

Differential evolution framework for big data optimization

Saber Elsayed^{1,2} · Ruhul Sarker¹

Received: 31 July 2015 / Accepted: 7 December 2015 / Published online: 7 January 2016
© Springer-Verlag Berlin Heidelberg 2016

Abstract During the last two decades, dealing with big data problems has become a major issue for many industries. Although, in recent years, differential evolution has been successful in solving many complex optimization problems, there has been research gaps on using it to solve big data problems. As a real-time big data problem may not be known in advance, determining the appropriate differential evolution operators and parameters to use is a combinatorial optimization problem. Therefore, in this paper, a general differential evolution framework is proposed, in which the most suitable differential evolution algorithm for a problem on hand is adaptively configured. A local search is also employed to increase the exploitation capability of the proposed algorithm. The algorithm is tested on the 2015 big data optimization competition problems (six single objective problems and six multi-objective problems). The results show the superiority of the proposed algorithm to several state-of-the-art algorithms.

Keywords Differential evolution · Big data optimization · EEG signals

1 Introduction

Optimizing big data problems has become a significant research topic for many organizations; for instance, in August

2010, the White House announced that along with healthcare and national security big data is a national challenge and priority. The underlying complexity of big data problems is having to deal with a massive number of decision variables, different mathematical properties of the objective function(s) and/or constraints, and the computational time required for the optimization process.

The concept of big data set/matrix or big database is different from the emerging research topic “big data”. By “big data” we mean data that is too big, too fast, or too hard for existing tools to process [14]. It is also defined using 4Vs—volume, velocity, variety and veracity [38], where volume measures the amount of data that can be captured, communicated, aggregated, stored and analyzed, velocity the speed of data creation, streaming and aggregation, variety the richness of the data representation—text, images video, audio, etc. while veracity suggests that, despite data being available, its quality is a major concern [38]. Big data is assumed to be a ‘big blessing’ but comes with huge challenges. Running optimization algorithms on voluminous data sets using available processors and storage units is difficult and, with the advent of streaming data sources, the optimization process must often be performed in real time, typically without a chance to revisit past entries [25]. The problem areas that can benefit from optimizing big data are found in many industries including, but not limited to healthcare, the public sector (i.e., enabling sophisticated tax agencies to apply automated algorithms to systematically check tax returns and automatically flag those that require further examination or auditing which can reduce the gap in tax revenue by up to 10 % points [16]), cyber security (i.e., providing network managers with the means to process millions of daily attacks and identify the more serious ones), logistics, and the air traffic, retail, manufacturing, and defence sectors [23]. The problem considered in this paper is abstracted from dealing with

✉ Saber Elsayed
s.elsayed@adfa.edu.au

Ruhul Sarker
r.sarker@adfa.edu.au

¹ School of Engineering and Information technology,
University of New South Wales, Canberra, Australia

² Faculty of Computers and Informatics, Zagazig University,
Zagazig, Egypt

Electroencephalographic (EEG) signals through independent component analysis (ICA) which to some extent approximate a big optimization problem, as it requires a real-time handling of EEG signals [10, 11, 37].

Evolutionary algorithms (EAs) are population-based stochastic search methods that have demonstrated great success in solving complex optimization problems, of which DE has shown its powerful search capabilities [1, 6–8, 19]. Although DE does not need to satisfy some mathematical properties, such as convexity, continuity and differentiability, of the objective function and constraints of a problem which are required by the deterministic methods [24], as it can flexibly represent knowledge, has innate parallelism behavior and performs better than many other EAs, it is not the best algorithm for all types of problems [6]. It has some basic operators (representation, mutation and crossover) and parameters (population size, mutation rate and crossover rate) but there is a research gap regarding adopting it to solve big data problems. In addition, no single operator and/or control parameter is the best for all types of test problems.

Recently, the adaptation of DE control parameters and search operators has attracted a great deal of interest among the evolutionary computation (EC) community. Based on how a DE algorithm's control parameters are adapted, it can be classified into one of the following three classes [5]: (1) *deterministic approaches*—the parameters are changed based on some deterministic rules regardless of any feedback from the algorithm; (2) *adaptive approaches*—the parameters are dynamically updated based on learning from the evolutionary process; and (3) *self-adaptive approaches*—the parameters are directly encoded within individuals and undergo recombination. Also, recent studies have been proposed, which used ensemble of DE operators [1, 6–8, 19]. However, such methodologies for optimizing big data problems do not exist [11].

In this paper, an automatic integration of DE variants framework is developed and analyzed considering big data problems, both single and multi-objective. In it, three populations are initially run in parallel for a predefined number of generations (cycle), and during this cycle information regarding the performance of every DE is recorded. After that, an exponential curve is fitted and used to predict the future performance that can be achieved by each DE operator. The one with the best predicted future value is selected to optimize the best set of individuals among all populations. Also, information sharing and local search procedures are adopted. The algorithm is tested on the 2015 big data optimization problems. The objective is to decompose the obtained EEG signals into two parts, where the first part needs to be similar to the original signals to keep useful information, while the other part is used to remove as much artifacts as possible. For the problems considered in this paper, data available represent around 20 KB of data per second if the data is

accessed in its binary form and 0.5 MB per second if the data is accessed in its text form. This scale requires solving an optimization problem every second. So the big optimization studied in this paper includes big volume of data and solving the problem quickly by ensuring quality of solutions. However, as the time constraint was not imposed in this version of problems as described in [10], we have demonstrated the problem solving ability for a single period. In practice, the process can be used repeatedly for multiple-period in a continuous time scale. Each optimization problem is based on a number of interdependent time series forming a dataset. The number of time series, together with the length of each time series, defines the number of variables in the problem. Each time series is of length 256; thus, each optimization problem has a multiple of 256 variables. The number of variables considered in this paper are 1024, 3072 and 4864. The solutions obtained are compared with the same several algorithms with the experimental results showing its superiority to all the algorithms considered.

This paper is organized as follows. An overview of DE is provided in Sect. 2. The proposed algorithm is then described in Sect. 3. The description of the problem and the simulation results are given in Sect. 4. Finally, the conclusions and future research directions are elaborated in Sect. 5.

2 Differential evolution

DE is as a powerful global search algorithm for real parameter optimization. DE differs from other EAs mainly in its generation of new vectors by adding the weighted difference vector (DV) between two individuals to a third individual [27]. It performs well when the feasible patches are parallel to the axes. Its main operators are briefly discussed below.

- Mutation: A mutant vector is generated by multiplying an amplification factor, F , by the difference between two random vectors and the result is added to a third random vector (DE/rand/1)

$$\vec{v}_z = \vec{x}_{r_1,j} + F \cdot (\vec{x}_{r_2,j} - \vec{x}_{r_3,j}) \quad (1)$$

where r_1, r_2, r_3 are different random integer numbers $\in [1, PS]$ and none of them is similar to z and PS the population size. As the mutation operator has a great effect on the performance of DE, different types have been introduced in the literature, such as DE/best/1 [26], DE/rand-to-best/1 [19] and DE/current-to-best [36].

- Crossover: There are two well-known crossover schemes, exponential and binomial. In the exponential crossover, firstly, an integer index, l , is randomly selected from a range $[1, D]$, where D is the problem dimension. This index acts as an initial position in the target vector from

where an exchange of variables with the donor vector begins. An integer index, L , that defines the number of components the donor vector contributes to the target vector, is randomly selected, such that $L \in [1, D]$. Subsequently, a trial vector is calculated as follows:

$$u_{z,j} = \begin{cases} v_{z,j} & \text{for } j = \langle l \rangle_D, \langle l + 1 \rangle_D, \dots, \langle l + L - 1 \rangle_D \\ x_{z,j} & \forall j \in [1, D] \end{cases} \quad (2)$$

where $j = 1, 2, \dots, D$ and $\langle l \rangle_D$ denotes a modulo function with a modulus of D and a starting location of l .

On the other hand, the binomial crossover is conducted on every variable with a predefined crossover probability, such that:

$$u_{z,j} = \begin{cases} v_{z,j} & \text{if } (rand \leq cr \text{ or } j = j_{rand}) \\ x_{z,j} & \text{otherwise} \end{cases} \quad (3)$$

$j_{rand} \in \{1, 2, \dots, D\}$ is a randomly selected index, which ensures \vec{u}_z gets at least one component from \vec{v}_z .

- Selection The selection process is simple, in which an offspring survives to the next generation, if it is better than its parent.

As mentioned earlier, many research studies have been proposed to deal with the fact that no single DE control parameter and/or search operator is the best for all kinds of optimization problems. However, pushing such mechanisms to solve big data problems has not been explored yet. Some of existing DE algorithms are discussed below.

Caraffini et al. [2] proposed a super fit multi-adaptive DE for solving unconstrained problems. In their proposed algorithm, 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. Additionally, F and Cr were generated using Cauchy and normal distributions, respectively. Both parameters were then adapted during the evolutionary process. Furthermore, a covariance matrix adaptive evolution strategy was used as a local search.

Elsayed et al. [6] proposed a general framework that divided the population into four sub-populations, each of which used one combination of search operators. During the evolutionary process, the sub-population sizes were adaptively varied based on the success of each operator, which was calculated based on changes in the fitness values, constraint violations and the feasibility ratio. The algorithm performed well on a set of constrained problems. Elsayed et al. [8] also proposed two novel DE variants, each of which utilized the strengths of multiple mutation and crossover operators, to solve constrained problems. The algorithms demonstrated

superior performance in comparison with the state-of-the-art algorithms.

Zamuda and Brest [35] introduced an algorithm that employed two mutation strategies in jDE [1], with population size adaptively reduced during the evolutionary process. The algorithm was tested on 22 real-world applications and performed better than two other algorithms.

Tvrđík and Polakova [31] introduced a DE algorithm for solving a set of constrained optimization problems. In it, with a predefined probability (q), one set of control parameters, from 12 available, was selected and during the evolutionary process, q was updated based on its success rate in previous steps. It was evaluated using a set of unconstrained problems and showed competitive performance [32].

Wang et al. [33] introduced a composite DE algorithm (CoDE), in which at 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.

In [15], a framework which used a mix of mutation strategies and discrete control parameters within DE was proposed for solving 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.

An improved adaptive DE algorithm was introduced [3], in which a mechanism was used to reduce the population size, along with using four mutation strategies with F and Cr were adapted using the Cauchy distribution.

Qiu et al. [22] proposed an adaptive cross-generation DE (ACGDE) for solving multi-objective problems. In it, two new mutation strategies were proposed to enhance both the convergence speed and diversity maintenance. The former was the neighborhood-based cross-generation (NCG) mutation which generated mutant vectors based on the difference between two individuals from consecutive generations, while the later was the population-based cross-generation (PCG), which incorporated the use of the entire set of individuals from two consecutive generations to produce new individuals. Both the NCG and PCG mutation operators were employed in a half-half manner. In addition, a self-adaptive mechanism was used to adapt both F and Cr values. The algorithm implemented into non-dominated sorting genetic algorithm II (NSGA-II) [4] and showed its superiority to many other algorithms.

Guo et al. [12] proposed a mechanism to properly select parents to participate in the mutation and crossover processes. In it, successful solutions were stored in an archive, then parents were selected from the archive when stagnation was occurred. The algorithm was employed with many DE variants and showed its ability to improve their results. In [13], an eigenvector-based crossover operator was proposed. It utilized eigenvectors of covariance matrix of individual solu-

tions to make the crossover rotationally invariant, that is the crossover operator exchanged information between the target and donor vectors in the eigenvector basis instead of the natural basis. The proposed crossover was employed with several algorithms and showed its effectiveness in solving 54 optimization problems.

Tang et al. [30] introduced a DE variant with an individual-dependent mechanism which incorporated an individual-dependent parameter (IDP) setting and an individual-dependent mutation (IDM) strategy. IDP setting ranked individuals based on their fitness values and then their assigned F and Cr were calculated based on each individual's ranking. In the IDM strategy, four vectors differences mutation were used to generate new offspring. The algorithm was tested on 28 unconstrained problems, with the results demonstrated that it was superior to state-of-the-art algorithms.

3 Automated DE framework

In this section, an automated DE framework (ADEF) is described.

3.1 ADEF

As described earlier, the main goals of the proposed algorithm are (1) extending DE for optimizing big data problems; and (2) developing a DE framework to automatically configure the best set of operators to use, as no single DE operator is the best for solving all types of optimization problems [9, 34].

The main steps of the proposed framework are presented in Algorithm 1. Firstly, κ DE variants are considered, where all of them start with the same population of individuals of size PS . Assume the population size of each DE variant is $PS_1, PS_2, \dots, PS_\kappa$, respectively. Subsequently, the solutions of each population are evolved by $DE_1, DE_2, \dots, DE_\kappa$, respectively. At each generation, information about each operator's performance is recorded to be used in the automatic selection step (Sect. 3.4). Once the best DE variant is determined, an information sharing scheme is adopted, such that

- For *single objective* problems: all $PS_{i=1:\kappa}$ are combined and sorted based on the fitness values. Then, redundant vectors are removed, while unique vectors are kept in x_{unique} . Subsequently, the best PS vectors from x_{unique} are selected.
- For *multi-objective* problems: all $PS_{i=1:\kappa}$ are combined. Then, the unique vectors are kept in x_{unique} . From x_{unique} , the non-dominated solutions are determined. Then, the first PS non-dominated solutions are selected. If the number of non-dominated solutions is less than PS , random solution vectors are selected from the remaining set of solutions from (\vec{x}_{unique}) to form the final PS .

Algorithm 1 The general steps of ADEF

```

1:  $PS \leftarrow$  Generate an initial population;
2:  $PS_1, PS_2, \dots, PS_\kappa \leftarrow PS$ ;
3:  $t \leftarrow 1$ ;
4: while  $t < t_{max}$  do
5:   while  $iter < CS$  do
6:     for  $i = 1 : \kappa$  do
7:       Evolve  $PS_i$  using the corresponding mutation, crossover and
       selection operators of  $DE_i$ ;
8:       Update  $cfe$ ;
9:       if the problem has a single objective then
10:        Record the performance based on  $refeq:sosuccess$ .
11:       else if the problem has multiple objectives then
12:        Record the performance based on  $refeq:success$ .;
13:       end if
14:     end for
15:      $t \leftarrow t + 1$ ;
16:   end while
17: Determine the best DE variant (Section 3.4);
18: Information sharing, as described in Section 3.1.
19: Apply a local search procedure;
20: Evolve the new population using the best DE variant for the
    remaining  $t_{max} - t$  fitness evaluations;
21: end while

```

Then, a local search procedure is applied to the best individual, in case of a single optimization problem, or a solution from the Pareto frontier, in case of a multi-objective problem. Subsequently, the best DE variant is used to evolve the entire population till the end of the evolutionary process.

3.2 DE variants

In this paper, three DE variants (three mutation operators with the binomial crossover) are considered

- DE_1 : DE/current-to-rand/1/bin

$$u_{z,j} = \begin{cases} x_{z,j} + F_z \cdot (x_{r_1,j} - x_{z,j} + x_{r_2,j} - x_{r_3,j}) \\ \text{if } (rand \leq cr_z \text{ or } j = j_{rand}) \\ x_{z,j} \quad \text{otherwise} \end{cases} \quad (4)$$

- DE_2 : DE/current-to- ϕ best/1/bin

$$u_{z,j} = \begin{cases} x_{z,j} + F_z \cdot (x_{\phi,j} - x_{z,j} + x_{r_1,j} - x_{r_2,j}) \\ \text{if } (rand \leq cr_z \text{ or } j = j_{rand}) \\ x_{z,j} \quad \text{otherwise} \end{cases} \quad (5)$$

- DE_3 : DE/rand/1/bin

$$u_{z,j} = \begin{cases} x_{r_1,j} + F_z \cdot (x_{r_2,j} - x_{r_3,j}) \\ \text{if } (rand \leq cr_z \text{ or } j = j_{rand}) \\ x_{z,j} \quad \text{otherwise} \end{cases} \quad (6)$$

Algorithm 2 MOEA/D

- 1: Generate uniform weight vectors and identify the neighborhood B_z of each individual $z = 1 : PS$;
- 2: Evaluate the initial population and update the ideal reference point (f^*);
- 3: **for** each subproblem z **do**
- 4: Select r_1, r_2 and r_3 from B_z ;
- 5: ϕ is randomly selected from the best [1-50%] individuals in B_z , Eq. 5;
- 6: Generate new offspring using Eqs. 4 to 6;
- 7: Replace a maximum of two individuals from the randomly mixed $B(z)$, when their fitness values are worse than the new offspring;
- 8: Update the ideal reference point (f^*);
- 9: **end for**

where \vec{x}_ϕ is a random vector selected from the range $[1, \phi]$, $r_1 \neq r_2 \neq r_3$ are random integer numbers and each one is different from z , while F_z and Cr_z are self-adaptively updated, as will be shown later.

3.3 Evolving individuals in multi-objective problems

To adapt the algorithm to handle a multi-objective problem, a multi-objective EA based on decomposition (MOEA/D) [20,21] is considered. The main idea behind MOEA/D is to decompose a multi-objective problem into a number of single objective optimization problems. A Tchebycheff scalarization function is adopted and then ADEF is applied, as described above.

$$\text{minimize } g(x|\lambda, f^*) = \text{maximize } \lambda|f_z - f_z^*$$

Subject to

$$x \in \Omega \tag{7}$$

where $\vec{\lambda} = (\lambda_1, \dots, \lambda_m)$ is a weight vector representing the weights for each of the m objectives to be optimized, such that $\sum_{z=1}^m \lambda = 1$ and $f^* = (f_1^*, \dots, f_m^*)$ is a vector representing the ideal objective values for m objectives. MOEA/D utilizes the relations between neighborhood among all sub-problems, optimization of a sub-problem by using the information from its neighboring sub-problems. The main steps of the adapted MOEA/D used in this paper are presented in Algorithm 2.

3.4 Selection of best DE variant

To decide the best DE variant, the following steps are conducted:

- At each iteration (t), and till the algorithm reaches CS generations, information from every DE is recorded, such that

- For *single objective* problems: the average fitness value of every $PS_{i=1:k}$ is recorded

$$\text{inf}_{k,t} = \frac{\sum_{z=1}^{PS_k} f_{z,t}}{PS_k} \tag{8}$$

- For *multi-objective* problems: information based on the quality of solutions and number of non-dominated solutions (num_{nds}) is recorded

$$\text{inf}_{k,t} = \frac{\sum_{z=1}^{PS_k} \sum_{obj=1}^m \lambda_{z,obj} f_{z,obj,t}}{PS_k} + \left(\frac{1}{num_{nds}} \right) \tag{9}$$

Note that the smaller the value of inf , the better-performing the DE variant.

- For each DE variant, the exponential curve is fitted, such that

$$y = ae^{bt} \tag{10}$$

- Find a and b ; then calculate the predicted values (\hat{y}).
- Calculate the goodness of fit or coefficient of determination to find the likelihood of future events falling within the predicted outcomes as follows:

$$R^2 = \left(\frac{n \sum y\hat{y} - \sum x \sum \hat{y}}{\sqrt{n \sum (y^2) - (\sum y)^2} \sqrt{n \sum (\hat{y}^2) - (\sum \hat{y})^2}} \right)^2 \tag{11}$$

where y is the actual values and \hat{y} is the predicted ones.

- For each DE variant, calculate the expected performance after the subsequent CS generations such that

$$Ef_{v_{i=1:k}} = \frac{a_i e^{b_i \times 2 \times CS}}{R_i^2} \tag{12}$$

It is very common that the best fitness value, within the population, improves rapidly at the early stages of the evolutionary process, while the improvement rate is too slow at later stages. From the pattern of improvement rate, one can argue that rate is decreasing exponentially, which motivated us to use an exponential function for prediction. However, other patterns can be considered which will be investigated in our future studies.

- Consequently, the one with the minimum expected value is considered the best and is selected to evolve its population till the end of the evolutionary process.

3.5 Adaptation of F and Cr

It is a fact that F and Cr have a great impact on the performance of any DE. In this paper, the mechanism proposed in [29], which is considered as an improvement of JADE [36], is adopted. There are some changes done to improve its performance on the problem on hand.

- A historical memory with H entries for both parameters (M_{Cr}, M_F) is initialized, where all values are set to a value of 0.5.
- Each individual x_z is associated with its own Cr_z and F_z , such that

$$Cr_z = \text{randni}(M_{Cr,r_z}, \sigma_{cr}) \tag{13}$$

$$F_z = \text{randci}(M_{F,r_z}, 0.1) \tag{14}$$

where r_z is randomly selected from $[1, H]$, and if a value is outside the range of $[0,1]$, it is handled by the technique used in [36], $\text{randni}(\mu, \sigma)$ and $\text{randci}(\mu, \sigma)$ are values randomly selected from normal and Cauchy distributions with mean μ and variance σ , σ_{cr} is set at a value of 0.01, while its original value was 0.1. To add to this, F_z is set to be within $[0.2-0.8]$.

- Similar to JADE, at the end of each generation, the Cr_z and F_z values used by the successful individuals are recorded in S_{Cr} and S_F , and then the contents of memory are updated as follows

$$M_{Cr,d,iter} = \begin{cases} \text{mean}_{WA}(S_{Cr}) & \text{if } S_{Cr} \neq \text{null} \\ M_{Cr,d,iter} & \text{otherwise} \end{cases} \tag{15}$$

$$M_{F,d,iter} = \begin{cases} \text{mean}_{WL}(S_F) & \text{if } S_F \neq \text{null} \\ M_{F,d,iter} & \text{otherwise} \end{cases} \tag{16}$$

where $1 \leq d \leq H$ is the the position in the memory to be updated. It is initialized to 1, and then incremented whenever a new element is inserted into the history, and if it is greater than $H=6$, it is set to 1. The weighted mean($\text{mean}_{WA}(S_{Cr})$) is computed as:

$$\text{mean}_{WA}(S_{Cr}) = \sum_{zd=1}^{S_{cr}} w_{zd} \cdot S_{cr,zd} \tag{17}$$

$$w_{zd} = \frac{\Delta f_{zd}}{\sum_{zd=1}^{S_{cr}} \Delta f_{zd}} \tag{18}$$

where $\Delta f_{zd} = |f(u_{zd,iter}) - f(x_{zd,iter})|$. The weighted Lehmer mean ($\text{mean}_{WL}(S_F)$) is calculated as follows:

$$\text{mean}_{WL}(S_F) = \frac{\sum_{zd=1}^{S_F} w_{zd} \cdot S_{F,zd}^2}{\sum_{zd=1}^{S_F} w_{zd} \cdot S_{F,zd}} \tag{19}$$

and w_{zd} is calculated similarly to Eq. 18.

3.6 Local search procedure

In this paper, the interior point method is applied as a local search. “*The interior point methods generate one or more interesting search directions at each iteration and effectively combine these directions to ensure that sufficient progress in a suitable convergence function is made*” [17]. For constrained problems, interior-point methods (1) starts with adding slack variables to make all inequality constraints into non-negativities; (2) then nonnegativity constraints are replaced with logarithmic barrier terms in the objective function. The idea of the barrier approach is to start from a point in the strict interior of the inequalities and construct a barrier that prevents any variable from reaching a boundary; (3) the equality constraints are incorporated into the objective using Lagrange multipliers; (4) solve the optimality conditions for the barrier problems using Newton’s method.

- For single objective problems, considering the optimization of Eq. 24, the interior point method is applied to the best solution obtained by of the successful DE variant a d for a predefined number of fitness evaluations.
- Similarly, in case of a multi-objective problem, a single solution from the Pareto Front is passed to the interior point to be optimized, bearing in mind the optimization of Eq. 24. The solution obtained is then evaluated based on Eqs. 22 and 23. Subsequently, the ideal reference point is updated, if required.

4 Experimental analysis

In this section, the simulation results obtained by the proposed algorithm are presented and discussed. The algorithm was coded in Matlab R2012b¹ and run on a PC with a 3.4 GHz Core I7 processor with 16 GB RAM, and Windows 7.

4.1 Problem description

The series of problems used in this paper are both single objective and multi-objective nonlinear optimization problems [10,11]. Each optimization problem is based on a number of interdependent time series forming a dataset. The number of time series together with the length of each time series define the number of variables in the problem. For this dataset, each time series is of length 256, where the number of times series are 4, 12 and 19, hence there are 1024, 3072 and 4864 variables, respectively.

¹ The code is available upon request.

Table 1 Computational results of ADEF, DECC-DG, JADE, SHADE and NSGA-II for single objective problems

Problem	Best fitness				Average fitness				Std.				Baseline			
	ADEF	DECC-DG	JADE	SHADE	ADEF	DECC-DG	JADE	SHADE	ADEF	DECC-DG	JADE	SHADE	ADEF	DECC-DG	JADE	SHADE
	D4	6.13E-02	3.47E+00	9.58E-01	7.71E-01	6.13E-02	3.82E+00	1.10E+00	8.97E-01	2.97E-06	2.26E-01	8.37E-02	6.51E-02	1.8709E+00		
D12	2.22E-03	6.96E+00	2.04E+00	1.81E+00	2.39E-03	7.16E+00	2.32E+00	2.09E+00	8.73E-05	9.86E-02	1.10E-01	9.39E-02	2.9294E+00			
D19	6.21E-03	2.66E+02	2.49E+00	2.29E+00	7.61E-03	2.70E+02	2.73E+00	2.45E+00	7.48E-04	2.05E+00	1.11E-01	9.52E-02	3.1854E+00			
D4N	5.93E-02	8.24E+00	9.53E-01	7.98E-01	5.93E-02	8.89E+00	1.11E+00	9.07E-01	1.87E-06	3.18E-01	7.71E-02	5.04E-02	1.7414E+00			
D12N	2.14E-03	7.15E+00	2.15E+00	1.85E+00	2.28E-03	7.39E+00	2.32E+00	2.08E+00	8.25E-05	8.10E-02	1.24E-01	8.81E-02	2.8236E+00			
D19N	6.54E-03	2.69E+02	2.46E+00	2.29E+00	7.88E-03	2.73E+02	2.66E+00	2.45E+00	7.45E-04	2.16E+00	1.02E-01	9.33E-02	3.1659E+00			

Bold values represent the best results obtained

Assume that a matrix X is of dimension $N \times M$, where N is the number of inter-dependent time series and M is the length of each time series. Let S be a matrix of $N \times M$, with N independent time series of length M , such that, given A , an $N \times N$ linear transformation matrix

$$X = A.S \tag{20}$$

The problem is to decompose S into two matrices: S_1 and S_2 with the same dimensionality of S , that is $S = S_1 + S_2$ and $X = A.S_1 + A.S_2$. Let C be the Pearson correlation coefficient between S_1 and X

$$\frac{cov(X, A.S_1)}{\sigma(X).\sigma(A.S_1)} \tag{21}$$

where, $cov(.)$ and $\sigma(.)$ are the covariance matrix and the variance, respectively. The objective is to maximize the diagonal elements of C , while minimizing off-diagonal elements to zeros. At the same time, the distance between S and S_1 should be as minimum as possible, that is S_1 needs to be similar as possible as S .

This definition generates two formulations for the problem, one as a single objective, while the other as a multi-objective problem.

- A multi-objective optimization problem is defined as follows:
Given, X , A , and S , find S_1 which optimizes the following two functions:

Table 2 Scores of proposed algorithm, in reference to baseline algorithm, for single objective problems

Problem	Best	Average	Std.
D4	1.81E+00	1.81E+00	1.52E-07
D12	2.93E+00	2.93E+00	8.73E-05
D19	3.18E+00	3.18E+00	7.48E-04
D4N	1.68E+00	1.68E+00	1.31E-07
D12N	2.82E+00	2.82E+00	8.25E-05
D19N	3.16E+00	3.16E+00	7.45E-04

Table 3 Scores of proposed algorithm, in reference to baseline algorithm, for multi-objective problems

Problem	Best	Average	Std.
D4	2.25E+01	2.25E+01	5.84E-03
D12	2.31E+01	2.31E+01	1.13E-03
D19	2.34E+01	2.34E+01	2.76E-03
D4N	2.19E+01	2.19E+01	7.39E-03
D12N	2.37E+01	2.37E+01	8.99E-04
D19N	2.20E+01	2.20E+01	1.99E-03

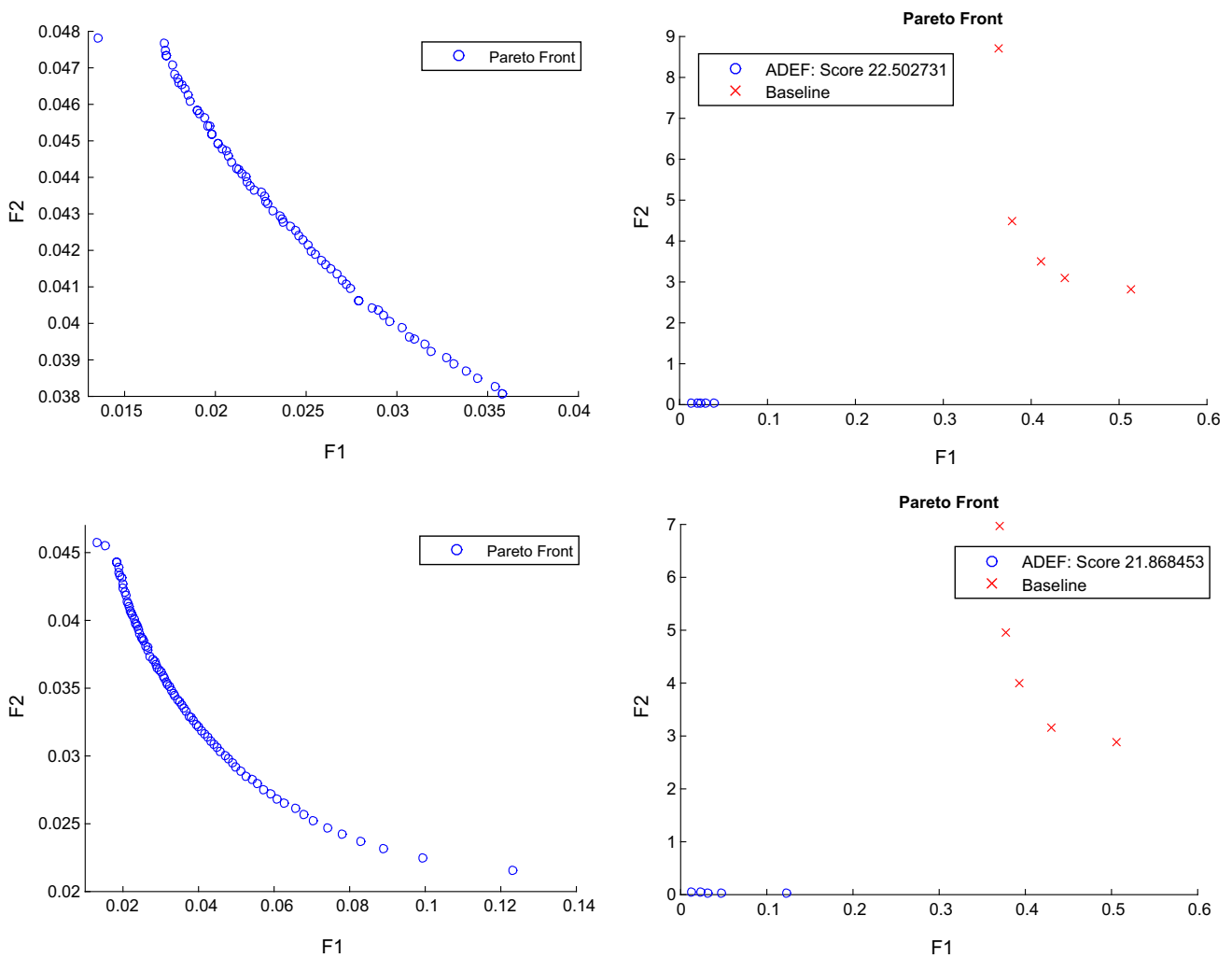


Fig. 1 Pareto frontier obtained by ADEF for D4 (first row) and D4N (second row) (figures presented in second column are for five solutions sampled uniformly from Pareto set and corresponding nearest solutions found by baseline algorithm)

$$\text{Minimize } f_1 = \frac{1}{(N.M)} \sum_{ij} (S_{ij} - S_{1,ij})^2 \tag{22}$$

$$\text{Minimize } f_2 = \frac{1}{(N^2 - N)} \sum_{i,j \neq i} (C_{ij}^2) + \frac{1}{N} \sum_i (1 - C_{ii})^2 \tag{23}$$

– The single objective formulation is defined as follows:

$$\begin{aligned} &\text{Minimize } f_1 + f_2 \\ &\text{Subject to } -8 \leq S_1 \leq 8 \end{aligned} \tag{24}$$

4.2 Parameter settings

φ was randomly selected within the range [1, 10 %PS] and [1, 50 %PS] for single and multi-objective problems, respectively. The reason for increasing this range was to maintain diversity. The maximum number of generations (t_{max}) was set to a value of 300, 400 and 500 iterations, for problems

with 1024, 3072 and 4864 variables, respectively, and each value was increased by 100 generations for multi-objective problems. CS was equal to 25 iterations. An extra parameter of MOEA/D was the neighborhood size which was set to a value of $\frac{PS}{5}$. Lastly, 30 runs/seeds were conducted.

4.3 Results and comparison with state-of-the-art algorithms

The results obtained are discussed in this subsection and compared several algorithms from the literature.

4.3.1 Single objective problems

In this subsection, ADEF is compared with the following algorithms:

1. A baseline algorithm which is based on NSGA-II. As we solve single objective problem, NSGA-II was used as it

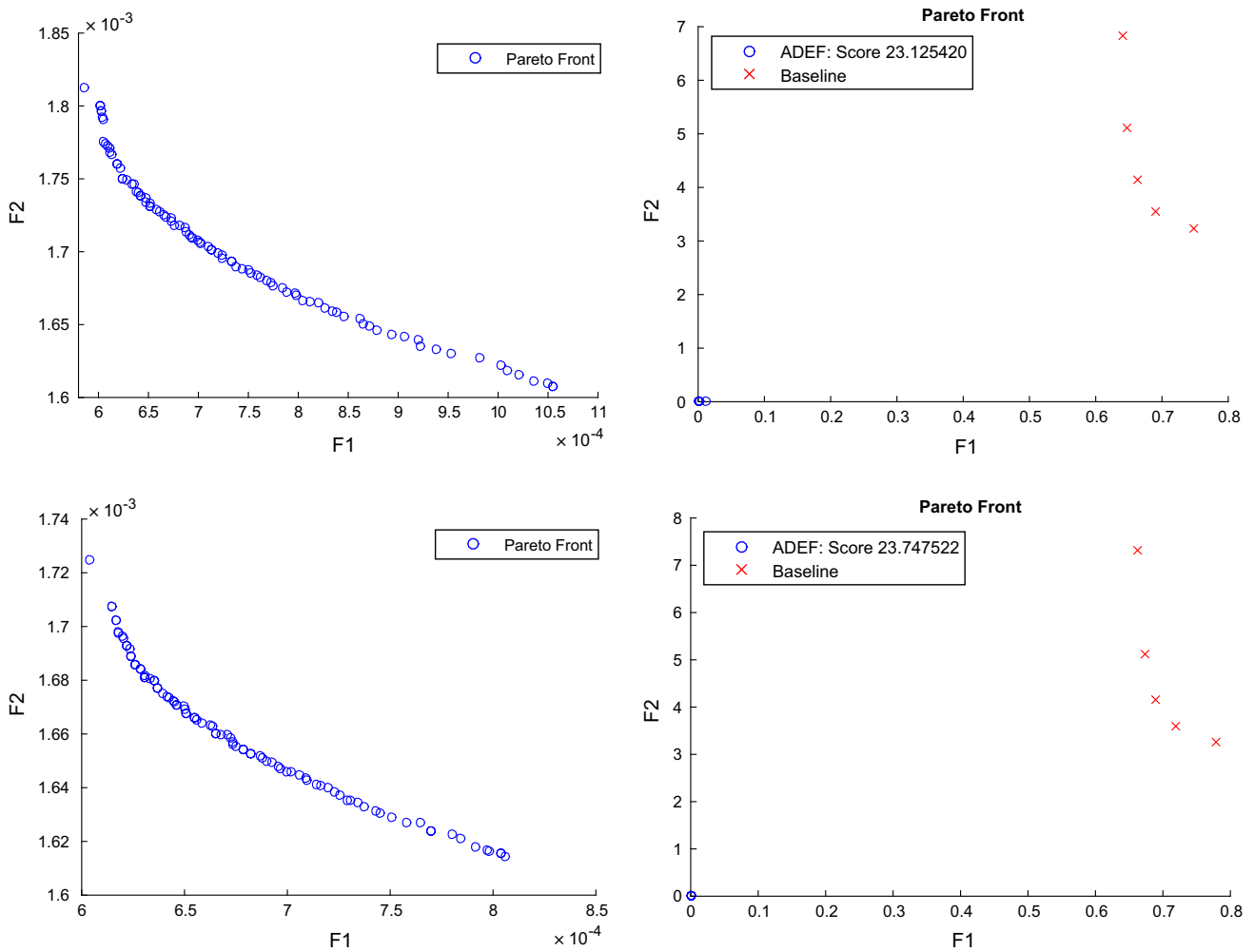


Fig. 2 Pareto frontier obtained by ADEF for D12 (first row) and D12N (second row) (figures presented in second column are for five solutions sampled uniformly from Pareto set and corresponding nearest solutions found by baseline algorithm)

is, while zeroing the second objective, as suggested by [10, 11].

2. A cooperative co-evolution DE with differential grouping algorithm (DECC-DG) [18], which used an automatic decomposition strategy to uncover the underlying interaction structure of the decision variables. All the parameters were set as mentioned in the published paper.
3. An adaptive DE algorithm with external archive (JADE) [36], where PS was set to a value of 100.
4. Success-history based adaptive DE (SHADE) [28], which is an improved version of JADE.

Firstly, as described in [10], the overall best solution found over 30 runs was considered as a measure of quality. Also, the stability of the algorithm was assessed using the variance of the set of best solutions found over 30 runs. The detailed results [best, average and standard deviation (*Std.*)] are shown in Table 1. From these results, it is clear that the proposed algorithm was much better than the other algorithms. Addi-

tionally, the standard deviation results obtained revealed that the proposed algorithm was stable.

Furthermore, the score of the proposed algorithm, in reference to the baseline algorithm, for each test function was calculated based on the procedure provided in [10]

$$score = \begin{cases} bf v_{bl} - bf v & \text{if } bf v < bf v_{bl} \\ -1000(bf v_{bl} - bf v) & \text{otherwise} \end{cases} \quad (25)$$

where $bf v$ and $bf v_{bl}$ are the best fitness values obtained by any variant and the baseline algorithm, respectively.

Based on the results shown in Table 2, it is clear that the proposed algorithm was the best.

4.3.2 Multi-objective problems

To measure the quality of the proposed algorithm, five solutions were sampled uniformly from the Pareto set, including the two extreme solutions. The average distance between

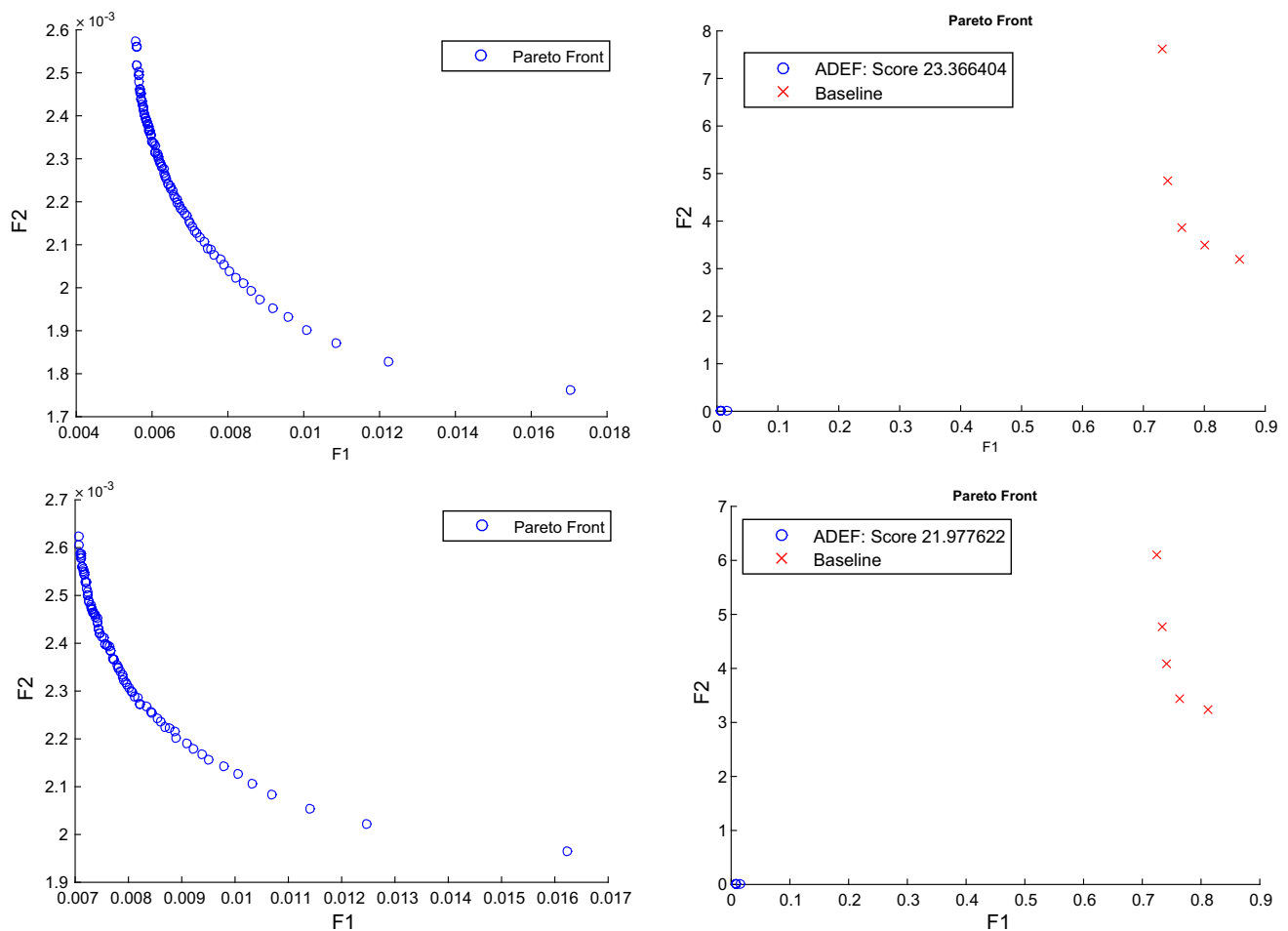


Fig. 3 Pareto frontier obtained by ADEF for D19 (first row) and D19N (second row) (figures presented in second column are for five solutions sampled uniformly from Pareto set and corresponding nearest solutions found by baseline algorithm)

those five solutions and corresponding nearest solutions found in the baseline algorithm (NSGA-II, as reported in [10]) were recorded. Only those solutions found by the proposed algorithm that dominated the corresponding solution in the baseline were considered. For those solutions found which were dominated by the corresponding solutions in the baseline, a fixed distance value of -1000 was used. Larger distance values mean the algorithm was better. For problems with noise, the evaluation was done using the data without noise [10]. The detailed results are recorded in Table 3, which revealed that the proposed algorithm was far better than the baseline algorithm.

To give more information about the algorithm performance, the Pareto frontiers obtained, of a random seed/run, for every problem is presented in Figs. 1, 2 and 3. From these figures, it is clear that the algorithm was better.

ADEF was also compared with MOEA/D based on DE (MOEA/D-DE) [20,21], where DE/rand/1 was employed with F and Cr were set to a value of 0.5 and the other parameters set as discussed in Sect. 4.2. Both algorithms were

compared in reference to the baseline algorithm and the algorithm with a higher average score was considered the best. The results shown in Table 4 clearly reveal that ADEF was better than MOEA/D-DE. Furthermore, the Pareto frontiers obtained by both methods are depicted in Figure 4 (x -axis and y -axis are in log scales), demonstrating that solutions obtained by ADEF dominated those of MOEA/D-DE.

4.4 Parameters analysis

In this subsection, CS and t_{max} are analyzed.

4.4.1 CS

Here, effects of CS on the selection of the most suitable DE, as described in Sect. 3.4, as well as the final results obtained, are analyzed. To do this, the algorithm was run with different values of CS , i.e., $CS = 10, 25$ (default), and 50 generations. All other parameters were set as those described in Sect. 4.2.

For single objective problems, considering the computational results obtained, as presented in Table 5, it was found that all variants were better than the baseline algorithm. Among them, ADEF with $CS = 25$ was the best for D12 and D12N, while using $CS = 50$ was the best for D19 and D19N, and no significant differences were found in solving problems with 1024 variables. For multi-objective problems, there were no significant differences in scores. The Pareto front plots are depicted in Fig. 5. From this figure, it can be concluded that when $CS = 25$ the Pareto front was evenly distributed along the two objectives for D4 and D12, while setting CS to a value of 50 was better for D19 and D19N.

Table 4 Average scores of ADEF and MOEA/D-DE for multi-objective problems

Problem	ADEF	MOEA/D-DE
D4	2.25E+01	7.78E+00
D12	2.31E+01	2.74E+00
D19	2.34E+01	3.15E+00
D4N	2.19E+01	7.81E+00
D12N	2.37E+01	3.57E+00
D19N	2.20E+01	1.69E+00

Bold values represent the best results obtained

It was also interesting to know why the results changed when using different CS values. One reason for this issue was the selection of the best DE variant. In other words, a bigger CS value meant more information was captured about each DE variant, which could help fitting the exponential curve well and hence the most preferred DE was properly selected. The results are presented in Table 6. Based on this analysis, for single objective problems, it was noticed that DE_3 was not preferred for any test problem with all CS values. For a small value of CS , DE_2 was more frequently preferred followed by DE_1 . However, when CS value was increased, DE_1 became first. For the multi-objective problems, it was noticed that DE_3 was selected more times than DE_2 when $CS = 10$, but DE_2 was considered better than DE_1 when CS was increased. However, it was noticed that DE_1 was the most preferred algorithm for the majority of test problems.

It was also good to show the running time, consumed by ADEF with different values of CS . Table 7 summarizes the results obtained regarding this issue. Note that the code has been written in Matlab, hence the computational time could be further reduced if the code was written in C++. From Table 7, ADEF with $CS = 10$ was faster than the other variants. For the multi-objective problems, the computational time was increased, in comparison with those of single objective prob-

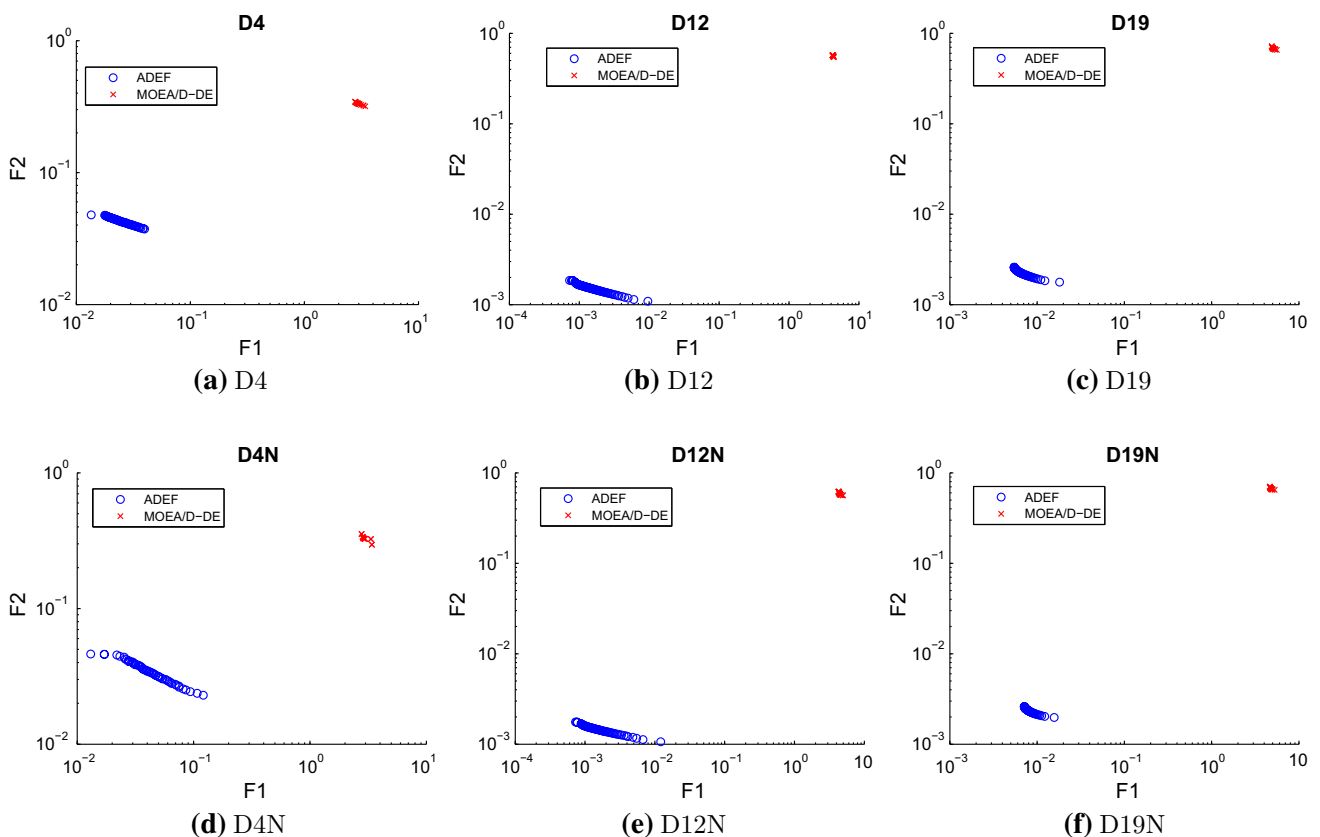


Fig. 4 Pareto frontiers obtained by ADEF and MOEA/D-D (x-axis and y-axis in log scales)

Table 5 Computational results of ADEF, using different values of CS , for single objective problems

Problem	Best			Average			Baseline
	$CS = 10$	$CS = 25$ (default)	$CS = 50$	$CS = 10$	$CS = 25$ (default)	$CS = 50$	
D4	6.13E-02	6.13E-02	6.13E-02	6.13E-02	6.13E-02	6.14E-02	1.8709E+00
D12	3.81E-03	2.22E-03	3.03E-03	4.29E-03	2.39E-03	3.74E-03	2.9294E+00
D19	5.68E-03	6.21E-03	4.12E-03	6.53E-03	7.61E-03	4.84E-03	3.1854E+00
D4N	5.93E-02	5.93E-02	5.93E-02	5.93E-02	5.93E-02	5.93E-02	1.7414E+00
D12N	3.62E-03	2.14E-03	2.86E-03	4.18E-03	2.28E-03	3.62E-03	2.8236E+00
D19N	5.84E-03	6.54E-03	3.99E-03	6.67E-03	7.88E-03	4.79E-03	3.1659E+00

Bold values represent the best results obtained

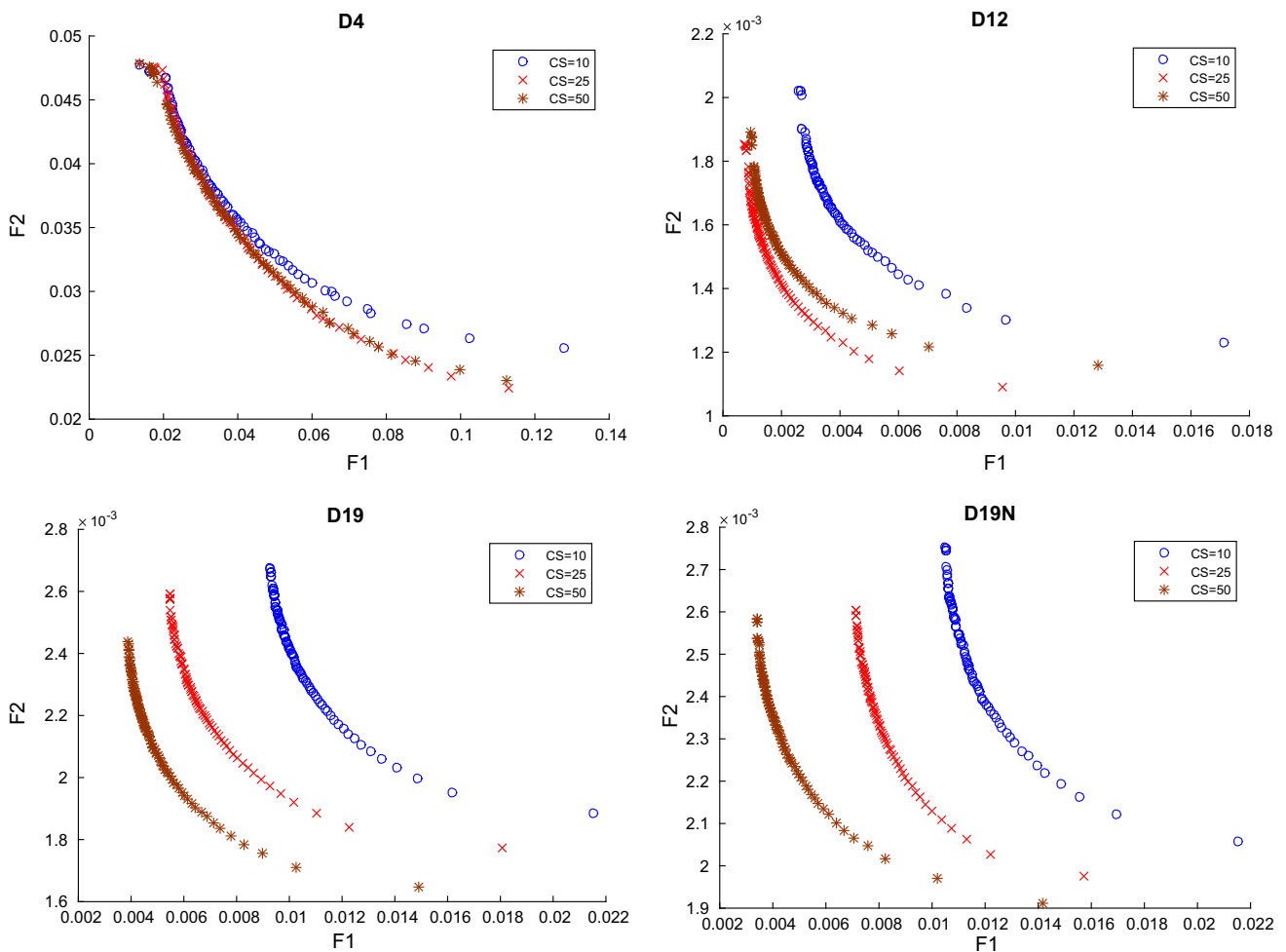


Fig. 5 Pareto frontier for D4, D12, D19 and D19N with different values of CS

lems. However, the results had a similar pattern to that for single objective problems.

4.4.2 t_{max} analysis

From the previous subsection, CS was set to a value of 25 for problems with 1024 and 3072 variables, while it was 50 for

4864 variables. Subsequently, ADEF was run with different values of t_{max} as presented in Table 8.

In regards to single objective problems, the best and average results are shown in Table 9. From the results obtained, it was found that in all cases the results were better than those of the baseline algorithm. Among the 3 cases, it was found that the results of case 1 were inferior to those of case 2 and

case 3. Also, the results of case 3 were slightly better than those of case 2. However, using a higher number of iterations meant more computational effort.

Considering the multi-objective problems, the final sets of non-dominated solutions for D4, D12, D19 and D19N are plotted in Fig. 6. From these plots, in general, case 2 and case 3 are better than case 1, and no much difference in the algorithm’s performance between case 1 and case 2 for solving D4 and D12. However, increasing the number of generations lead to better performance in solving D19 and D19N, but this came with the curse of computational time required.

Table 10 shows the computational time of ADEF with different values of t_{max} , which logically revealed that ADEF based on case 1 had the smallest computational time.

Table 6 Number of times a DE algorithm considered the best out of 30 runs for each test problem

Problems	CS	Single objective			Multi-objective		
		DE ₁	DE ₂	DE ₃	DE ₁	DE ₂	DE ₃
D4	10	14	16	0	14	3	13
	25	22	8	0	15	7	8
	50	26	4	0	27	2	1
D12	10	8	22	0	12	10	8
	25	18	12	0	18	7	5
	50	29	1	0	29	1	0
D19	10	13	17	0	11	12	7
	25	16	14	0	13	11	6
	50	29	1	0	30	0	0
D4N	10	11	19	0	13	9	8
	25	16	14	0	18	11	1
	50	22	8	0	29	1	0
D12N	10	12	18	0	18	4	8
	25	17	13	0	25	4	1
	50	28	2	0	30	0	0
D19N	10	13	17	0	11	9	10
	25	17	13	0	18	7	5
	50	29	1	0	30	0	0

Bold values represent the best results obtained

Table 7 Computational time, in seconds, of ADEF with different values of CS for single and multi-objective problems

Problem	Single objective			Multi-objective		
	CS = 10	CS = 25	CS = 50	CS = 10	CS = 25	CS = 50
D4	8.11E+00	8.75E+00	9.50E+00	1.17E+01	1.23E+01	1.40E+01
D12	3.03E+01	3.23E+01	3.52E+01	3.81E+01	4.10E+01	4.43E+01
D19	5.78E+01	6.01E+01	6.55E+01	6.89E+01	7.12E+01	7.82E+01
D4N	8.06E+00	8.33E+00	9.44E+00	1.17E+01	1.25E+01	1.40E+01
D12N	3.01E+01	3.19E+01	3.54E+01	3.85E+01	4.07E+01	4.42E+01
D19N	5.78E+01	6.01E+01	6.58E+01	6.89E+01	7.18E+01	7.78E+01

4.5 Improved ADEF and discussion

From Sect. 4.4.1, it was clear that DE₃ was not preferred in solving single objective problems. However, this was not the case in solving multi-objective problems. In addition, it was noted that DE₁ was not good enough during the exploitation stage (later stages of the optimization process) compared with DE₂, as shown in Fig. 7.

Therefore, for single objective problems, two changes were undertaken: (1) DE₃ was replaced with DE/current-to- ϕ best/1/bin with an archive [36] (DE₄), where the archive size was set to a value of 1.4PS. Note that DE₂ did not use such an archive; (2) In case that DE₁ was selected (as described in Sect. 3.4) and during the exploitation stage (here it is defined as the second half of the optimization process), it was replaced with DE/current-to- ϕ best/1/bin with an archive. This variant was run with and without using a local search, and named IADEF and IADEF-NLS, respectively. The stopping criterion was similar case 2, with CS was 25 for problems with 1024 and 3072 variables and 50 for 4864 variables, as mentioned in Sect. 4.4.2.

For a fair comparison, ADEF was also run with and without using a local search (ADEF-NLS). The experimental results of all the variants and state-of-the-art algorithms are shown in Table 11. From these results, it is clear that IADEF was superior to all other algorithms. In addition, IADEF-NLS was better than ADEF-NLS as well as all the other algorithms. Furthermore, it was clear that IADEF and ADEF

Table 8 Different values of t_{max}

Problems	Cases	D/ t_{max}		
		1024	3072	4864
Single objective	1	150	200	250
	2 (default)	300	400	500
	3	600	800	1000
Multi-objective	1	200	250	300
	2 (default)	400	500	600
	3	800	1000	1200

Table 9 Computational results of ADEF for single objective problems using different values of t_{max}

Problem	Best			Average			Baseline
	Case 1	Case 2 (default)	Case 3	Case 1	Case 2 (default)	Case 3	
D4	6.13E-02	6.13E-02	6.13E-02	6.18E-02	6.13E-02	6.13E-02	1.8709E+00
D12	2.28E-03	2.22E-03	2.15E-03	2.49E-03	2.39E-03	2.28E-03	2.9294E+00
D19	4.23E-03	4.12E-03	3.87E-03	5.07E-03	4.84E-03	4.46E-03	3.1854E+00
D4N	5.93E-02	5.93E-02	5.93E-02	5.96E-02	5.93E-02	5.93E-02	1.7414E+00
D12N	2.18E-03	2.14E-03	2.08E-03	2.39E-03	2.28E-03	2.18E-03	2.8236E+00
D19N	4.12E-03	3.99E-03	3.76E-03	5.04E-03	4.79E-03	4.43E-03	3.1659E+00

Bold values represent the best results obtained

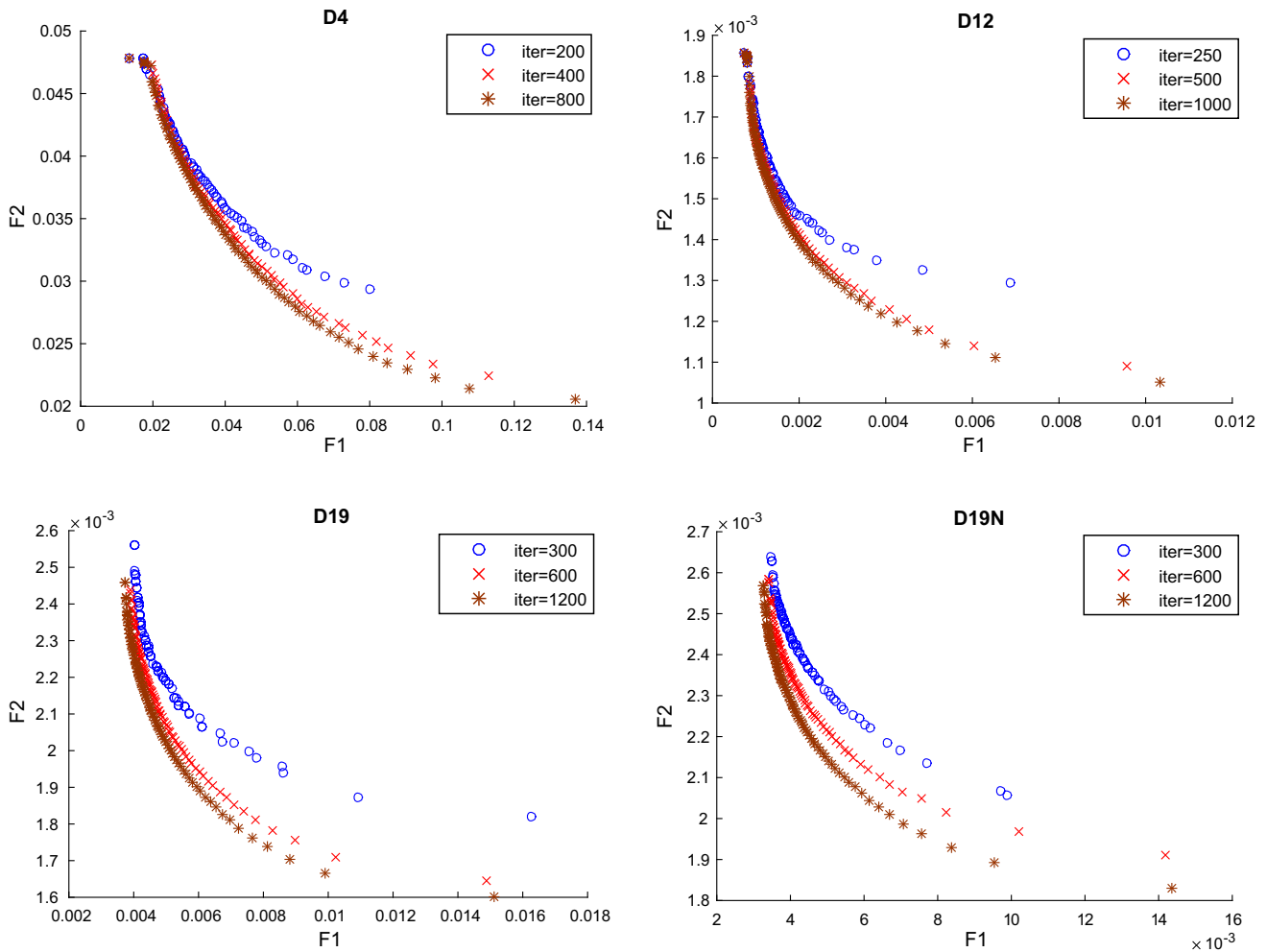


Fig. 6 Pareto frontier of ADEF for D4, D12, D19 and D19N with different values of t_{max}

with and without using a local search were better than DE₁, DE₂, DE₃ and DE₄.

It is also important to mention that we have taken the prediction accuracy into consideration to properly select the better-performing DE variant, as described by Eqs. 11 and 12, in which the predicted value, which could be achieved by each DE, was penalized, if the prediction accuracy was low. To add to this, we calculated the average prediction rate for

DE₁, DE₂ and DE₄ and found that the prediction rates were high, as depicted in Fig. 8. Also, it was noted that the function used for prediction had a great effect of the prediction accuracy. As an example, a linear function was considered for prediction, with its prediction accuracy was lower than that of the exponential function, as demonstrated in Fig. 8.

Generally speaking, it was noticed that for better results and convergence patterns, it was better to (1) apply a powerful

Table 10 Computational time, in seconds, of ADEF with different values of CS for single objective problems

Problem	Single objective			Multi-objective		
	Case 1	Case 2 (default)	Case 3	Case 1	Case 2 (default)	Case 3
	D4	5.86E+00	8.75E+00	1.436E+01	7.88E+00	1.23E+01
D12	2.04E+01	3.23E+01	5.609E+01	2.47E+01	4.10E+01	7.353E+01
D19	4.26E+01	6.55E+01	1.136E+02	4.97E+01	7.82E+01	1.405E+02
D4N	5.78E+00	8.33E+00	1.393E+01	7.97E+00	1.25E+01	2.297E+01
D12N	2.02E+01	3.19E+01	5.532E+01	2.49E+01	4.07E+01	7.442E+01
D19N	4.30E+01	6.58E+01	1.105E+02	4.95E+01	7.78E+01	1.368E+02

Bold values represent the best results obtained

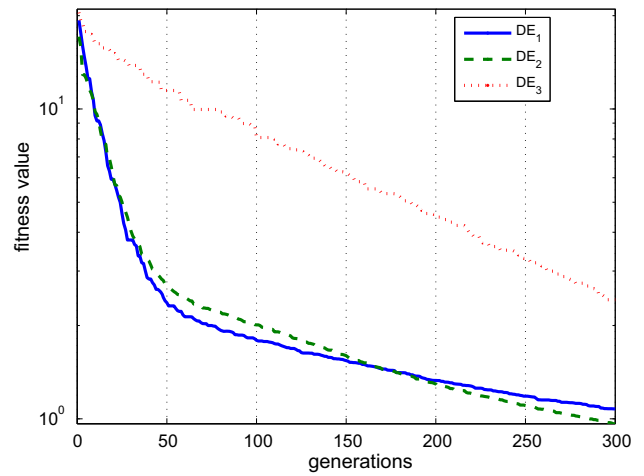


Fig. 7 Convergence of DE₁, DE₂ and DE₃ for D4 (y-axis in log scale)

local search; (2) properly select the right set of operators. Here, it was useful to avoid using DE/rand/1/bin for single objective problems; (3) design or use a powerful mechanism for adapting the control parameters; (4) carefully design a framework that combines all of these components together.

5 Conclusions and future work

In this paper, a differential evolution framework was proposed for optimizing the big data 2015 benchmark problems with both single and multi-objective problems. In the proposed algorithm, three differential evolution variants were run in parallel, each of which used to evolve its own population of individuals. After a predefined number of generations, exponential curve was fitted to predict the future performance of each variant. The one with the best predicted value was considered the winner. At the same time, an information sharing scheme was adopted as well as a local search procedure. Subsequently, the best differential evolution was continued to evolve the new set of individuals till the end of the optimization process.

The test functions used contained 1024, 3072 and 4864 variables. Based on the results obtained, the algorithm was found better than a baseline algorithm, for single and multi-objective problems. The algorithm was also compared with state-of-the-art algorithms as well as each independent DE variant used in the proposed framework. The results showed the superiority of the proposed algorithm. In addition, some parameters were analyzed which improved the algorithm’s performance. Also, it was noted that the proper selection of operators, control parameters and local search procedure had great effects on the algorithm’s performance.

For future work, we intend to use feature reduction mechanisms to reduce the complexity of problems and computational time.

Table 11 Computational results of IADEF, IADEF-NL, ADEF, ADEF-NLS, DECC-DG, JADE, SHADE, DE₁, DE₂, DE₃ and DE₄ for single objective problems

Problem	IADEF	ADEF	IADEF-NLS	ADEF-NLS	DECC-DG	JADE	SHADE	DE ₁	DE ₂	DE ₃	DE ₄
Best fitness											
D4	6.13E-02	6.13E-02	5.48E-01	5.99E-01	3.47E+00	9.58E-01	7.71E-01	9.76E-01	8.43E-01	2.40E+00	6.11E-01
D12	2.22E-03	2.22E-03	1.03E+00	1.15E+00	6.96E+00	2.04E+00	1.81E+00	1.59E+00	1.66E+00	4.00E+00	1.46E+00
D19	3.33E-03	4.12E-03	1.33E+00	1.72E+00	2.66E+02	2.49E+00	2.29E+00	1.77E+00	1.90E+00	4.39E+00	1.74E+00
D4N	5.93E-02	5.93E-02	5.89E-01	5.90E-01	8.24E+00	9.53E-01	7.98E-01	9.91E-01	8.34E-01	2.34E+00	6.28E-01
D12N	2.11E-03	2.14E-03	1.03E+00	1.17E+00	7.15E+00	2.15E+00	1.85E+00	1.59E+00	1.64E+00	3.85E+00	1.46E+00
D19N	3.42E-03	3.99E-03	1.31E+00	1.78E+00	2.69E+02	2.46E+00	2.29E+00	1.74E+00	1.85E+00	4.34E+00	1.80E+00
Average fitness											
D4	6.13E-02	6.13E-02	6.28E-01	7.70E-01	3.82E+00	1.10E+00	8.97E-01	1.07E+00	9.29E-01	2.56E+00	6.86E-01
D12	2.34E-03	2.39E-03	1.12E+00	1.46E+00	7.16E+00	2.32E+00	2.09E+00	1.69E+00	1.77E+00	4.13E+00	1.61E+00
D19	3.66E-03	4.84E-03	1.42E+00	1.83E+00	2.70E+02	2.73E+00	2.45E+00	1.85E+00	2.07E+00	4.56E+00	1.91E+00
D4N	5.93E-02	5.93E-02	6.35E-01	7.75E-01	8.89E+00	1.11E+00	9.07E-01	1.08E+00	9.26E-01	2.54E+00	6.97E-01
D12N	2.23E-03	2.28E-03	1.10E+00	1.46E+00	7.39E+00	2.32E+00	2.08E+00	1.67E+00	1.78E+00	4.14E+00	1.60E+00
D19N	3.77E-03	4.79E-03	1.46E+00	1.84E+00	2.73E+02	2.66E+00	2.45E+00	1.84E+00	2.03E+00	4.55E+00	1.93E+00

Bold values represent the best results obtained

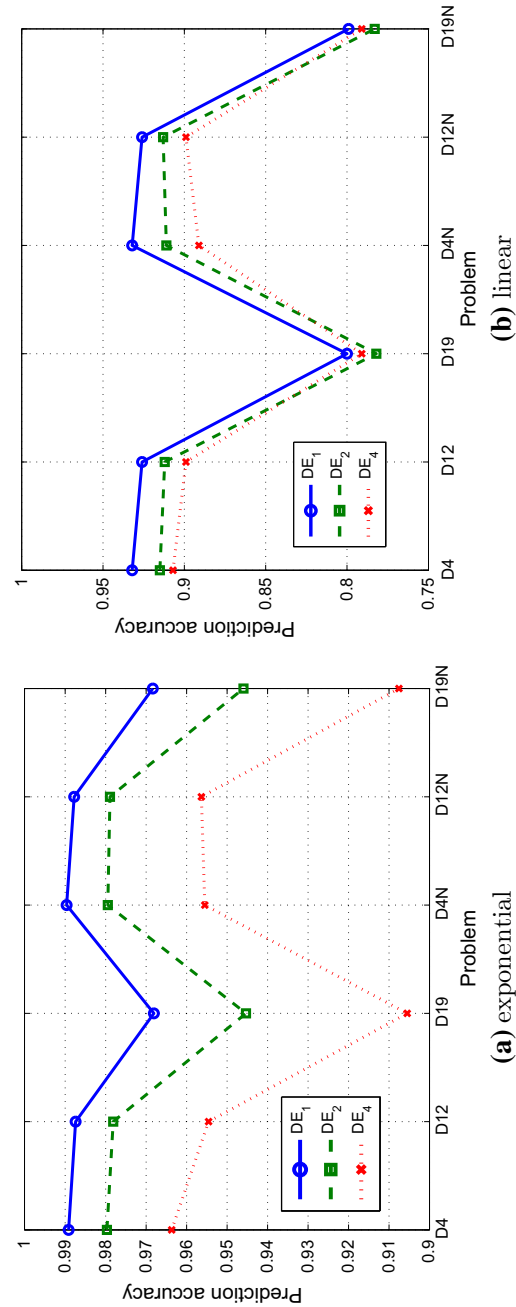


Fig. 8 Prediction accuracy of DE₁, DE₂ and DE₄ using exponential and linear functions

References

1. Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans Evol Comput* 10(6):646–657
2. Caraffini F, Neri F, Cheng J, Zhang G, Picinali L, Iacca G, Mininno E (2013) Super-fit multicriteria adaptive differential evolution. In: 2013 IEEE congress on evolutionary computation (CEC), pp 1678–1685. doi:[10.1109/CEC.2013.6557763](https://doi.org/10.1109/CEC.2013.6557763)
3. Choi TJ, Ahn CW (2015) An adaptive cauchy differential evolution algorithm with population size reduction and modified multiple mutation strategies. In: Proceedings of the 18th Asia Pacific symposium on intelligent and evolutionary systems, vol 2. Springer, pp 13–26
4. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans Evol Comput* 6(2):182–197
5. Eiben AE, Smith JE (2003) Introduction to evolutionary computing. In: Natural computing series. Springer, Berlin
6. Elsayed SM, Sarker RA, Essam DL (2011) Multi-operator based evolutionary algorithms for solving constrained optimization problems. *Comput Oper Res* 38(12):1877–1896
7. Elsayed SM, Sarker RA, Essam DL (2013a) An improved self-adaptive differential evolution algorithm for optimization problems. *Ind Inform IEEE Trans* 9(1):89–99. doi:[10.1109/iii.2012.2198658](https://doi.org/10.1109/iii.2012.2198658)
8. Elsayed SM, Sarker RA, Essam DL (2013b) Self-adaptive differential evolution incorporating a heuristic mixing of operators. *Comput Optim Appl* 54(3):771–790
9. Elsayed SM, Sarker RA, Essam DL, Hamza NM (2014) Testing united multi-operator evolutionary algorithms on the cec2014 real-parameter numerical optimization. In: IEEE congress on evolutionary computation, pp 1650–1657
10. Goh SK, Abbass HA, Tan KC (2015a) Optimization of big data 2015 competition. <http://www.husseinabbass.net/BigOpt.html>. Accessed Jan 2015
11. Goh SK, Abbass HA, Tan KC, Al-Mamun A (2015b) Decompositional independent component analysis using multi-objective optimization. *Soft Comput*. doi:[10.1007/s00500-015-1587-7](https://doi.org/10.1007/s00500-015-1587-7)
12. Guo S, Yang C, Hsu P, Tsai J (2014) Improving differential evolution with successful-parent-selecting framework. *IEEE Trans Evol Comput* PP(99):1. doi:[10.1109/TEVC.2014.2375933](https://doi.org/10.1109/TEVC.2014.2375933)
13. Guo SM, Yang CC (2015) Enhancing differential evolution utilizing eigenvector-based crossover operator. *IEEE Trans Evol Comput* 19(1):31–49. doi:[10.1109/TEVC.2013.2297160](https://doi.org/10.1109/TEVC.2013.2297160)
14. Madden S (2012) From databases to big data. *IEEE Internet Comput* 16(3):4–6
15. Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl Soft Comput* 11(2):1679–1696
16. Manyika J, Chui M, Brown B, Bughin J, Dobbs R, Roxburgh C, Byers AH (2011) Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute. http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/big_data_the_next_frontier_for_innovation. Accessed Mar 2014
17. Mehrotra S (1992) On the implementation of a primal-dual interior point method. *SIAM J Optim* 2(4):575–601
18. Omidvar M, Li X, Mei Y, Yao X (2014) Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Trans Evol Comput* 18(3):378–393. doi:[10.1109/TEVC.2013.2281543](https://doi.org/10.1109/TEVC.2013.2281543)
19. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans Evol Comput* 13(2):398–417
20. Qingfu Z, Hui L (2007) Moea/d: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput* 11(6):712–731
21. Qingfu Z, Wudong L, Hui L (2009) The performance of a new version of moea/d on cec09 unconstrained mop test instances. In: IEEE congress on evolutionary computation, pp 203–208
22. Qiu X, Xu J, Tan K, Abbass H (2015) Adaptive cross-generation differential evolution operators for multi-objective optimization. *Evol Comput IEEE Trans* PP(99):1. doi:[10.1109/TEVC.2015.2433672](https://doi.org/10.1109/TEVC.2015.2433672)
23. Sagiroglu S, Sinanc D (2013) Big data: a review. In: 2013 international conference on collaboration technologies and systems (CTS). IEEE, pp 42–47
24. Sarker R, Kamruzzaman J, Newton C (2003) Evolutionary optimization (evopt): a brief review and analysis. *Int J Comput Intell Appl* 3(4):311–330
25. Slavakis K, Giannakis G, Mateos G (2014) Modeling and optimization for big data analytics: (statistical) learning tools for our era of data deluge. *IEEE Signal Process Mag* 31(5):18–31
26. Storn R (1996) On the usage of differential evolution for function optimization. In: Biennial conference of the North American Fuzzy Information Processing Society (NAFIPS), pp 519–523
27. Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report
28. Tanabe R, Fukunaga A (2013a) Evaluating the performance of shade on cec 2013 benchmark problems. In: IEEE congress on evolutionary computation, pp 1952–1959. doi:[10.1109/CEC.2013.6557798](https://doi.org/10.1109/CEC.2013.6557798)
29. Tanabe R, Fukunaga A (2013b) Success-history based parameter adaptation for differential evolution. In: IEEE congress on evolutionary computation, pp 71–78. doi:[10.1109/CEC.2013.6557555](https://doi.org/10.1109/CEC.2013.6557555)
30. Tang L, Dong Y, Liu J (2015) Differential evolution with an individual-dependent mechanism. *IEEE Trans Evol Comput* 19(4):560–574. doi:[10.1109/TEVC.2014.2360890](https://doi.org/10.1109/TEVC.2014.2360890)
31. Tvrdík J, Polakova R (2010) Competitive differential evolution for constrained problems. In: IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8
32. Tvrdík J, Polakova R (2013) Competitive differential evolution applied to cec 2013 problems. In: IEEE congress on evolutionary computation (CEC). IEEE, pp 1651–1657
33. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans Evol Comput* 15(1):55–66
34. Yang Z, Tang K, Yao X (2011) Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Comput* 15(11):2141–2155
35. 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. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp 154–161
36. Zhang J, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. *IEEE Trans Evol Comput* 13(5):945–958
37. Zhang Y, Zhou M, Jiang Z, Liu J (2015) A multi-agent genetic algorithm for big optimization problems. In: IEEE congress on evolutionary computation, pp 703–707. doi:[10.1109/CEC.2015.7256959](https://doi.org/10.1109/CEC.2015.7256959)
38. Zhi-Hua Z, Chawla NV, Yaochu J, Williams GJ (2014) Big data opportunities and challenges: discussions from data analytics perspectives [discussion forum]. *IEEE Comput Intell Mag* 9(4):62–74