εDEag	**0.000000E+00**	1.798980E-01	8.8131560E-01	2.160345E+01	2.160376E+01	1.104834E-04
	ECHT-DE	**0.00000E+00**	2.3392E+13	5.2988E+13	1.9922E+09	1.9409E+11	4.3524E+11
	ECHT-ARMOR-DE	**0.0000E+00**	2.8246E+00	1.6E+00	1.1716E-04	3.1316E+08	1.20E+09
	MS-(μ+λ)-CDE	1.4458E+10	4.8297E+13	4.2378E+13	4.7916E+01	1.1977E+13	2.1951E+13
	BRGA	1.655E+01	8.201E+06	1.570E+07	5.610E+01	5.780E+06	1.110E+07
	LSHADE-SFS	**0.000000E+00**	1.798978E-01	8.994890E-01	1.205931E+01	2.131498E+01	1.984036E+00
	LSHADE-AP	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	2.150558E+00	2.101213E+01	3.982252E+00
C16	**DE-AOPS**	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**
	DE-DPS	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**
	εDEag	**0.000000E+00**	3.702054E-01	3.7104790E-01	**0.000000E+00**	2.168404E-21	1.062297E-20
	ECHT-DE	**0.000000E+00**	3.9327E-02	4.2815E-02	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**
	ECHT-ARMOR-DE	**0.000000E+00**	2.8478E-02	5.0E-02	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
	MS-(μ+λ)-CDE	**0.00000E+00**	8.2716E-02	1.3780E-01	**0.00000E+00**	6.9181E-03	2.8588E-02
	BRGA	6.663E-01	9.894E-01	8.576E-02	9.780E-01	1.050E+00	2.510E-02
	LSHADE-SFS	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
	LSHADE-AP	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
C17	**DE-AOPS**	**0.000000E+00**	6.289787E-26	1.790341E-25	2.580746E-32	**1.236845E-19**	**5.494802E-19**
	DE-DPS	**0.000000E+00**	1.061273E-24	3.88726E-24	3.236013E-12	8.766191E-02	1.55966E-01
	εDEag	1.463180E-17	1.249561E-01	1.9371970E-01	2.165719E-01	6.326487E+00	4.986691E+00
	ECHT-DE	**0.00000E+00**	1.1152E-01	3.3152E-01	**0.00000E+00**	2.7496E-01	3.7832E-01
	ECHT-ARMOR-DE	**0.00000E+00**	**3.6978E-33**	**3.1E-33**	3.3564E-16	4.0336E-01	3.51E-01
	MS-(μ+λ)-CDE	**0.00000E+00**	4.9304E-34	1.7065E-33	2.5480E-02	2.6504E-01	2.3245E-01
	BRGA	2.864E+00	1.704E+01	9.554E+00	4.760E+01	1.160E+02	4.770E+01
	LSHADE-SFS	**0.000000E+00**	7.395571E-34	2.044028E-33	**0.000000E+00**	4.602433E-18	2.301215E-17
	LSHADE-AP	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	7.284498E-23	3.442614E-22
C18	**DE-AOPS**	**0.000000E+00**	1.502663E-24	7.260554E-24	6.441314E-26	5.294587E-20	1.156098E-19
	DE-DPS	**0.000000E+00**	3.209082E-23	7.12457E-23	6.747743E-13	9.459914E-08	1.91224E-07
	εDEag	3.731440E-20	9.678765E-19	1.8112340E-18	1.226054E+00	8.754569E+01	1.664753E+02
	ECHT-DE	**0.000000E+00**	**0.00000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
	ECHT-ARMOR-DE	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**
	MS-(μ+λ)-CDE	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	2.4371E-20	7.4534E-01	1.8865E+00
	BRGA	1.280E-01	2.589E+00	3.917E+00	1.200E+01	4.130E+01	2.310E+01
	LSHADE-SFS	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	2.558945E-31	2.523441E-31
	LSHADE-AP	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**	1.157669E-30	2.122595E-30

Appendix B: The best combination of F (green squares) and Cr (blue circles) at every combinations reduction level for the 10D and 30D test problems, respectively. Each integer value of each parameter represents its corresponding range as described in Section 4. Each subplot is for the best solution obtained.