Memetic Comp. (2012) 4:209–229
DOI 10.1007/s12293-012-0089-8

REGULAR RESEARCH PAPER

Cognitive learning in differential evolution and its application to model order reduction problem for single-input single-output systems

Jagdish Chand Bansal · Harish Sharma

Received: 8 July 2011 / Accepted: 18 July 2012 / Published online: 4 August 2012

Abstract Differential evolution (DE) is a well known and simple population based probabilistic approach for global optimization over continuous spaces. It has reportedly outperformed a few evolutionary algorithms and other search heuristics like the particle swarm optimization when tested over both benchmark and real world problems. DE, like other probabilistic optimization algorithms, has inherent drawback of premature convergence and stagnation. Therefore, in order to find a trade-off between exploration and exploitation capability of DE algorithm, a new parameter namely, cognitive learning factor (CLF) is introduced in the mutation process. Cognitive learning is a powerful mechanism that adjust the current position of individuals by the means of some specified knowledge (previous experience of individuals). The proposed strategy is named as cognitive learning in differential evolution ($CLDE$). To prove the efficiency of various approaches of CLF in DE, $CLDE$ is tested over 25 benchmark problems. Further, to establish the wide applicability of CLF, $CLDE$ is applied to two advanced DE variants. $CLDE$ is also applied to solve a well known electrical engineering problem called model order reduction problem for single input single output systems.

Keywords Optimization · Differential evolution · Cognitive learning factor · Model order reduction · Integral square error

J. C. Bansal (✉) · H. Sharma
ABV-Indian Institute of Information Technology and Management, Gwalior, India
e-mail: jcbansal@gmail.com

H. Sharma
e-mail: harish0107@rediffmail.com

1 Introduction

Differential evolution (DE) scheme is a relatively simple, fast and population based stochastic search technique, proposed by Storn and Price [45]. DE falls under the category of evolutionary algorithms (EAs). But in some sense it differs significantly from EAs, e.g. trial vector generation process (explained in Sect. 2) uses the information of distance and direction from current population to generate a new trial vector. Furthermore, in EAs, crossover is applied first to generate a trial vector, which is then used within the mutation operation to produce one offspring while, in DE, mutation is applied first and then crossover.

Researchers are continuously working to improve the performance of DE. Some of the recently developed versions of DE with appropriate applications can be found in [4]. Experiments over several numerical benchmarks [55] show that DE performs better than the genetic algorithm (GA) [20] or the particle swarm optimization (PSO) [23]. DE has successfully been applied to various areas of science and technology, such as chemical engineering [28], signal processing [9], mechanical engineering design [47], machine intelligence, and pattern recognition [40]. Recently, machine intelligence and cybernetics are the most well-liked field in which DE algorithm has become a popular strategy.

There are two fundamental processes which drive the evolution of a DE population: the variation process, which enables exploring different areas of the search space, and the selection process, which ensures the exploitation of the previous experience. However, it has been shown that DE may occasionally stop proceeding toward the global optimum even though the population has not converged to a local optimum [26]. Therefore, to maintain the proper balance between exploration and exploitation behavior of DE, a new control parameter called cognitive learning factor (CLF) is

introduced in DE and DE with CLF is named as Cognitive Learning in differential evolution ($CLDE$). In terminology of social science, Cognitive Learning is about enabling people to learn by using their reason, intuition and perception. This technique is often used to change people's behavior. The same phenomenon is also applied in $CLDE$. In $CLDE$, a weight factor (CLF) is associated with the individual's experience in the mutation operation. By varying this weight, the exploration and exploitation capabilities of DE may be modified.

Furthermore, to show the efficiency of the proposed strategy, model order reduction (MOR) problem for single input single output systems (SISO), is also solved by the $CLDE$ algorithm. MOR problem requires the minimization of an error function in order to get the reduced order model of higher order model. The error function is a function of integral square error (ISE) and the impulse response energy (IRE) of the system. The ISE is taken between the transient responses of original higher order model and the reduced low order model pertaining to a unit step input. The obtained results are compared with existing conventional methods and the results available in the literature. Results reported are encouraging and show that this technique is comparable in quality with existing methods.

The rest of the paper is organized as follows: Sect. 2 describes a brief overview of basic differential evolution algorithm. In Sect. 3, some basic improvements on Differential evolution algorithm are briefly reviewed. Cognitive learning differential evolution algorithm ($CLDE$) is proposed in Sect. 4. In Sect. 5, cognitive learning factor concept is applied to DE and to some recent variants of DE. Then a comparative study has been carried out. Application of $CLDE$ to model order reduction (MOR) problem for single input single output (SISO) systems is shown in Sect. 6. Finally, in Sect. 7, paper is concluded.

2 Brief overview of differential evolution algorithm

DE has several strategies based on three criteria [44]:

- Methods of selecting the target vector,
- Number of difference vectors used and
- The types of crossover.

In this paper $DE/rand/1/bin$ scheme is used where DE stands for differential evolution, 'rand' specifies that the target vector is selected randomly, '1' is for number of differential vectors and 'bin' notation is for *binomial* crossover. The popularity of differential evolution is due to its applicability to a wider class of problems and ease of implementation. Differential evolution consists of the properties of both evolutionary algorithms and swarm intelligence. The detailed description of DE is as follows:

Like other population based search algorithms, in DE a population of potential solutions (individuals) searches the solution. In a D-dimensional search space, an individual is represented by a D-dimensional vector $(x_{i1}, x_{i2}, \ldots, x_{iD})$, $i = 1, 2, \ldots, NP$ where NP is the population size (number of individuals).

In DE, there are three operators: mutation, crossover and selection. Initially, a population is generated randomly with uniform distribution then the mutation, crossover and selection operators are applied to generate a new population. Trial vector generation is a crucial step in DE process. The two operators mutation and crossover are used to generate the trial vectors. The selection operator is used to select the best trial vector for the next generation. DE operators are briefly explained in following subsections.

2.1 Mutation

A trial vector is generated by the DE mutation operator for each individual of the current population. For generating the trial vector, a target vector is mutated with a weighted difference. An offspring is produced in the crossover operation using the newly generated trial vector. If G is the index for generation counter, the mutation operator for generating a trial vector $u_i(G)$ from the parent vector $x_i(G)$ is defined as follows:

- Select a target vector, $x_{i1}(G)$, from the population, such that $i \neq i1$.
- Again, randomly select two individuals, x_{i2} and x_{i3}, from the population such that $i, i1, i2$ and $i3$, all are different.
- Then the target vector is mutated for calculating the trial vector as follows:

$$u_i(G) = x_{i1}(G) + F(x_{i2}(G) - x_{i3}(G)) \tag{1}$$

where $F \in (0, 1)$ is the mutation scale factor which is used in controlling the amplification of the differential variation [12].

2.2 Crossover

Offspring $x'_i(G)$ is generated using the crossover of parent vector, $x_i(G)$ and the trial vector, $u_i(G)$ as follows:

$$x'_{ij}(G) = \begin{cases} u_{ij}(G), & \text{if } j \in J \\ x_{ij}(G), & \text{otherwise.} \end{cases}$$

where J is the set of cross over points or the points that will go under perturbation, $x_{ij}(G)$ is the jth element of the vector $x_i(G)$.

Different methods may be used to determine the set, J of which binomial crossover and exponential crossover are

the most frequently used [12,52]. In this paper, the DE and its variants are implemented using the binomial crossover. In this crossover, the crossover points are randomly selected from the set of possible crossover points, $\{1, 2, \ldots, D\}$, where D is the problem dimension. Algorithm 1 shows the steps of binomial crossover to generate crossover points [12]. In this algorithm, CR is the probability that the considered crossover point will be included. The larger the value of CR, the more crossover points will be selected. Here,

Algorithm 1 Binomial crossover:

```
J = φ
j* ~ U(1, D);
J ← J ∪ j*;
for each j ∈ 1 ... D do
  if U(0, 1) < CR and j ≠ j* then
    J ← J ∪ j;
  end if
end for
```

J is a set of crossover points, CR is crossover probability, $U(1, D)$ is a uniformly distributed random integer in between 1 and D, $U(0, 1)$ is a uniformly distributed random number between 0 and 1.

2.3 Selection

The Selection operator first selects the individual for the mutation operation to generate the trial vector and then it selects the best, between the parent and the offspring based on their fitness values for the next generation. If the fitness of the parent is better than the offspring then parent is selected otherwise offspring is selected:

$$x_i(G+1) = \begin{cases} x'_i(G), & \text{if } f(x'_i(G)) > f(x_i(G)). \\ x_i(G), & \text{otherwise.} \end{cases}$$

This ensures that the population's average fitness does not deteriorate.

Algorithm 2 shows the pseudo-code for the basic differential evolution strategy [12]. Here, F (scale factor) and CR (crossover probability) are the control parameters of $DE/rand/1/bin$ strategy and play significant role to influence the performance of the DE. P is the population vector.

3 Brief review on basic improvements in differential evolution

In order to get rid of the drawbacks of the basic DE, researchers have improved DE in many ways. The potentials where DE can be improved may be broadly classified into three categories:

Algorithm 2 Differential evolution algorithm:

```
Initialize the control parameters, F and CR;
Create and initialize the population, P(0), of NP individuals;
while stopping condition(s) not true do
  for each individual, x_i(G) ∈ P(G) do
    Evaluate the fitness, f(x_i(G));
    Create the trial vector, u_i(G) by applying the mutation operator;
    Create an offspring, x'_i(G), by applying the crossover operator;
    if f(x'_i(G)) is better than f(x_i(G)) then
      Add x'_i(G) to P(G + 1);
    else
      Add x_i(G) to P(G + 1);
    end if
  end for
end while
Return the individual with the best fitness as the solution;
```

- Fine tuning of DE control parameters NP, F and CR.
- Hybridization of DE with other population based probabilistic or deterministic algorithms.
- Introducing new control parameters.

This paper concentrates on the third area of the DE research, i.e. the paper introduces a new control parameter, namely, cognitive learning factor in DE process. Rest of this section briefly reviews introduction of new control parameters in DE process.

Storn and Price [45] have observed that the value of F should be in the range of [0.5, 1] and 0.5 is a good initial choice. The value of NP should be in the range of $[5D, 10D]$, where, D is the dimension of the problem.

Fuzzy adaptive differential evolution ($FADE$) is introduced by Liu and Lampinen [27]. It is based on the fuzzy logic controllers, whose inputs incorporate the relative function values and individuals of successive generations to adapt the parameters for the mutation and crossover operation. They showed by the experimental results over a set of benchmark functions that the FADE algorithm performance is better than the conventional DE algorithm.

Gamperle et al. [14] determined different parameter values for DE specially for the Sphere, Rastrigin' and Rosenbrock' functions. They showed that the global optimum searching capability and the convergence speed are very sensitive for the values of the control parameters NP, F, and CR. They specified that the population size $NP \in [3D, 8D]$, with the scaling factor $F = 0.6$ and the crossover rate CR in [0.3, 0.9] are the good choice for the parameter setting.

Zaharie proposed a parameter adaptation strategy for DE (ADE) [59] which is based on controlling the population diversity. In ADE, multi-population approach is also implemented. Furthermore, Zaharie and Petcu [60] introduced an adaptive Pareto DE algorithm, based on the same line of thinking, for multi-objective optimization.

Abbass [1] proposed a self-adaptive strategy for crossover rate CR to solve multi-objective optimization problems.

In Abbass strategy, CR value is encoded into each individual, simultaneously evolved with other search variables. There was a different scale factor F, uniformly distributed in [0, 1], for each variable.

Furthermore, Qin et al. introduced a self-adaptive DE ($SaDE$) [46] algorithm, in which all the control parameters that are used in the trial vector generation strategies and selection process are steadily self-adapted by learning from their previous experiences.

Omran et al. [39] introduced a self-adaptive scaling factor F. They generated the value of CR for each individual from a normal distribution N (0.5, 0.15). This approach (called SDE) was tested on four benchmark functions and verified to be performed better than other versions of DE.

Besides, setting the control parameters (F and CR), some researchers also tuned the population size (NP) for improving the performance. Teo introduced a variant of DE which is based on the idea of self adapting populations (DESAP) [54].

Noman and Iba [38] introduced a crossover-based local search method for DE called the fittest individual refinement (FIR). An exploration capability of DE is hastened by the FIR scheme as it enhances DE's search capability in the neighborhood for the best solution in successive generations.

Furthermore, Yan et al. [58] proposed a new variant of DE called simulated annealing differential evolution ($SADE$). In SADE algorithm, each individual contains a set of F values instead of single value within the range [0.1, 1], control parameters F and CR are encoded into the individual and their values are changed, based on the two new probability factors τ_1 and τ_2. F is reinitialized with the probability τ_1 by a random value otherwise it remains unchanged. The crossover probability CR also reinitialized with probability τ_2 and within the range [0, 1]. CR is assigned to each individual but in an identical fashion. CR changes its value with probability τ_2 with a random value otherwise it remains unchanged for the next generation.

Neri and Tirronen [37] proposed a self-adaptive strategy called scale factor local search differential evolution ($SFLSDE$) strategy. $SFLSDE$ is a self-adaptive scheme with the two local search algorithms. These local search algorithms are used for detecting the value of scale factor F corresponding to an offspring with a better performance. Therefore, the local search algorithms support in the global search (exploration process) and in generating offspring with high performance.

Swagatam Das et al. [8] proposed a new variant of differential evolution algorithm called differential evolution using a neighborhood-based mutation operator ($DEGL$). The proposed scheme balances the exploration and exploitation abilities of DE. $DEGL$ introduces four new control parameters: α, β, w, and the neighborhood radius k. In $DEGL$, w is the most important parameter as it controls the balance between the exploration and exploitation capabilities. It is shown in the following expression.

$$u = w \times Global + (1 - w) \times Local$$

where u is a trial vector and $w \in [0, 1]$. Small values of w favor the local neighborhood component, thereby resulting in better exploration. On the other hand, large values favor the global variant component, encouraging exploitation. Therefore, values of w near about 0.5 result the most balanced $DEGL$ version.

4 Cognitive learning in differential evolution

4.1 A few drawbacks of DE

The inherent drawback with most of the population based stochastic algorithms is premature convergence. DE is not an exception. Any population based algorithm is regarded as an efficient algorithm if it is fast in convergence and able to explore the maximum area of the search space. In other words, if a population based algorithm is capable of balancing between exploration and exploitation of the search space, then the algorithm is regarded as an efficient algorithm. From this point of view, the basic DE is not an efficient algorithm [32]. Also some studies proved that stagnation is another inherent drawback with DE, i.e. DE sometimes stops proceeding toward the global optima even though the population has not converged to local optima or any other point [26]. Mezura-Montes et al. [32] compared the different variants of DE for global optimization and found that DE shows a poor performance and remains inefficient in exploring the search space, especially for multimodal functions. Price et al. [45] also drawn the same conclusions. The problem of premature convergence and stagnation is a matter of serious consideration for designing a comparatively efficient[1] DE algorithm.

4.2 Motivation for cognitive learning factor

This section proposes a new parameter namely cognitive learning in differential evolution algorithm ($CLDE$).

4.2.1 Cognitive learning factor in DE

Exploration of the whole search space and exploitation of the near optimal solution region may be balanced by maintaining the diversity in early and later iterations of any randomized search algorithm. Mutation equation (1) in DE may be seen in the following way:

[1] As it is not possible to design a fully efficient population based stochastic algorithm.

$$u_i(G) = A \times x_{i1}(G) + B \times (x_{i2}(G) - x_{i3}(G))$$

i.e. the trial vector $u_i(G)$ is the weighted sum of target vector $x_i(G)$ and the difference $(x_{i2}(G) - x_{i3}(G))$ of two random vectors. Here, A is the weight to target vector and B is the weight to the difference of random vectors. In basic DE, A is set to be 1, while B is the scaling factor F. Studies have been carried out with varying scaling factor F for better exploration and exploitation mechanism [3]. To the best of authors' knowledge no study has been carried out to set the weight A in DE mutation equation. In this paper, the experiments are performed over benchmark problems to find an optimal strategy to set the weight A named as cognitive learning factor (CLF) and denoted by 'C' (for this study). CLF is the weight to individual's current position or in other words this is the weight to self confidence and therefore, it is named so.

In this way, the modified mutation operation of DE becomes:

$$u_i(G) = C \times x_{i1}(G) + F(x_{i2}(G) - x_{i3}(G)). \quad (2)$$

Symbols have their usual meaning. Now different strategies to set CLF 'C' may produce different results. It is expected that a random CLF ($\in$ [0, 1]) will give lower weight to personal experience than the weight in basic DE and therefore the algorithm will produce a higher diversity and slow convergence. For linearly decreasing (1–0.1) CLF, diversity will increase with iterations. In the case of linearly increasing (0.1–1) CLF, diversity will be relatively high in early iterations and will keep on reducing in successive iterations while convergence rate is expected to be low in early iterations and high in later iterations. Theoretically, linearly increasing CLF should improve the results. Therefore, experiments are performed over scalable test problems of optimization with all three strategies of setting CLF: constant; linearly decreasing; linearly increasing.

The cognitive learning factor algorithm ($CLDE$) is similar to the basic DE algorithm except the mutation operation. The Pseudo-code in Algorithm 3 presents the steps of the $CLDE$ algorithm. $CLDE$ is a simple algorithm which, despite its simplicity, can be a very efficient possibility for optimization of various real world optimization problems.

4.3 Control Parameters in $CLDE$

As stated by Storn et al. [45,53], DE is very sensitive to the choice of F and CR. Some settings of control parameters are suggested by Storn et al. [45,53]:

- $F \in [0.5, 1]$.
- $CR \in [0.8, 1]$.
- $NP = [5D, 10D]$, where D is the number of decision variables in the problem.

Algorithm 3 Cognitive learning in differential evolution ($CLDE$)

```
Initialize the control parameters, F, CR and C(0);
Create and initialize the population, P(0), of NP individuals;
while stopping condition(s) not true do
  for each individual, x_i(G) ∈ P(G) do
    Evaluate the fitness, f(x_i(G));
    Create the trial vector, u_i(G) by applying the scale factor (mutation operator) F and cognitive learning factor C(G) as follows;
        u_i(G) = C(G) × x_i1(G) + F(x_i2(G) − x_i3(G));
    Create an offspring, x'_i(G), by applying the crossover operator;
    if f(x'_i(G)) is better than f(x_i(G)) then
      Add x'_i(G) to P(G + 1);
    else
      Add x_i(G) to P(G + 1);
    end if
  end for
end while
Return the individual with the best fitness as the solution;
```

$CLDE$ introduces one new parameter: C called the cognitive learning factor, therefore, there are four controlling parameters (F, CR, NP and C) in $CLDE$. Cognitive learning factor C is the most important parameter in $CLDE$ as it controls the balance between the exploration and exploitation capabilities of the algorithm. Three different strategies have been considered for the selection and adaptation of C:

1. Random cognitive learning factor ($RCLF$): in this strategy the C is a uniformly distributed random number in [0, 1] for each iteration. Random choice of C may introduce higher diversity and lower convergence.
2. Linearly decreasing cognitive learning factor ($LDCLF$): C is linearly decreased from 1 to 0.1. Initially C is set to 1 and gradually decreased generation by generation up to 0.1 as follows:

$$C(G+1) = C(G) - \frac{(1-0.1)}{N}$$

   where $C(0) = 1$, N is the total number of generation and G is the current generation.
3. Linearly increasing cognitive learning factor ($LICLF$): C is linearly increased from 0.1 to 1. Initially C is set to 0.1 and the gradually increased generation by generation up to 1 as follows:

$$C(G+1) = C(G) + \frac{(1-0.1)}{N}$$

   where $C(0) = 0.1$, N is the total number of generation and G is the current generation.

Based on different types of CLF, three variants of $CLDE$ are designed namely, random cognitive learning in differential

evolution ($RCLDE$), linearly decreasing cognitive learning in differential evolution ($LDCLDE$) and linearly increasing cognitive learning in differential evolution ($LICLDE$). In the next section $CLDE$ with different strategies of setting CLF is tested with 25 benchmark problems. Experiments have also been carried out to test the efficiency of CLF over two advanced variants of DE, namely simulated annealing differential evolution ($SADE$) [58] and scale factor local search differential evolution ($SFLSDE$) [37]. Then the proposed algorithm is applied to a problem of control theory in Sect. 6.

5 Experimental results and discussion

5.1 Test problems under consideration

In order to see the effect of cognitive learning factor on DE, 25 test problems of optimization are selected (listed in Tables 1 and 2). 16 test problems are given in Table 1 which are scalable in nature and has the solution at origin. Number of decision variables for these problems is set to be 30. Table 2 contains nine problems which are relatively complex optimization problems. The solution of these problems is neither at origin, axes or diagonal, i.e. the problems are unbiased. Number of decision variables for this set of problems is mentioned in the Table 2. All problems are of continuous variables and have different degree of complexity and multimodality.

5.2 Experimental setting for $CLDE$

To test DE or DE variants over test problems the following experimental setting is adopted:

- The crossover probability $CR = 0.33$ [14].
- The scale factor which controls the implication of the differential variation $F = 0.5$ [44].
- Population size $NP = 100$.
- The algorithm terminates when either maximum number of iterations are reached or the error is $\leq \epsilon$. Here the maximum number of iterations are set 1,000. The value of ϵ for problems of Table 1 is 0.01 while for Table 2 problems it is mentioned in the corresponding table for each problem separately.
- The number of simulations = 100.

5.3 Comparison among DE with variants of $CLDE$

Three strategies ($RCLDE$, $LDCLDE$, $LICLDE$) of setting CLF in DE (explained in Sect. 4.3) are implemented. Numerical results with experimental setting of Sect. 5.2 are tabulated in Table 3. In Table 3, success rate (SR) (a simulation is said to be successful if the objective function value is $\leq \epsilon$, refer Sect. 5.2, in maximum 1,000 generations), mean error (ME), average function evaluations for successful runs (AFE) and standard deviation (SD) is reported. In Table 3, if any algorithm shows no success

Table 1 Test problems

Test problem	Objective function	Search space
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-5.12, 5.12]$
De Jong f4	$f_2(x) = \sum_{i=1}^{n} i.(x_i)^4$	$[-5.12, 5.12]$
Griewank	$f_3(x) = 1 + \frac{1}{4{,}000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$	$[-600, 600]$
Rosenbrock	$f_4(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-30, 30]$
Rastrigin	$f_5(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]$
Ackley	$f_6(x) = -20\exp(-0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi.x_i)) + 20 + e$	$[-30, 30]$
DropWave	$f_7(x) = -\frac{1+\cos(\sqrt[12]{\sum_{i=1}^{n} {x_i}^2})}{\frac{1}{2}\sum_{i=1}^{n} {x_i}^2 + 2}$	$[-5.12, 5.12]$
Alpine	$f_8(x) = \sum_{i=1}^{n} \lvert x_i \sin x_i + 0.1x_i \rvert$	$[-10, 10]$
Michalewicz	$f_9(x) = -\sum_{i=1}^{n} \sin x_i (\sin(\frac{i.{x_i}^2}{\pi}))$	$[0\,\pi]$
Cosine mixture	$f_{10}(x) = \sum_{i=1}^{n} {x_i}^2 - 0.1(\sum_{i=1}^{n} \cos 5.\pi.x_i) + 0.1n$	$[-1, 1]$
Exponential	$f_{11}(x) = -(\exp(-0.5\sum_{i=1}^{n} {x_i}^2)) + 1$	$[-1, 1]$
Zakharov	$f_{12}(x) = \sum_{i=1}^{n} {x_i}^2 + (\sum_{i=1}^{n} \frac{ix_i}{2})^2 + (\sum_{i=1}^{n} \frac{ix_1}{2})^4$	$[-5.12, 5.12]$
Cigar	$f_{13}(x) = {x_0}^2 + 100{,}000 \sum_{i=1}^{n} {x_i}^2$	$[-10, 10]$
brown3	$f_{14}(x) = \sum_{i=1}^{n-1} ({x_i}^{{x_{i+1}}^2+1} + {x_{i+1}}^{{x_i}^2+1})$	$[-1, 4]$
Schewel	$f_{15}(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]$
Sum of powers	$f_{16}(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	$[-1, 1]$

Table 2 Test problems

Test problem	Objective function	Search range	Optimum value	D	Acceptable error (ϵ)
Shifted Rosenbrock	$f_{17}(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$, $z = x - o + 1$, $x = [x_1, x_2, \ldots, x_D]$, $o = [o_1, o_2, \ldots, o_D]$	$[-100, 100]$	$f(o) = f_{bias} = 390$	10	1.0E−01
Shifted sphere	$f_{18}(x) = \sum_{i=1}^{D} z_i^2 + f_{bias}$, $z = x - o$, $x = [x_1, x_2, \ldots, x_D]$, $o = [o_1, o_2, \ldots, o_D]$	$[-100, 100]$	$f(o) = f_{bias} = -450$	10	1.0E−05
Shifted Rastrigin	$f_{19}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{bias}$ $z = (x - o)$, $x = (x_1, x_2, \ldots, x_D)$, $o = (o_1, o_2, \ldots, o_D)$	$[-5, 5]$ $[-5, 5]$	$f(o) = f_{bias} = -330$ $f(o) = f_{bias} = -330$	10 10	1.0E−02 1.0E−02
Shifted Griewank	$f_{20}(x) = \sum_{i=1}^{D} \frac{z_i^2}{4{,}000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + f_{bias}$, $z = (x - o)$, $x = [x_1, x_2, \ldots, x_D]$, $o = [o_1, o_2, \ldots, o_D]$	$[-600, 600]$	$f(o) = f_{bias} = -180$	10	1.0E−05
Shifted Ackley	$f_{21}(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D} \times \cos(2\pi z_i)) + 20 + e + f_{bias}$, $z = (x - o)$, $x = (x_1, x_2, \ldots, x_D)$, $o = (o_1, o_2, \ldots, o_D)$	$[-32, 32]$	$f(o) = f_{bias} = -140$	10	1.0E−05
Kowalik function	$f_{22}(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	$[-5, 5]$	$f(0.192833, 0.190836, 0.123117, 0.135766) = 0.000307486$	4	1.0E−05
Six-hump camel back	$f_{23}(x) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$[-5, 5]$	$f(-0.0898, 0.7126) = -1.0316$	2	1.0E−05
Sinusoidal problem	$f_{24}(x) = -[A\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n} sin(B(x_i - z))]$, $A = 2.5, B = 5, z = 30$	$[0, 180]$	$f(90 + z) = -(A + 1)$	10	1.00E−02
Moved axis parallel hyper-ellipsoid	$f_{25}(x) = \sum_{i=1}^{D} 5i \times x_i^2$	$[-5.12, 5.12]$	$f(x) = 0; x(i) = 5 * i, i = 1 : D$	30	1.0E−15

or in other words if SR of an algorithm is 0, then for that algorithm entry in the column of AFE is taken to be 10,0000 which is calculated by multiplying NP with Maximum number of generations. This is done to make the comparison fair in terms of function evaluations. The discussion about $SADE$, $SACLDE$, $SFLSDE$, $SFLSCLDE$ may be found in Sect. 5.4. Table 3 shows that most of the time inclusion of CLF in DE improves the reliability, efficiency and accuracy. Except for Rosenbrock problem $CLDE$ versions improve the results over DE. Some more intensive statistical analysis based on t test, performance index and boxplots has been carried out for results of basic DE and variants of $CLDE$($RCLDE$, $LDCLDE$ and $LICLDE$).

5.3.1 Statistical analysis

In order to extract the best strategy of setting CLF in DE, a comparative analysis is done for DE, $LICLDE$, $RCLDE$ and $LDCLDE$. Statistical comparison is carried out using t test, boxplots and performance index [11].

The t test is quite popular among researchers in the field of evolutionary computing. In this paper Student's t test is applied according to the description given in [7] for a confidence level of 0.95. Table 4 shows the results of the t test for the null hypothesis that there is no difference in the mean number of function evaluations of 100 runs using DE and variants of CLDE. Note that here '+' indicates the significant difference (or the null hypothesis is rejected) at a 0.05 level of significance, '−' implies that there is no significant difference while '=' indicates that the comparison is not possible. Table 4 is divided into two parts. In the first part DE is compared with the $CLDE$ variants. It is observed from this part of the table that significant differences are observed in 45 comparisons out of 75 comparisons. Therefore, it can be concluded that the results of variants of $CLDE$ are significantly different from the basic DE algorithm. Furthermore, in the second part of the table, $LICLDE$ is compared with other two variants of $CLDE$. Here, 39 comparisons are significantly different out of 50 comparisons. Therefore, it can be stated that the results of $LICLDE$ are significantly different from the $RCLDE$ and $LDCLDE$.

For the purpose of comparison in terms of performance, boxplot analysis is carried out for all the considered algorithms. The empirical distribution of data is efficiently represented graphically by the boxplot analysis tool [57]. Analysis of univariate expressions, where the variability of measurements may be affected by many parameters, is effectively done by the boxplot tool. The degree of dispersion and skewness in the data are easily analyzed by measuring the spacings between the different parts of the box. The Boxplots for DE and $CLDE$ variants are shown in Fig. 1. It is clear from this

Table 3 Comparison of the results of test problems

Test problem	Algorithm	SR	ME	AFE	SD
f_1	DE	51	0.59×10^{-2}	28,111.77	0.11×10^{-2}
	RCLDE	100	0.45×10^{-2}	7,516	0.14×10^{-2}
	LICLDE	100	0.36×10^{-2}	4,858	0.15×10^{-2}
	LDCLDE	100	0.55×10^{-2}	20,681	0.14×10^{-2}
	SADE	100	0.90×10^{-2}	13,044	0.78×10^{-3}
	SACLDE	100	0.70×10^{-2}	5,027	0.23×10^{-2}
	SFLSDE	100	0.91×10^{-2}	30,078.68	0.86×10^{-3}
	SFLSCLDE	100	0.70×10^{-2}	2,353.89	0.21×10^{-2}
f_2	DE	97	0.37×10^{-2}	28,936.08	0.18×10^{-2}
	RCLDE	100	0.21×10^{-2}	6,305	0.13×10^{-2}
	LICLDE	100	0.16×10^{-2}	3,962	0.13×10^{-2}
	LDCLDE	100	0.31×10^{-2}	19,950	0.14×10^{-2}
	SADE	100	0.83×10^{-2}	11,164	0.13×10^{-2}
	SACLDE	100	0.56×10^{-2}	4,139	0.28×10^{-2}
	SFLSDE	100	0.84×10^{-2}	29,102.08	0.13×10^{-2}
	SFLSCLDE	100	0.46×10^{-2}	1,517.31	0.29×10^{-2}
f_3	DE	49	0.60×10^{-2}	51,126.53	0.11×10^{-2}
	RCLDE	100	0.41×10^{-2}	12,786	0.12×10^{-2}
	LICLDE	100	0.39×10^{-2}	8,250	0.16×10^{-2}
	LDCLDE	100	0.51×10^{-2}	30,506	0.14×10^{-2}
	SADE	99	0.90×10^{-2}	23,532.32	0.87×10^{-3}
	SACLDE	100	0.74×10^{-2}	8,307	0.19×10^{-2}
	SFLSDE	100	0.92×10^{-2}	55,004.73	0.64×10^{-3}
	SFLSCLDE	100	0.74×10^{-2}	5,390	0.20×10^{-2}
f_4	DE	0	25.96	100,000	1.29
	RCLDE	0	28.10	100,000	0.21
	LICLDE	0	28.42	100,000	0.16
	LDCLDE	0	27.77	100,000	0.27
	SADE	0	18.38	100,000	5.78
	SACLDE	0	16.81	100,000	0.90
	SFLSDE	0	28.02	100,000	15.03
	SFLSCLDE	0	28.26	100,000	0.17
f_5	DE	0	91.01	100,000	8.01
	RCLDE	100	0.42×10^{-2}	38,393	0.12×10^{-2}
	LICLDE	100	0.38×10^{-2}	11,926	0.14×10^{-2}
	LDCLDE	1	34.16	99,300	16.09
	SADE	0	32.69	100,000	3.95
	SACLDE	100	0.73×10^{-2}	9,601	0.19×10^{-2}
	SFLSDE	0	52.30	100,000	5.60
	SFLSCLDE	100	0.70×10^{-2}	5,456	0.20×10^{-2}
f_6	DE	0	0.15×10^{-1}	100,000	0.18×10^{-2}
	RCLDE	100	0.65×10^{-2}	13,006	0.11×10^{-2}
	LICLDE	100	0.59×10^{-2}	8,426	0.14×10^{-2}
	LDCLDE	100	0.69×10^{-2}	31,239	0.93×10^{-3}
	SADE	100	0.94×10^{-2}	23,913	0.48×10^{-3}
	SACLDE	100	0.84×10^{-2}	8,542	0.12×10^{-2}
	SFLSDE	100	0.95×10^{-2}	54,489.09	0.40×10^{-3}
	SFLSCLDE	100	0.83×10^{-2}	5,629.15	0.14×10^{-2}

Table 3 continued

Test problem	Algorithm	SR	ME	AFE	SD
f_7	DE	99	0.10×10^{-5}	263.64	0.3×10^{-5}
	RCLDE	100	0.10×10^{-5}	283	0.40×10^{-5}
	LICLDE	100	0.20×10^{-5}	296	0.50×10^{-5}
	LDCLDE	100	0.1×10^{-5}	263	0.2×10^{-5}
	SADE	100	0.14×10^{-5}	200	0.22×10^{-5}
	SACLDE	100	0.16×10^{-5}	200	0.29×10^{-5}
	SFLSDE	100	0.16×10^{-5}	199.88	0.27×10^{-5}
	SFLSCLDE	100	0.15×10^{-5}	199.94	0.32×10^{-5}
f_8	DE	0	0.21×10^{-1}	100,000	0.14×10^{-2}
	RCLDE	100	0.67×10^{-2}	12,708	0.15×10^{-2}
	LICLDE	100	0.65×10^{-2}	7,312	0.12×10^{-2}
	LDCLDE	100	0.68×10^{-2}	41,345	0.10×10^{-2}
	SADE	100	0.96×10^{-2}	54,745	0.40×10^{-3}
	SACLDE	100	0.85×10^{-2}	7,249	0.12×10^{-2}
	SFLSDE	76	0.99×10^{-2}	90,989.01	0.56×10^{-3}
	SFLSCLDE	100	0.86×10^{-2}	4,247.89	0.13×10^{-2}
f_9	DE	99	0.32×10^{-3}	31,955.55	0.87×10^{-3}
	RCLDE	97	0.35×10^{-3}	27,853.61	0.68×10^{-3}
	LICLDE	100	0.27×10^{-3}	25,852	0.47×10^{-3}
	LDCLDE	97	0.32×10^{-3}	22,319.59	0.68×10^{-3}
	SADE	100	0.51×10^{-2}	420	0.28×10^{-2}
	SACLDE	100	0.51×10^{-2}	442	0.28×10^{-2}
	SFLSDE	100	0.45×10^{-2}	450.85	0.29×10^{-2}
	SFLSCLDE	100	0.40×10^{-2}	439.83	0.27×10^{-2}
f_{10}	DE	47	0.62×10^{-2}	28,344.68	0.14×10^{-2}
	RCLDE	100	0.43×10^{-2}	7,322	0.14×10^{-2}
	LICLDE	100	0.37×10^{-2}	4,646	0.15×10^{-2}
	LDCLDE	100	0.52×10^{-2}	21,000	0.15×10^{-2}
	SADE	100	0.89×10^{-2}	13,410	0.85×10^{-3}
	SACLDE	100	0.73×10^{-2}	4,807	0.21×10^{-2}
	SFLSDE	100	0.91×10^{-2}	29,480.51	0.75×10^{-3}
	SFLSCLDE	100	0.66×10^{-2}	2,013.93	0.21×10^{-2}
f_{11}	DE	48	0.60×10^{-2}	17,077.08	0.11×10^{-2}
	RCLDE	100	0.44×10^{-2}	4,689	0.14×10^{-2}
	LICLDE	100	0.38×10^{-2}	3,043	0.15×10^{-2}
	LDCLDE	100	0.53×10^{-2}	14,049	0.14×10^{-2}
	SADE	100	0.89×10^{-2}	7,471	0.95×10^{-3}
	SACLDE	100	0.70×10^{-2}	2,718	0.19×10^{-2}
	SFLSDE	100	0.90×10^{-2}	17,897.15	0.99×10^{-3}
	SFLSCLDE	100	0.67×10^{-2}	787.77	0.24×10^{-2}
f_{12}	DE	0	58.73	100,000	8.35
	RCLDE	0	0.16	100,000	0.58×10^{-1}
	LICLDE	0	0.26	100,000	0.85×10^{-1}
	LDCLDE	0	0.78	100,000	0.22
	SADE	100	0.96×10^{-2}	47,991	0.39×10^{-3}
	SACLDE	100	0.83×10^{-2}	13,778	0.16×10^{-2}
	SFLSDE	0	21.26	100,000	5.58
	SFLSCLDE	100	0.73×10^{-2}	17,202.78	0.19×10^{-2}

Table 3 continued

Test problem	Algorithm	SR	ME	AFE	SD
f_{13}	DE	47	0.62×10^{-2}	63,378.72	0.12×10^{-2}
	RCLDE	100	0.46×10^{-2}	16,419	0.16×10^{-2}
	LICLDE	100	0.36×10^{-2}	10,812	0.14×10^{-2}
	LDCLDE	100	0.51×10^{-2}	36,365	0.14×10^{-2}
	SADE	100	0.89×10^{-2}	30,924	0.94×10^{-3}
	SACLDE	100	0.75×10^{-2}	10,869	0.19×10^{-2}
	SFLSDE	100	0.91×10^{-2}	69,632.25	0.76×10^{-3}
	SFLSCLDE	100	0.69×10^{-2}	7,684.98	0.20×10^{-2}
f_{14}	DE	48	0.62×10^{-2}	26,625	0.14×10^{-2}
	RCLDE	100	0.45×10^{-2}	7,021	0.15×10^{-2}
	LICLDE	100	0.39×10^{-2}	4,482	0.16×10^{-2}
	LDCLDE	100	0.54×10^{-2}	19,914	0.16×10^{-2}
	SADE	100	0.90×10^{-2}	15,839	0.79×10^{-3}
	SACLDE	100	0.68×10^{-2}	5,146	0.21×10^{-2}
	SFLSDE	100	0.90×10^{-2}	32,814.09	0.81×10^{-3}
	SFLSCLDE	100	0.64×10^{-2}	2,341.84	0.21×10^{-2}
f_{15}	DE	0	0.14×10^{-1}	100,000	0.19×10^{-2}
	RCLDE	100	0.66×10^{-2}	13,832	0.11×10^{-2}
	LICLDE	100	0.62×10^{-2}	8,962	0.11×10^{-2}
	LDCLDE	100	0.68×10^{-2}	31,824	0.82×10^{-3}
	SADE	100	0.94×10^{-2}	26,924	0.51×10^{-3}
	SACLDE	100	0.85×10^{-2}	8,889	0.12×10^{-2}
	SFLSDE	100	0.95×10^{-2}	52,935.07	0.46×10^{-3}
	SFLSCLDE	100	0.84×10^{-2}	6,028.6	0.12×10^{-2}
f_{16}	DE	61	0.61×10^{-2}	18,934.43	0.13×10^{-2}
	RCLDE	100	0.43×10^{-2}	5,158	0.14×10^{-2}
	LICLDE	100	0.42×10^{-2}	3,316	0.17×10^{-2}
	LDCLDE	100	0.54×10^{-2}	15,244	0.13×10^{-2}
	SADE	100	0.90×10^{-2}	8,537	0.75×10^{-3}
	SACLDE	100	0.74×10^{-2}	3,201	0.18×10^{-2}
	SFLSDE	100	0.91×10^{-2}	20,138.46	0.74×10^{-3}
	SFLSCLDE	100	0.66×10^{-2}	1,039.48	0.23×10^{-2}
f_{17}	DE	100	9.50×10^{-2}	62,360	4.96×10^{-3}
	RCLDE	0	1.52	100,000	119.22
	LICLDE	100	9.2×10^{-2}	70,570	6.54×10^{-3}
	LDCLDE	0	9.11	100,000	8.26
	SADE	94	0.25	42,603	0.76
	SACLDE	98	0.25	33,490	0.76
	SFLSDE	95	0.28	58,089	0.85
	SFLSCLDE	95	0.28	58,089	0.84
f_{18}	DE	100	7.85×10^{-6}	22,132	1.64×10^{-6}
	RCLDE	0	7.01	100,000	0.26
	LICLDE	100	8.13×10^{-6}	23,933	1.51×10^{-6}
	LDCLDE	0	6.59	100,000	6.58
	SADE	100	8.02×10^{-6}	7,904	1.64×10^{-6}
	SACLDE	100	8.16×10^{-6}	6,400	1.42×10^{-6}
	SFLSDE	100	8.21×10^{-6}	12,282	1.52×10^{-6}
	SFLSCLDE	100	8.21×10^{-6}	12,282	1.52×10^{-6}

Table 3 continued

Test problem	Algorithm	SR	ME	AFE	SD
f_{19}	DE	0	73.89	100,000	11.46
	RCLDE	0	240.75	100,000	8.04
	LICLDE	0	69.78	100,050	9.54
	LDCLDE	0	223.15	100,000	11.63
	SADE	0	112.18	100,000	15.06
	SACLDE	0	103.61	100,000	12.18
	SFLSDE	0	117.86	100,000	15.42
	SFLSCLDE	0	117.86	100,000	15.42
f_{20}	DE	0	0.21	100,000	7.49×10^{-2}
	RCLDE	0	57.16	100,000	0.11
	LICLDE	3	0.19	96,356	0.15
	LDCLDE	0	48.36	100,000	0.12
	SADE	97	3.52×10^{-4}	40,480	2.20×10^{-3}
	SACLDE	95	2.22×10^{-4}	47,632	1.26×10^{-3}
	SFLSDE	98	2.05×10^{-4}	42,331	1.42×10^{-3}
	SFLSCLDE	98	2.05×10^{-4}	42,331	1.42×10^{-3}
f_{21}	DE	100	8.94×10^{-6}	33,667	9.65×10^{-7}
	RCLDE	0	12.89	100,000	0.16
	LICLDE	100	8.91×10^{-6}	31,021	9.18×10^{-7}
	LDCLDE	0	16.50	100,000	0.32
	SADE	100	8.84×10^{-6}	11,735	1.08×10^{-6}
	SACLDE	100	8.90×10^{-6}	10,882	9.62×10^{-7}
	SFLSDE	100	9.01×10^{-6}	18,133	7.92×10^{-7}
	SFLSCLDE	100	9.01×10^{-6}	18,133	7.92×10^{-7}
f_{22}	DE	86	2.05×10^{-4}	22,795	2.87×10^{-4}
	RCLDE	100	6.58×10^{-5}	23,014	1.41×10^{-5}
	LICLDE	100	6.86×10^{-5}	18,575	2.28×10^{-5}
	LDCLDE	8	0.05	92,781	2.90×10^{-4}
	SADE	27	5.22×10^{-4}	79,045	2.78×10^{-4}
	SACLDE	59	3.30×10^{-4}	52,552	2.90×10^{-4}
	SFLSDE	18	5.63×10^{-4}	86,491	2.25×10^{-4}
	SFLSCLDE	18	5.63×10^{-4}	86,491	2.24×10^{-4}
f_{23}	DE	49	1.73×10^{-5}	52485	1.56×10^{-5}
	RCLDE	95	3.65×10^{-2}	39,227	3.35×10^{-6}
	LICLDE	55	1.54×10^{-5}	48,568	1.49×10^{-5}
	LDCLDE	34	0.41	67,450	4.96×10^{-6}
	SADE	44	1.79×10^{-5}	56,950	1.42×10^{-5}
	SACLDE	61	1.42×10^{-5}	40,456	1.52×10^{-5}
	SFLSDE	48	1.70×10^{-5}	53,162	1.51×10^{-5}
	SFLSCLDE	48	1.71×10^{-5}	53,162	1.52×10^{-5}
f_{24}	DE	0	0.57	100,000	0.11
	RCLDE	4	0.92	96,992	0.29
	LICLDE	13	0.53	94,705	0.28
	LDCLDE	0	3.48	100,000	0.06
	SADE	0	0.97	100,000	0.13
	SACLDE	0	0.98	100,000	0.15
	SFLSDE	0	0.43	100,000	0.13
	SFLSCLDE	0	0.44	100,000	0.13

Table 3 continued

Test problem	Algorithm	SR	ME	AFE	SD
f_{25}	DE	0	1.22×10^{-8}	100,000	6.97×10^{-9}
	RCLDE	100	7.21×10^{-16}	8,388	1.50×10^{-16}
	LICLDE	100	7.77×10^{-16}	6,496	1.74×10^{-16}
	LDCLDE	100	8.16×10^{-16}	33283	1.32×10^{-16}
	SADE	100	9.08×10^{-16}	34,998	6.96×10^{-17}
	SACLDE	100	7.31×10^{-16}	10,158	1.72×10^{-16}
	SFLSDE	100	9.09×10^{-16}	64,269	9.06×10^{-17}
	SFLSCLDE	100	9.09×10^{-16}	64,269	9.06×10^{-17}

Table 4 Results of the Student's t test

Test problem	Student's t test with DE			Student's t test with LICLDE	
	LICLDE	LDCLDE	RCLDE	LDCLDE	RCLDE
f_1	+	+	+	+	+
f_2	+	+	+	+	+
f_3	+	+	+	+	+
f_4	=	=	=	=	=
f_5	+	−	+	+	+
f_6	+	+	+	+	+
f_7	−	−	−	−	−
f_8	+	+	+	+	+
f_9	−	−	−	−	−
f_{10}	+	+	+	+	+
f_{11}	+	+	+	+	+
f_{12}	=	=	=	=	=
f_{13}	+	+	+	+	+
f_{14}	+	+	+	+	+
f_{15}	+	+	+	+	+
f_{16}	+	+	+	+	+
f_{17}	−	−	−	+	+
f_{18}	−	−	−	+	+
f_{19}	=	=	=	=	=
f_{20}	+	=	=	+	+
f_{21}	+	−	−	+	+
f_{22}	+	−	−	+	+
f_{23}	+	−	+	+	−
f_{24}	+	=	+	+	+
f_{25}	+	+	+	+	+

figure that strategy (2) $LICLDE$ is best among all mentioned strategies as interquartile range and Median are low for strategy (2) $LICLDE$.

In order to compare the consolidated performance of DE with $CLDE$ variants, the value of a performance index PI [11] is computed. This index gives a weighted importance to the success rate, the mean error as well as the average number of function evaluations. The value of this performance index for a computational algorithm under comparison is given by

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i)$$

where $\alpha_1^i = \frac{Sr^i}{Tr^i}$; $\alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i}, & \text{if } Sr^i > 0. \\ 0, & \text{if } Sr^i = 0. \end{cases}$; and $\alpha_3^i = \frac{Mo^i}{Ao^i}$

$i = 1, 2, \ldots, N_p$

- Sr^i is the number of successful runs of ith problem.
- Tr^i is the total number of runs of ith problem.

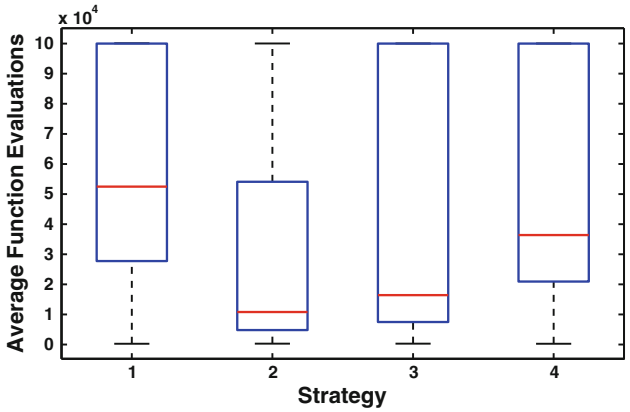

Fig. 1 *Boxplot graph* for average function evaluation: *1* DE, *2* $LICLDE$, *3* $RCLDE$ and *4* $LDCLDE$

- Mf^i = is the minimum of average number of function evaluations of successful runs used by all algorithms in obtaining the solution of ith problem.
- Af^i = is the average number of function evaluations of successful runs used by an algorithm in obtaining the solution of ith problem.
- Mo^i = is the minimum of mean error obtained by all the algorithms for the ith problem.
- Ao^i = is the mean error obtained by an algorithm for the ith problem.
- N_p = is the total number of problems analyzed.

k_1, k_2 and k_3 ($k_1 + k_2 + k_3 = 1$ and $0 \leq k_1, k_2, k_3 \leq 1$) are the weights assigned to success rate, average number of function evaluations of successful runs and mean error, respectively. From the above definition it is clear that PI is a function of k_1, k_2 and k_3. Since $k_1 + k_2 + k_3 = 1$ one of $k_i, i = 1, 2, 3$ could be eliminated to reduce the number of dependent variables from the expression of PI. We adopt the same methodology as given in [11], i.e. equal weights are assigned to two terms at a time in the PI expression. This way PI becomes a function of one variable. The resultant cases are as follows:

1. $k_1 = W, k_2 = k_3 = \frac{1-W}{2}, 0 \leq W \leq 1;$
2. $k_2 = W, k_1 = k_3 = \frac{1-W}{2}, 0 \leq W \leq 1;$
3. $k_3 = W, k_1 = k_2 = \frac{1-W}{2}, 0 \leq W \leq 1.$

The graphs corresponding to each of the cases (1), (2) and (3) are shown in Figs. 2, 3, and 4, respectively. In these figures the horizontal axis represents the weight W and the vertical axis represents the performance index PI.

In case (1), the average number of function evaluations of successful runs and mean error are given equal weights. PIs of all four algorithms ($LICLDE$, $RCLDE$, $LDCLDE$, DE) are superimposed in the Fig. 2 for comparison and to get a ranking of the performance of the four algorithms. It is observed that for differential evolution algorithm with linearly increasing cognitive learning factor ($LICLDE$), the value of PI is more than all the remaining two, i.e. DE with linearly decreasing cognitive learning factor ($LDCLDE$), and DE with random cognitive learning factor ($RCLDE$). The $CLDE$ performs in the order $LICLDE > RCLDE > LDCLDE > DE$.

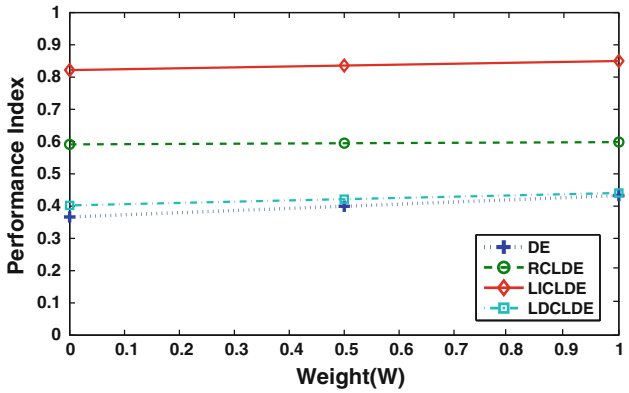

Fig. 2 Performance index for case (1)

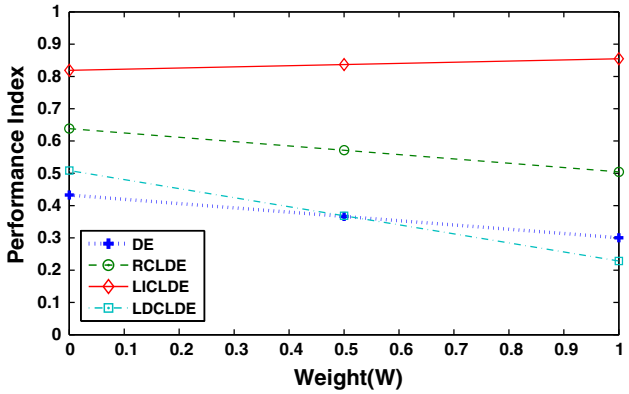

Fig. 3 Performance index for case (2)

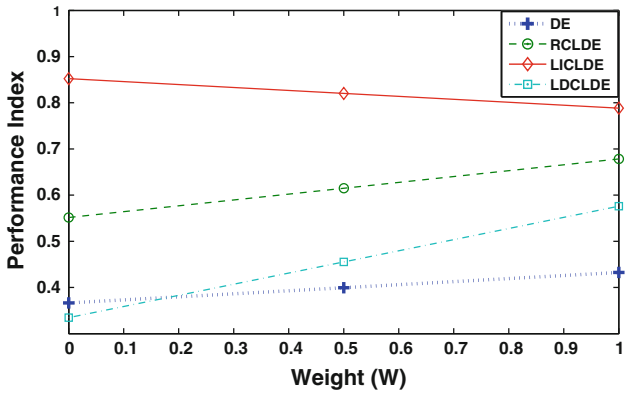

Fig. 4 Performance index for case (3)

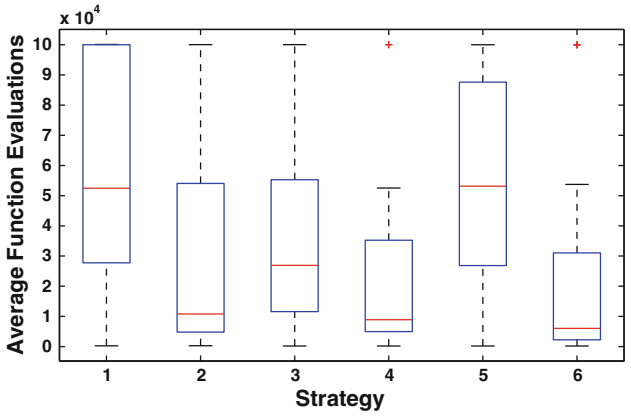

Fig. 5 *Boxplot graph* for average function evaluation. *1* DE, *2* $LICLDE$, *3* $SADE$, *4* $SACLDE$, *5* $SFLSDE$ and *6* $SFLSCLDE$

In case (2), equal weights are assigned to the success rate and average function evaluations of successful runs. From Fig. 3, it is clear that all $CLDE$ perform same as in case (1).
In case (3), equal weights are assigned to the success rate and mean error. Again the same conclusion is drawn from Fig. 4.

As an overall conclusion from PI it is concluded that the $LICLDE$ is better than the other four algorithm.

5.4 $SADE$ and $SFLSDE$ with cognitive learning factor

It is obvious from the Sect. 5.3 that the $CLDE$ performs better with the linearly increasing cognitive learning factor. The experimental findings support our theoretical suggestions that linearly increasing cognitive learning factor in DE should produce relatively better results.

Further, it will be interesting to investigate that whether linearly increasing cognitive learning factor improves the performance of some modified differential evolution algorithms. In this paper, linearly increasing CLF is tested with two modified DE algorithms: simulated annealing differential evolution $SADE$ [58] and scale factor local search differential evolution $SFLSDE$ [37]. The $SADE$ algorithm with $LICLF$ is denoted as $SACLDE$ and $SFLSDE$ with $LICLF$ is denoted as $SFLSCLDE$.

The experimental results are shown in Table 3. It can be observed that $LICLF$ improves the performance of $SADE$ and $SFLSDE$. More intensive comparative analysis using statistical tools; boxplots and performance index among DE, $LICLDE$, $SADE$, $SACLDE$, $SFLSDE$ and $SFLSCLDE$ have been carried out. Boxplots based on the data of average function evaluations for all these algorithms are shown in Fig. 5.

Following are the observations of boxplot analysis:

- By comparing DE with $LICLDE$, it is clear that interquartile range and median of $LICLDE$ is significantly less than that of DE.
- By comparing interquartile range and median of $SADE$ with $SACLDE$, it can be stated that performance of $SACLDE$ is significantly better than $SADE$.
- By comparing interquartile range and median of *SFLSDE* and *SFLSCLDE*, it is clear that *SFLSCLDE* is better than *SFLSDE*.

It is clear from the Boxplot analysis that after applying the cognitive learning factor C in the mutation operation of DE, $SADE$ and $SFLSDE$, the performance of these algorithms is significantly improved.

For the comparison of the performance between DE and $LICLDE$, $SADE$ and $SACLDE$, $SFLSDE$ and $SFLSCLDE$ the value of a performance index PI is computed. The PI is calculated by varying (i) success rate, (ii) function evaluation and (iii) mean error, for all the mentioned algorithms.The graphs corresponding to each of the cases (i), (ii) and (iii) are shown in Fig. 6a, b, and c, respectively.

By analyzing the Fig. 6, it is observed that for each case, PI of differential evaluation algorithm using cognitive learning factor is significantly higher than the corresponding differential evaluation algorithm. The pair wise performance order of PI for all the three cases is as follows:

- $LICLDE > DE$.
- $SACLDE > SADE$.
- $SFLSCLDE > SFLSDE$.

Therefore, we can say that the effect of cognitive learning factor is significant on the performance of differential evolution algorithm and some of its variants.

6 Application of $LICLDE$ in model order reduction (MOR) problem

Model order reduction (MOR) problem is studied in the branch of systems and control theory. In a real world situation, usually we get a system of very high order which is inappropriate for representing some properties that are important for effective use of the system. Model order reduction (MOR) problem deals with reduction of complexity of a dynamical system, while preserving their input–output behavior. Although many conventional approaches [5,6,10,16,24,29] of model order reduction guarantee the stability of the reduced order model but sometimes the model may turn out to be non-minimum phase. Therefore to obtain better reduced order models, the use of some kind of optimization is necessary by itself and in combination with other techniques. Error minimization is one of the popular techniques for model order reduction of continuous time systems. In this technique, lower order model is obtained by minimizing an error function constructed from the time responses (or alternatively frequency responses) of the system and reduced

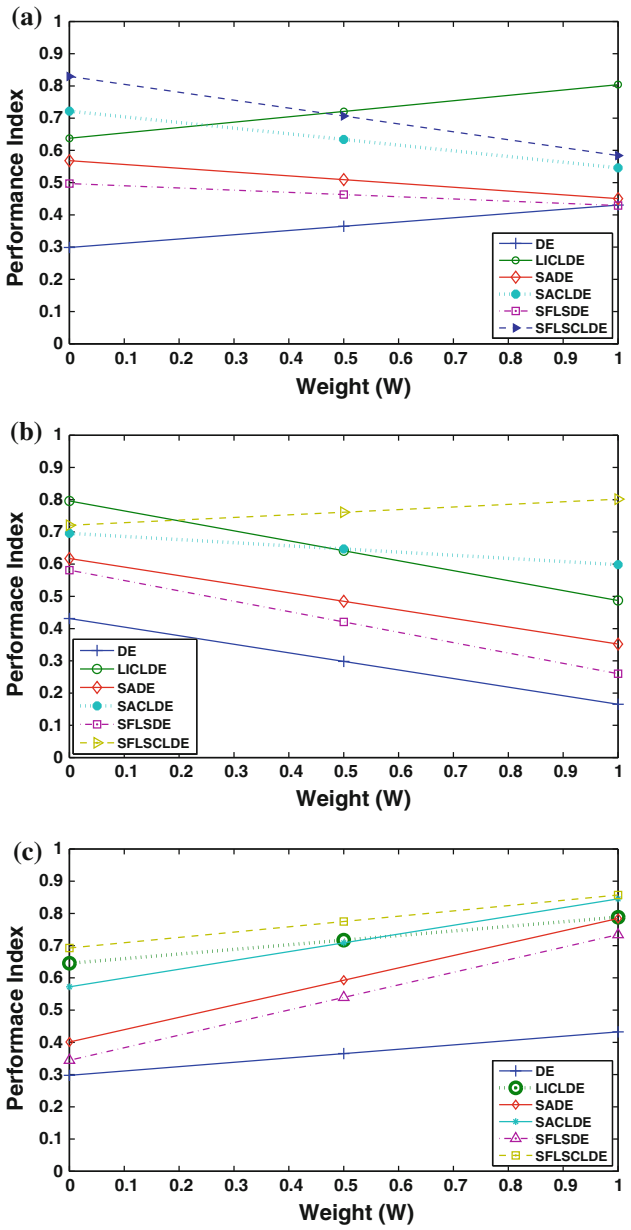

Fig. 6 Performance index: **a** for case (i), **b** for case (ii) and **c** for case (iii)

order model. Various error criteria are available for minimization such as integral square error (ISE), integral time square errors (ITSE), integral of absolute error (IAE) and integral time absolute errors (ITAE). Out of these ISE is used most frequently.

All the methods discussed in the literature [18,19,22,25,33,36,50,51,56], have considered to minimize integral squared error (ISE) between the transient responses of original higher order model and the reduced order model pertaining to a unit step input. But in this paper, minimization is carried out based on both integral squared error (ISE) and impulse response energy (IRE).

6.1 MOR as an optimization problem

Consider an nth order linear time invariant dynamic SISO system given by

$$G(s) = \frac{N(s)}{D(s)} = \frac{\sum_{i=0}^{n-1} a_i s^i}{\sum_{i=0}^{n} b_i s^i} \tag{3}$$

where a_i and b_i are known constants.
The problem is to find a rth order reduced model in the transfer function form $R(s)$, where $r < n$ represented by Eq. (4), such that the reduced model retains the important characteristics of the original system and approximates its step response as closely as possible for the same type of inputs with minimum integral square error.

$$R(s) = \frac{N_r(s)}{D_r(s)} = \frac{\sum_{i=0}^{r-1} a_i' s^i}{\sum_{i=0}^{r} b_i' s^i} \tag{4}$$

where a_i' and b_i' are unknown constants.
Mathematically, the integral square error of step responses of the original and the reduced system can be expressed by error index às [15],

$$J = \int_0^\infty [y(t) - y_r(t)]^2 \, dt. \tag{5}$$

where $y(t)$ is the unit step response of the original system and $y_r(t)$ is the unit step response of the reduced system. This error index ìs the function of unknown coefficients a_i' and b_i'. The aim is to determine the coefficients a_i' and b_i' of reduced order model so that the error index ìs minimized.

6.2 Modified objective function for MOR

In this paper, minimization is carried out based on both ISE and IRE. The low order model is obtained by minimizing an error function, constructed from minimization of the integral square error (ISE) between the transient responses of original higher order model and the reduced low order model pertaining to a unit step input as well as minimization of the difference between the high order model's impulse response energy (IRE) and the reduced low order IRE.

The impulse response energy (IRE) for the original and the various reduced order models is given by:

$$IRE = \int_0^\infty g(t)^2 \, dt. \tag{6}$$

where, $g(t)$ is the impulse response of the system.

Therefore, in this paper, both, ISE and IRE, are used to construct the objective function for minimizing the ISE and difference between IRE of high order model and reduced order model. The following modified objective function is constructed to carry out the results.

Table 5 List of MOR problem examples

S. no.	Source	Original model
1	Shamash [48]	$G_1(s) = \frac{18s^7+514s^6+5{,}982s^5+36{,}380s^4+122{,}664s^3+222{,}088s^2+185{,}760s+40{,}320}{s^8+36s^7+546s^6+4{,}536s^5+22{,}449s^4+67{,}284s^3+118{,}124s^2+109{,}584s+40{,}320}$
2	Lucas [30]	$G_2(s) = \frac{8{,}169.13s^3+50{,}664.97s^2+9{,}984.32s+500}{100s^4+10{,}520s^3+52{,}101s^2+10{,}105s+500}$
3	Pal [42]	$G_3(s) = \frac{s+4}{s^4+19s^3+113s^2+245s+150}$
4	Aguirre [2]	$G_4(s) = \frac{4.269s^3+5.10s^2+3.9672s+0.9567}{4.3992s^4+9.0635s^3+8.021s^2+5.362s+1}$
5	Eydgahi et al. [13]	$G_5(s) = \frac{s^4+35s^3+291s^2+1{,}093s+1{,}700}{s^9+9s^8+66s^7+294s^6+1{,}029s^5+2{,}541s^4+4{,}684s^3+5{,}856s^2+4{,}629s+17{,}00}$

Table 6 Comparison of the methods for example 1

Method of order reduction	Reduced models; $R_1(s)$	ISE	IRE
Original	$G_1(s)$	–	21.740
LICLDE	$\frac{17.203s+5.3633}{s^2+6.9298s+5.3633}$	0.8×10^{-3}	21.74
DE	$\frac{20s+5.6158}{s^2+9.2566s+5.6158}$	0.3729×10^{-1}	21.908
Pade approximation	$\frac{15.1s+4.821}{s^2+5.993s+4.821}$	1.6177	19.426
Routh approximation	$\frac{1.99s+0.4318}{s^2+1.174s+0.4318}$	1.9313	1.8705
Gutman et al. [17]	$\frac{4[133,747,200s+203,212,800]}{85,049,280s^2+552,303,360s+812,851,200}$	8.8160	4.3426
Hutton and Friedland [21]	$\frac{1.98955s+0.43184}{s^2+1.17368s+0.43184}$	18.3848	1.9868
Krishnamurthy and Sheshadri [24]	$\frac{155,658.6152s+40,320}{65,520s^2+75,600s+40,320}$	17.5345	2.8871
Mittal et al. [34]	$\frac{7.0908s+1.9906}{s^2+3s+2}$	6.9159	9.7906
Mukherjee and Mishra [36]	$\frac{7.0903s+1.9907}{s^2+3s+2}$	6.9165	9.7893
Mukherjee et al. [35]	$\frac{11.3909s+4.4357}{s^2+4.2122s+4.4357}$	2.1629	18.1060
Pal [41]	$\frac{151,776.576s+40,320}{65,520s^2+75,600s+40,320}$	17.6566	2.7581
Prasad and Pal [43]	$\frac{17.98561s+500}{s^2+13.24571s+500}$	18.4299	34.1223
Shamash [48]	$\frac{6.7786s+2}{s^2+3s+2}$	7.3183	8.9823

Table 7 Comparison of the methods for example 2

Method of order reduction	Reduced models; $R_2(s)$	ISE	IRE
Original	$G_2(s)$	–	34.069
LICLDE	$\frac{103.3218182s+867.893179}{s^2+169.4059231s+867.893179}$	$0.36228741 \times 10^{-2}$	34.069918
DE	$\frac{220.8190s+35011.744}{s^2+1229.4502s+35011.744}$	0.4437568×10^{-2}	34.069218
Singh [49]	$\frac{93.7562s+1}{s^2+100.10s+10}$	0.8964×10^{-2}	43.957
Pade approximation	$\frac{23.18s+2.36}{s^2+23.75s+2.36}$	0.46005×10^{-2}	11.362
Routh approximation	$\frac{0.1936s+0.009694}{s^2+0.1959s+0.009694}$	2.3808	0.12041
Gutman et al. [17]	$\frac{0.19163s+0.00959}{s^2+0.19395s+0.00959}$	2.4056	0.11939
Chen et al. [6]	$\frac{0.38201s+0.05758}{s^2+0.58185s+0.05758}$	1.2934	0.17488
Marshall [31]	$\frac{83.3333s+499.9998}{s^2+105s+500}$	0.193×10^{-2}	35.450

$$objective\_value = |ISE| + \frac{|IRE_R - IRE_O|}{IRE_R + IRE_O} \tag{7}$$

where ISE is an integral squared error of difference between the responses given by Eq. (5), IRE_O is the impulse response energy of the original high order model and IRE_R is the impulse response energy of the reduced order model. The advantage of this modified objective function is that it minimizes ISE as well as the differences of IRE of both the models (high order and reduced order).

6.3 Experimental results and numerical examples

Total five examples are taken into consideration in this section (see Table 5).

The best solution obtained out of 100 runs is reported as the global optimal solution. The reported solutions are in the form of step and impulse responses. The results obtained by DE are compared with that of $LICLDE$ and other stochastic as well as deterministic methods.

Tables 6, 7, 8, 9 and 10 present the original and the reduced systems for examples 1, 2, 3, 4, and 5, respectively. In these

Table 8 Comparison of the methods for example 3

Method of order reduction	Reduced models; $R_3(s)$	ISE	IRE
Original	$G_3(s)$	–	0.26938×10^{-3}
LICLDE	$\frac{-0.0195s+0.2884}{s^2+14.9813s+10.82}$	0.43168×10^{-5}	0.27×10^{-3}
DE	$\frac{0.0296s+0.2175}{s^2+12.3952s+8.156}$	$0.1451930426 \times 10^{-4}$	0.27×10^{-3}
Singh [49]	$\frac{-494.596s+405.48}{150s^2+2487s+15205.5}$	0.2856×10^{-2}	0.2476×10^{-3}
Pade approximation	$\frac{-0.005017s+0.08247}{s^2+4.09s+3.093}$	∞	0.27192×10^{-3}
Routh approximation	$\frac{0.009865s+0.03946}{s^2+2.417s+1.48}$	∞	0.23777×10^{-3}

Table 9 Comparison of the methods for example 4

Method of order reduction	Reduced models; $R_4(s)$	ISE	IRE
Original	$G_4(s)$	–	0.54536
LICLDE	$\frac{0.7853s+2.949}{s^2+3.1515s+3.0823}$	0.338×10^{-1}	0.54538
DE	$\frac{1.0755s+9.567}{s^2+9.4527s+10}$	0.364×10^{-1}	0.54535
Singh [49]	$\frac{4.0056s+0.9567}{8.021s^2+5.362s+1}$	0.22372	0.27187
Pade approximation	$\frac{1.869s+0.5585}{s^2+2.663s+0.5838}$	∞	0.75619
Routh approximation	$\frac{0.6267s+0.1511}{s^2+0.847s+0.158}$	∞	0.31715

Table 10 Comparison of the methods for example 5

Method of order reduction	Reduced models; $R_5(s)$	ISE	IRE
Original	$G_5(s)$, see [13]	–	0.47021
LICLDE	$\frac{-0.6372s+1.0885}{s^2+1.5839s+1.0885}$	0.209×10^{-1}	0.4718
DE	$\frac{-0.8068s+1.3083}{s^2+2.0221s+1.3083}$	0.302×10^{-1}	0.4845
Pade approximation	$\frac{-0.8153s+1.392}{s^2+2.081s+1.392}$	0.330×10^{-1}	0.49414
Routh approximation	$\frac{0.2643s+0.411}{s^2+1.119s+0.411}$	0.131	0.21486

tables results obtained by $LICLDE$ are compared with that of the basic DE, Pade approximation method, Routh approximation method and other earlier reported results. Corresponding unit step responses of the original and the reduced systems using $LICLDE$, DE, Pade approximation and Routh approximation are shown in Figs. 7, 9, 11, 13 and 15, respectively. The impulse responses of the original and the reduced systems using $LICLDE$, DE, Pade approximation and Routh approximation are shown in Figs. 8, 10, 12, 14 and 16, respectively.

It can be observed that for examples 1, 3 and 5, $ISEs$ obtained by $LICLDE$ are significantly less than that of other methods. Also for these examples, $IREs$ of the reduced models obtained by $LICLDE$ are most close to that of the originals. The ISE for examples 3 and 4 obtained by Pade and Routh approximation method are coming out to be infinity because of the steady state error between the original and the reduced system. For example 2, It may be seen that in Table 7, $LICLDE$ provides least value of the ISE except [31] whereas IRE is most close to that of the originals. For example 4, the ISE is still least by $LICLDE$ but not significantly as compared to DE. Again IRE obtained by DE and $LICLDE$ are almost same for example 4. It may also be seen that the steady state responses of the original and the reduced

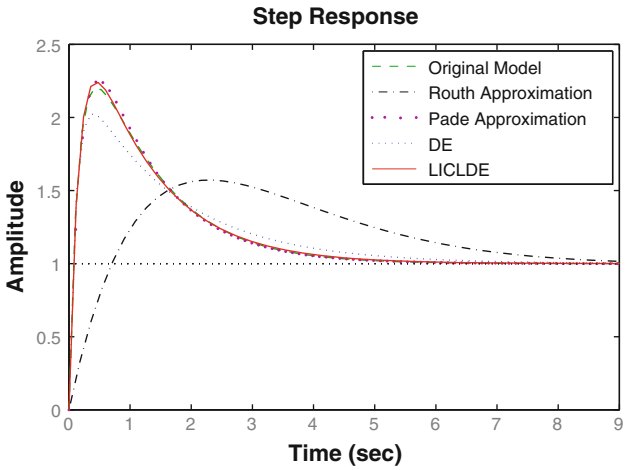

Fig. 7 Comparison of step responses for example 1

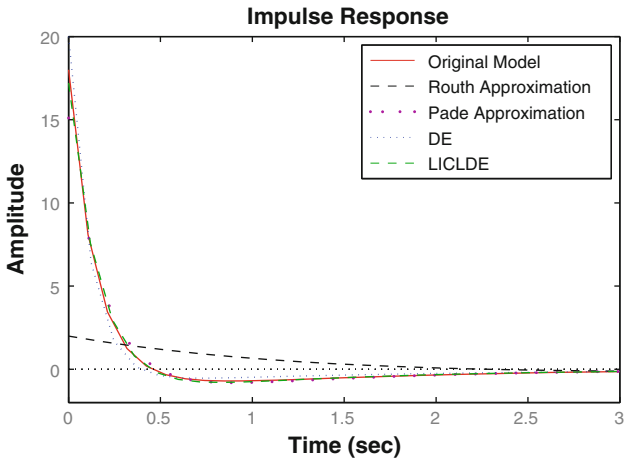

Fig. 8 Comparison of impulse responses for example 1

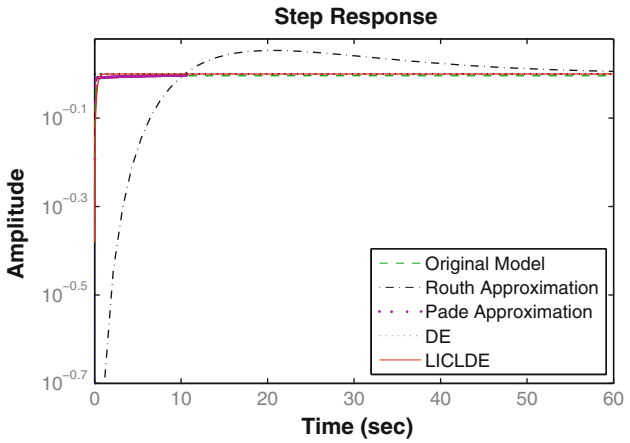

Fig. 9 Comparison of step responses for example 2

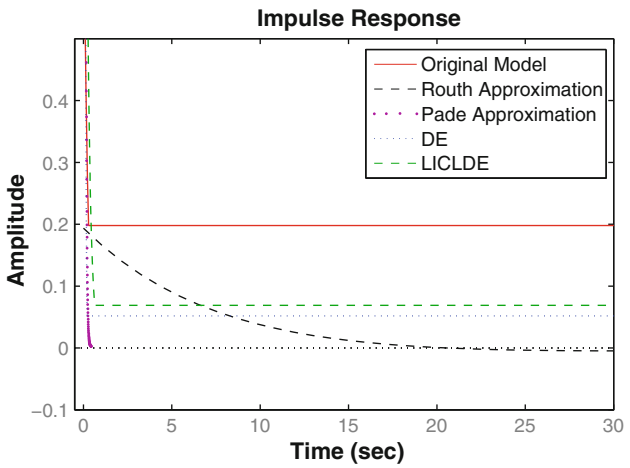

Fig. 10 Comparison of impulse responses for example 2

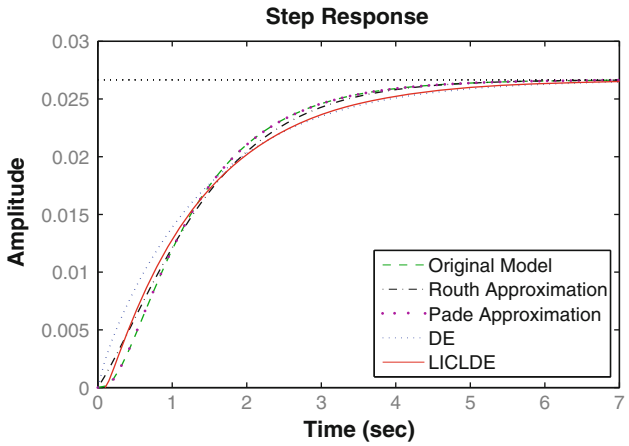

Fig. 11 Comparison of step responses for example 3

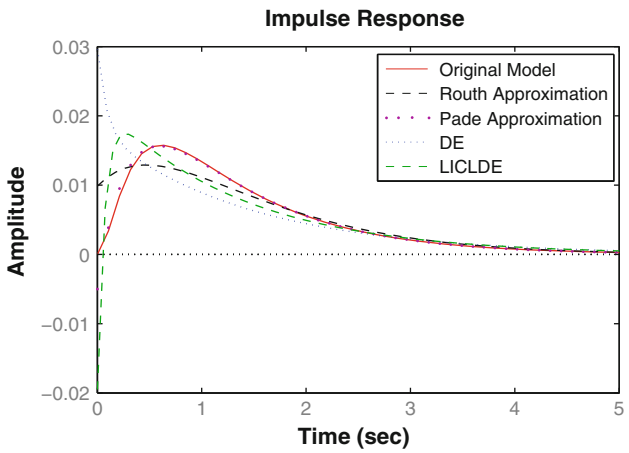

Fig. 12 Comparison of impulse responses for example 3

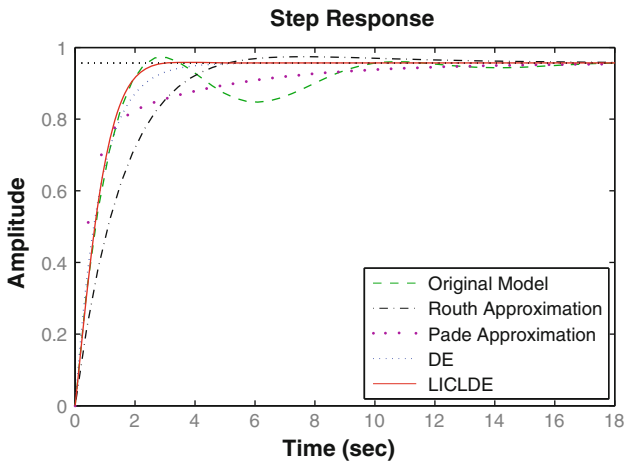

Fig. 13 Comparison of step responses for example 4

order models by $LICLDE$ are exactly matching while the transient response matching is also very close as compared to other methods. Thus these examples establish the superiority of $LICLDE$ over other methods for this problem.

Overall, $LICLDE$ performance is superior than the basic DE and other deterministic as well as probabilistic methods. Thus, $LICLDE$ may be treated as a robust method to solve MOR problem.

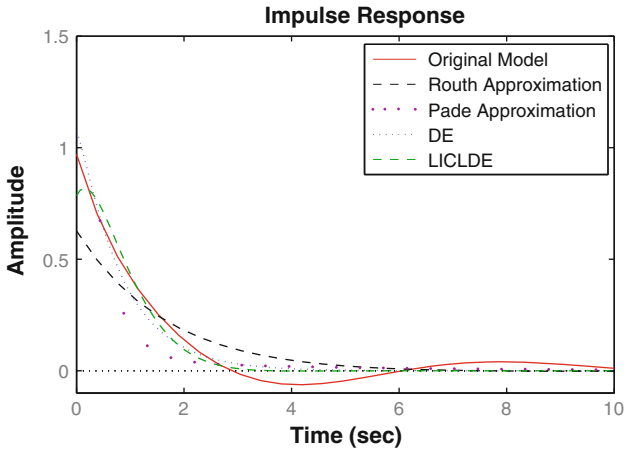

Fig. 14 Comparison of impulse responses for example 4

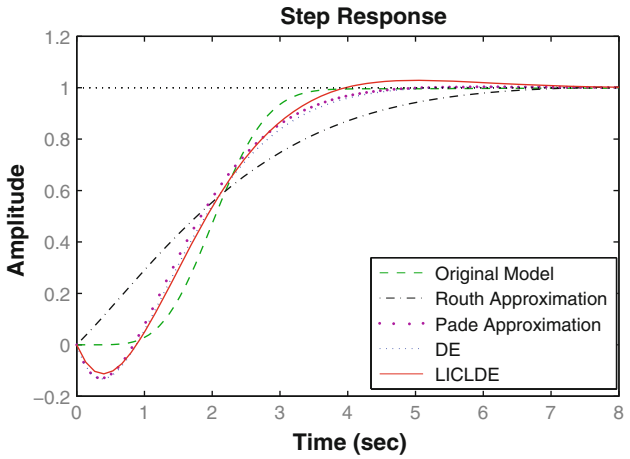

Fig. 15 Comparison of step responses for example 5

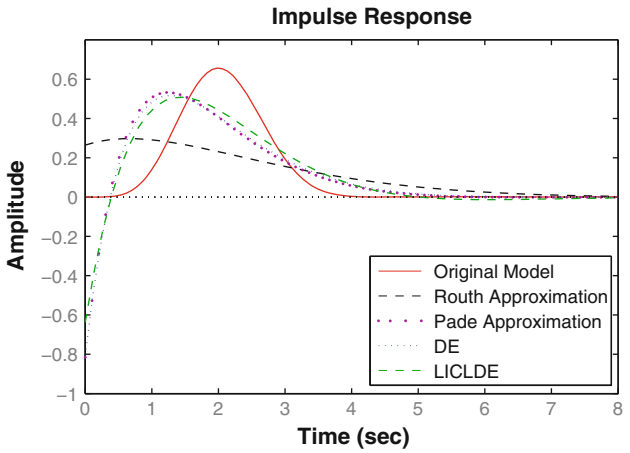

Fig. 16 Comparison of impulse responses for example 5

7 Conclusion

In this paper, basic differential evolution algorithm is improved by introducing a new control parameter (cognitive learning factor) in DE search procedure. With the help of experiments over test problems it is showed that the reliability (due to success rate), efficiency (due to average number of function evaluations) and accuracy (due to mean objective function value) of basic as well as two modified (SADE and SFLSDE) versions of DE algorithm with this new control parameter is higher than that of its original versions. The modified DE so obtained is named as Cognitive Learning DE (CLDE).

Basic DE and CLDE algorithms are then successfully applied to model order reduction problem for single input and single output system. The novelty in this application is the newly designed objective function for this problem. The objective function takes care of both ISE and IRE simultaneously. To validate and to show the versatility of DE and CLDE, five systems of different orders are reduced using these algorithms. It is showed that CLDE outperforms in terms of ISE and IRE when compared to the results obtained by other algorithms and other formulation of objective function.

Based on this study, it is concluded that CLDE particularly LICLDE is a better candidate in the field of nature inspired algorithms for function optimization.

The future scope of this work is the implementation of cognitive learning factor to other biologically inspired algorithms.

References

1. Abbass HA (2002) The self-adaptive Pareto differential evolution algorithm. In: Proceedings of the 2002 congress on evolutionary computation 2002. CEC'02, vol 1. IEEE, NY, pp 831–836
2. Aguirre LA (1992) The least squares padé method for model reduction. Int J Syst Sci 23(10):1559–1570
3. 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
4. Chakraborty UK (2008) Advances in differential evolution. Springer, Berlin
5. Chen CF, Shieh LS (1968) A novel approach to linear model simplification. Int J Control 8(6):561–570
6. Chen TC, Chang CY, Han KW (1979) Reduction of transfer functions by the stability-equation method. J Franklin Inst 308(4):389–404
7. Croarkin C, Tobias P (2010) Nist/sematech e-handbook of statistical methods. Retrieved 1 March 2010
8. Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood-based mutation operator. IEEE Trans Evol Comput 13(3):526–553
9. Das S, Konar A (2006) Two-dimensional iir filter design with modern search heuristics: a comparative study. Int J Comput Intell Appl 6(3):329–355
10. Davison JE (1966) A method for simplifying linear dynamic systems. IEEE Trans Autom Control AC-11 1:93–101
11. Thakur M, Deep K (2007) A new crossover operator for real coded genetic algorithms. Appl Math Comput 188(1):895–911

12. Engelbrecht AP (2007) Computational intelligence: an introduction. Wiley, London
13. Eydgahi A, Shore E, Anne P, Habibi J, Moshiri B (2003) A matlab toolbox for teaching model order reduction techniques. In: International conference on engineering education, Valencia, Spain, pp 1–7
14. Gamperle R, Muller SD, Koumoutsakos A (2002) A parameter study for differential evolution. Adv Intell Syst Fuzzy Syst Evol Comput 10:293–298
15. Gopal M (2002) Control systems: principles and design. Tata McGraw-Hill, NY
16. Gustafson RD (1966) A paper and pencil control system design. Trans ASME J Basic Eng 329–336
17. Gutman PO, Mannerfelt CF, Molander P (1982) Contributions to the model reduction problem. IEEE Trans Autom Control AC-27 2:454–455
18. Hickin J, Sinha NK (1976) Reduction of linear system by canonical forms. Electron Lett 12(21):551–553
19. Hickin J, Sinha NK (1978) Canonical forms for aggregated models. Int J Control 27(3):473–485
20. Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
21. Hutton M, Friedland B (1975) Routh approximations for reducing order of linear, time-invariant systems. IEEE Trans Autom Control 20(3):329–337
22. Hwang C (1984) Mixed method of Routh and ISE criterion approaches for reduced-order modeling of continuous-time systems. J Dyn Syst Meas Control 106:353
23. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, 1995, vol 4. IEEE, NY, pp 1942–1948
24. Krishnamurthy V, Seshadri V (1978) Model reduction using the Routh stability criterion. IEEE Trans Autom Control 23(4):729–731
25. Lamba SS, Gorez R, Bandyopadhyay B (1988) New reduction technique by step error minimization for multivariable systems. Int J Syst Sci 19(6):999–1009
26. Lampinen J, Zelinka I (2000) On stagnation of the differential evolution algorithm. In: Proceedings of MENDEL. Citeseer, pp 76–83
27. Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Comput Fusion Found Methodol Appl 9(6):448–462
28. Liu PK, Wang FS (2008) Inverse problems of biological systems using multi-objective optimization. J Chin Inst Chem Eng 39(5):399–406
29. Lucas TN (1983) Factor division: a useful algorithm in model reduction. In: IEE Proceedings of the control theory and applications, vol 130. IET, pp 362–364
30. Lucas TN (1986) Continued-fraction expansion about two or more points: a flexible approach to linear system reduction. J Franklin Inst 321(1):49–60
31. Marshall S (1983) Comments on viability of methods for generating stable reduced order models. IEEE Trans Autom Control 28(5):630–631
32. Mezura-Montes E, Velázquez-Reyes J, Coello Coello CA (2006) A comparative study of differential evolution variants for global optimization. In: Proceedings of the 8th annual conference on genetic and evolutionary computation. ACM, pp 485–492
33. Mishra RN, Wilson DA (1980) A new algorithm for optimal reduction of multivariable systems. Int J Control 31(3):443–466
34. Mittal AK, Prasad R, Sharma SP (2004) Reduction of linear dynamic systems using an error minimization technique. J Inst Eng India IE(I) J EL 84:201–206
35. Mukherjee S et al (2005) Model order reduction using response-matching technique. J Franklin Inst 342(5):503–519
36. Mukherjee S, Mishra RN (1987) Order reduction of linear systems using an error minimization technique. J Franklin Inst 323(1):23–32
37. Neri F, Tirronen V (2009) Scale factor local search in differential evolution. Memetic Comput 1(2):153–171
38. Noman N, Iba H (2005) Enhancing differential evolution performance with local search for high dimensional function optimization. In: Proceedings of the 2005 conference on genetic and evolutionary computation. ACM, pp 967–974
39. Omran M, Salman A, Engelbrecht A (2005) Self-adaptive differential evolution. Comput Intell Secur, pp 192–199
40. Omran MGH, Engelbrecht AP, Salman A (2005) Differential evolution methods for unsupervised image classification. In: The 2005 IEEE congress on evolutionary computation, 2005, vol 2. IEEE, NY, pp 966–973
41. Pal J (1979) Stable reduced-order Pade approximants using the Routh-Hurwitz array. Electron Lett 15(8):225–226
42. Pal J (1986) An algorithmic method for the simplification of linear dynamic scalar systems. Int J Control 43(1):257–269
43. Prasad R, Pal J (1991) Stable reduction of linear systems by continued fractions. J Inst Eng India Part EL Electr Eng Div 72:113
44. Price KV (1996) Differential evolution: a fast and simple numerical optimizer. In: 1996 biennial conference of the North American fuzzy information processing society, 1996, NAFIPS. IEEE, NY, pp 524–527
45. Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin
46. 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
47. Rogalsky T, Kocabiyik S, Derksen RW (2000) Differential evolution in aerodynamic optimization. Can Aeronaut Space J 46(4):183–190
48. Shamash Y (1975) Linear system reduction using Pade approximation to allow retention of dominant modes. Int J Control 21(2):257–272
49. Singh N (2007) Reduced order modeling and controller design. PhD thesis. Indian Institute of Technology Roorkee, India
50. Sinha NK, Bereznai GT (1971) Optimum approximation of high-order systems by low-order models. Int J Control 14(5):951–959
51. Sinha NK, Pille W (1971) A new method for reduction of dynamic systems. Int J Control 14(1):s111–s118
52. Storn R (1996) On the usage of differential evolution for function optimization. In: 1996 biennial conference of the North American fuzzy information processing society, 1996, NAFIPS. IEEE, NY, pp 519–523
53. Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Int Comput Sci Inst 1:1–12
54. Teo J (2006) Exploring dynamic self-adaptive populations in differential evolution. Soft Comput Fusion Found Method Appl 10(8):673–686
55. Vesterstrom J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Congress on evolutionary computation, 2004. CEC2004, vol 2. IEEE, NY, pp 1980–1987
56. Vilbe P, Calvez LC (1990) On order reduction of linear systems using an error minimization technique. J Franklin Inst 327(3):513–514
57. Williamson DF, Parker RA, Kendrick JS (1989) The box plot: a simple visual method to interpret data. Ann Internal Med 110(11):916
58. Yan JY, Ling Q, Sun Q (2006) A differential evolution with simulated annealing updating method. In: International

conference on machine learning and cybernetics. IEEE, NY, pp 2103–2106
59. Zaharie D (2003) Control of population diversity and adaptation in differential evolution algorithms. In: Proc of MENDEL, pp 41–46
60. Zaharie D, Petcu D (2004) Adaptive Pareto differential evolution and its parallelization. Parallel Process Appl Math 3019:261–268